LilyPond

Le système de gravure musicale

Manuel de notation

L'équipe de développement de LilyPond

Ce document constitue le manuel de notation de GNU LilyPond 2.23.2. Sa lecture requiert une familiarité avec le contenu présenté dans le Section "Manuel d'initiation" dans *Manuel d'initiation*.

Pour connaître la place qu'occupe ce manuel dans la documentation, consultez la page Section "Manuels" dans *Informations générales*.

Si vous ne disposez pas de certains manuels, la documentation complète se trouve sur `https://lilypond.org/`.

Copyright © 1998–2021 par les auteurs. *The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.*

La traduction de la notice de droits d'auteur ci-dessous vise à faciliter sa compréhension par le lecteur non anglophone, mais seule la notice en anglais a valeur légale.

> Vous avez le droit de copier, distribuer et/ou modifier ce document selon les termes de la Licence GNU de documentation libre, version 1.1 ou tout autre version ultérieure publiée par la Free Software Foundation, "sans aucune section invariante". Une copie de la licence est fournie à la section "Licence GNU de documentation libre".
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled "GNU Free Documentation License".

Pour LilyPond version 2.23.2

Table des matières

1 Notation musicale générale 1

1.1 Hauteurs 1

1.1.1 Écriture des hauteurs de note 1

Hauteurs avec octave absolue 1

Octaves relatives 2

Altérations 6

Nom des notes dans d'autres langues 8

1.1.2 Modification de plusieurs hauteurs 10

Vérifications d'octave 10

Transposition 11

Inversion 14

Rétrogradation 15

Transformations modales 15

1.1.3 Gravure des hauteurs 18

Clefs 18

Armure 22

Marques d'octaviation 25

Instruments transpositeurs 28

Altérations accidentelles automatiques 29

Ambitus 37

1.1.4 Têtes de note 40

Têtes de note spécifiques 40

Têtes de note avec nom de note 41

Têtes de note à forme variable 43

Improvisation 47

1.2 Rythme 47

1.2.1 Écriture du rythme 48

Durées 48

N-olets 51

Changement d'échelle des durées 56

Liaisons de prolongation 57

1.2.2 Écriture des silences 61

Silences 61

Silences invisibles 63

Silences valant une mesure 65

1.2.3 Gravure du rythme 69

Métrique 69

Indication métronomique 73

Levées 77

Musique sans métrique 78

Notation polymétrique 80

Découpage automatique des notes 83

Gravure de lignes rythmiques 84

1.2.4 Barres de ligature 87

Barres de ligature automatiques 87

Définition des règles de ligature automatique 90

Barres de ligature manuelles 99

Liens de croches en soufflet 102

1.2.5 Mesures 102
Barres de mesure 102
Numéros de mesure 110
Vérification des limites et numéros de mesure 116
Indications de repère 117
1.2.6 Fonctionnalités rythmiques particulières 119
Notes d'ornement 119
Alignement et cadences 125
Gestion du temps 125
1.3 Signes d'interprétation 126
1.3.1 Signes d'interprétation attachés à des notes 127
Articulations et ornements 127
Nuances 129
Personnalisation des indications de nuance 136
1.3.2 Signes d'interprétation sous forme de courbe 138
Liaisons d'articulation 139
Liaisons de phrasé 141
Signes de respiration 143
Chutes et sauts 145
1.3.3 Signes d'interprétation sous forme de ligne 146
Glissando 146
Arpèges 150
Trilles 154
1.4 Répétitions et reprises 156
1.4.1 Répétition d'un long passage 157
Répétitions explicites 157
Répétitions simples 158
Fins alternatives 159
Autres aspects des fragments répétés 159
Barre de *segno* 161
Indications de reprise manuelles 166
1.4.2 Autres types de répétition 167
Répétitions de mesure 168
Répétitions en trémolo 170
1.5 Notes simultanées 172
1.5.1 Monophonie 173
Notes en accords 173
Répétition d'accords 175
Expressions simultanées 177
Clusters 178
1.5.2 Plusieurs voix 179
Polyphonie sur une portée 179
Styles de voix 182
Résolution des collisions 183
Fusion de silences 188
Regroupement automatique de parties 189
Saisie de musique en parallèle 194
1.6 Notation sur la portée 197
1.6.1 Gravure des portées 197
Initialisation de nouvelles portées 197
Regroupement de portées 199
Imbrication de regroupements de portées 202
Séparation des systèmes 204
1.6.2 Modification de portées individuelles 205

Symbole de la portée 205
Portées d'ossia 209
Masquage de portées 212
1.6.3 Écriture de parties séparées 217
Noms d'instrument 217
Citation d'autres voix 221
Mise en forme d'une citation 224
Compression de mesures vides 230
1.7 Annotations éditoriales 232
1.7.1 Dans la portée 232
Indication de la taille de fonte musicale 232
Doigtés 236
Glissement de doigt 239
Dictée à trous 241
Coloration d'objets 242
Parenthèses 244
Hampes 245
1.7.2 Hors de la portée 246
Nom des notes 246
Info-bulle 248
Quadrillage temporel 249
Crochets d'analyse 251
1.8 Texte 253
1.8.1 Ajout de texte 254
Vue d'ensemble des objets textuels 254
Commentaires textuels 256
Indication textuelle avec extension 257
Indications textuelles 259
Texte indépendant 262
1.8.2 Mise en forme du texte 263
Introduction au formatage de texte 263
Sélection de la fonte et de la taille 265
Alignement du texte 268
Éléments graphiques dans du texte formaté 273
Notation musicale dans du texte formaté 275
Texte avec sauts de page 278
1.8.3 Fontes 279
Localisation des fontes 279
Familles de fontes 279
Fonctionnalités des fontes 281
Attribution d'une fonte en particulier 282
Choix des fontes par défaut 284
Fontes musicales 286

2 Notation spécialisée 287

2.1 Musique vocale 287
2.1.1 Vue d'ensemble de la musique vocale 287
Références en matière de musique vocale 287
Saisie des paroles 288
Alignement des paroles sur la mélodie 289
Durée automatique des syllabes 291
Durée explicite des syllabes 294
Plusieurs syllabes sur une note 295
Plusieurs notes pour une même syllabe 296

Traits d'union et de prolongation 299
Changement graduel de voyelle 299
2.1.2 Situations particulières en matière de paroles 300
Travail avec des paroles et variables 300
Positionnement vertical des paroles 302
Positionnement horizontal des syllabes 306
Paroles et reprises 308
Paroles alternatives 316
Polyphonie et paroles communes 318
2.1.3 Couplets 319
Numérotation des couplets 319
Indication de nuance dans les couplets 320
Indication du personnage et couplets 321
Rythme différent selon le couplet 321
Paroles en fin de partition 324
Paroles sur plusieurs colonnes en fin de partition 326
2.1.4 Chansons 327
Références en matière de chanson 328
Feuille de chant 328
2.1.5 Chorale 329
Références en matière de chorale 329
Mise en forme d'une partition chorale 330
2.1.6 Opéras et musiques de scène 332
Références en matière d'opéra et musique de scène 332
Indication du rôle 333
Citation-repère 335
Musique parlée 339
Dialogue et musique 339
2.1.7 Chants liturgiques 340
Références en matière de chant liturgique 340
Cantiques et hymnes 340
Psalmodie 347
Mesure incomplète et musique liturgique 350
2.1.8 Musique vocale ancienne 353
2.2 Instruments utilisant des portées multiples 353
2.2.1 Vue d'ensemble des claviers 354
Généralités sur les instruments à clavier 354
Changement de portée manuel 355
Changement de portée automatique 357
Lignes de changement de portée 359
2.2.2 Piano 362
Pédales de piano 362
2.2.3 Accordéon 363
Symboles de jeux 363
2.2.4 Harpe 364
Généralités sur la harpe 364
Pédales de harpe 364
2.3 Cordes non frettées 365
2.3.1 Vue d'ensemble de la notation pour cordes non frettées 366
Références en matière de cordes non frettées 366
Indications d'archet 366
Harmoniques 367
Snap (Bartók) pizzicato 368
2.4 Instruments à cordes frettées 368

2.4.1 Vue d'ensemble des cordes frettées 369
Références en matière de cordes frettées 369
Indications du numéro de corde 370
Tablatures par défaut 371
Tablatures personnalisées 389
Tablatures sous forme d'étiquette 393
Tablatures prédéfinies 403
Tablatures automatiques 414
Doigtés pour la main droite 417
2.4.2 Guitare 418
Indication de la position et du barré 418
Indication des harmoniques et notes étouffées 419
Indication de *power chord* 420
2.4.3 Banjo 421
Tablatures pour banjo 421
2.4.4 Luth 422
Tablatures pour luth 422
2.5 Percussions 423
2.5.1 Vue d'ensemble des percussions 423
Références en matière de notation pour percussions 423
Notation de base pour percussions 423
Roulements de tambour 424
Hauteurs en percussions 425
Portées de percussion 425
Personnalisation de portées de percussion 428
Notes fantômes 430
2.6 Instruments à vent 431
2.6.1 Vue d'ensemble des instruments à vent 431
Références en matière d'instruments à vent 431
Doigtés pour vents 432
2.6.2 Cornemuse 434
Définitions pour la cornemuse 434
Exemple pour la cornemuse 435
2.6.3 Bois 436
2.6.3.1 Diagrammes pour bois 436
2.7 Notation des accords 444
2.7.1 Mode accords 445
Généralités sur le mode accords 445
Accords courants 446
Extension et altération d'accords 447
Renversement d'accord et vocification spécifique 450
2.7.2 Gravure des accords 450
Impression des noms d'accord 450
Personnalisation des noms d'accord 453
2.7.3 Basse chiffrée 458
Introduction à la basse chiffrée 459
Saisie de la basse chiffrée 460
Gravure de la basse chiffrée 462
2.8 Musique contemporaine 465
2.8.1 Hauteur et harmonie en musique contemporaine 465
Généralités en matière de hauteur et d'harmonie 465
Notation microtonale 465
Armures contemporaines et harmonie 465
2.8.2 Approches du rythme en musique contemporaine 465

Généralités sur le rythme en musique contemporaine 465
N-olets et musique contemporaine 465
Métriques contemporaines 465
Notation polymétrique étendue 466
Ligatures et musique contemporaine 466
Barres de mesure et musique contemporaine 466
2.8.3 Notation graphique 466
2.8.4 Techniques de partition contemporaine 467
2.8.5 Nouvelles techniques instrumentales 467
2.8.6 Informations complémentaires et exemples pertinents 467
Ouvrages et articles sur la notation en musique contemporaine 468
Partitions et exemples 468
2.9 Notations anciennes 468
2.9.1 Formes de notation ancienne prises en charge 469
2.9.2 Considérations communes aux musiques anciennes 470
Contextes prédéfinis 470
Ligatures 470
Guidons 471
2.9.3 Typographie de musique ancienne 472
Contextes de musique mensurale 472
Clefs anciennes 473
Métriques anciennes 474
Têtes de note anciennes 475
Crochets anciens 475
Silences anciens 476
Altérations et armures anciennes 477
Altérations suggérées (*musica ficta*) 477
Ligatures mensurales 478
2.9.4 Typographie du chant grégorien 479
Contextes du chant grégorien 479
Clefs grégoriennes 480
Altérations et armures grégoriennes 481
Divisions 481
Articulations grégoriennes 482
Points d'augmentation (*morae*) 483
Neumes et ligatures grégoriennes 483
2.9.5 Typographie de notation kiévienne 490
Contextes de notation kiévienne 490
Clefs kiéviennes 491
Notes kiéviennes 491
Altérations kiéviennes 492
Barre de mesure kiévienne 492
Mélismes kiéviens 492
2.9.6 Réédition de musique ancienne 493
Des incipits 494
Mise en forme de la musique mensurale 494
Transcription de chant grégorien 495
Éditions ancienne et moderne à partir d'une même source 498
2.10 Musiques du monde 500
2.10.1 Noms des notes et altérations non-occidentaux 500
Extension des systèmes de notation et d'accordage 500
2.10.2 Musique arabe 501
Références pour la musique arabe 501
Noms des notes en arabe 501

Armures arabes ... 502
Métriques arabes ... 504
Exemple de musique arabe ... 505
Lectures complémentaires pour la musique arabe ... 505
2.10.3 Musique classique turque ... 506
Références pour la musique classique turque ... 506
Noms de note en turc ... 506
Armures turques ... 506
Lectures complémentaires pour la musique turque ... 507

3 Généralités en matière d'entrée et sortie ... 509
3.1 Agencement du code ... 509
3.1.1 Structure d'une partition ... 509
3.1.2 Plusieurs partitions dans un même ouvrage ... 510
3.1.3 Plusieurs éditions pour une même source ... 511
3.1.4 Nom des fichiers de sortie ... 512
3.1.5 Structure de fichier ... 513
3.2 Titres et entêtes ... 515
3.2.1 Création de titres et entête ou pied de page ... 515
Généralités en matière de titrages ... 515
Mise en forme par défaut des titrages subalternes ... 518
Mise en forme par défaut des entête et pied de page ... 522
3.2.2 Titrages personnalisés ... 523
Mise en forme personnalisée des champs de titrage ... 523
Mise en forme personnalisée des titrages ... 523
Mise en forme personnalisée des entête et pied de page ... 526
3.2.3 Création des métadonnées des fichiers de sortie ... 528
3.2.4 Notes de bas de page ... 529
Notes de bas de page dans une expression musicale ... 529
Notes de bas de page dans du texte indépendant ... 534
3.2.5 Référencement des numéros de page ... 537
3.2.6 Table des matières ... 538
3.3 Travail sur des fichiers texte ... 541
3.3.1 Insertion de fichiers LilyPond ... 541
3.3.2 Différentes éditions à partir d'une même source ... 543
Utilisation de variables ... 543
Utilisation de balises ... 544
Globalisation des réglages ... 549
3.3.3 Caractères spéciaux ... 549
Codage du texte ... 549
Unicode ... 550
Équivalents ASCII ... 550
3.4 Contrôle des sorties ... 552
3.4.1 Extraction de fragments musicaux ... 552
3.4.2 Ignorer des passages de la partition ... 552
3.4.3 Formats de sortie alternatifs ... 553
Sortie SVG ... 553
3.4.4 Changement des fontes musicales ... 554
3.5 Génération de fichiers MIDI ... 556
3.5.1 Notation prise en compte dans le MIDI ... 556
3.5.2 Notation non prise en compte dans le MIDI ... 557
3.5.3 Le bloc MIDI ... 557
3.5.4 Gestion des nuances en MIDI ... 558
Indication des nuances en MIDI ... 558

Réglage du volume en MIDI ... 559
Réglage de propriétés dans le bloc MIDI ... 561
3.5.5 Gestion des instruments MIDI ... 562
3.5.6 Gestion des répétitions en MIDI ... 563
3.5.7 Affectation des canaux MIDI ... 564
3.5.8 Propriétés de contextes et effets MIDI ... 566
3.5.9 Amélioration du rendu MIDI ... 567
Le script `articulate` ... 567
Le script `swing` ... 568
3.6 Extraction d'informations musicales ... 569
3.6.1 Affichage de notation au format LilyPond ... 569
3.6.2 Affichage de la musique sous forme d'expression Scheme ... 569
3.6.3 Enregistrement d'événements musicaux dans un fichier ... 569

4 Gestion de l'espace ... 571
4.1 Mise en forme de la page ... 571
4.1.1 Le bloc `\paper` ... 571
4.1.2 Format du papier et adaptation automatique ... 572
Format du papier ... 572
Adaptation automatique au format ... 573
4.1.3 Variables d'espacement vertical fixe ... 574
4.1.4 Variables d'espacement vertical fluctuant ... 575
Structure des variables d'espacement vertical fluctuant ... 575
Liste des variables d'espacement fluctuant ... 576
4.1.5 Variables d'espacement horizontal ... 577
Variables de marge et de largeur ... 577
Variables spécifiques à l'impression recto-verso ... 578
Variables d'indentation et de décalage ... 579
4.1.6 Autres variables du bloc `\paper` ... 580
Variables de gestion des sauts de ligne ... 580
Variables de gestion des sauts de page ... 580
Variables de gestion des numéros de page ... 581
Variables supplémentaires d'entête et *markup* ... 582
Variables de débogage ... 583
4.2 Mise en forme de la partition ... 583
4.2.1 Le bloc `\layout` ... 583
4.2.2 Définition de la taille de portée ... 585
4.3 Sauts ... 588
4.3.1 Sauts de ligne ... 588
4.3.2 Sauts de page ... 592
Saut de page manuel ... 592
Optimisation des sauts de page ... 593
Minimisation des sauts de page ... 593
Présentation en page continue ... 594
Présentation en ligne continue ... 594
Présentation en rouleau ... 594
Optimisation des tournes ... 594
4.4 Espacement vertical ... 595
4.4.1 Espacement vertical au sein d'un système ... 595
Propriétés d'espacement au sein d'un système ... 596
Espacement de portées isolées ... 599
Espacement de portées regroupées ... 600
Espacement des lignes rattachées à des portées ... 601
4.4.2 Positionnement explicite des portées et systèmes ... 603

4.4.3 Résolution des collisions verticales . . . 611
4.5 Espacement horizontal . . . 612
4.5.1 Généralités sur l'espacement horizontal . . . 612
4.5.2 Changement d'espacement en cours de partition . . . 614
4.5.3 Modification de l'espacement horizontal . . . 615
Étirement uniforme des n-olets . . . 616
Espacement strict des notes . . . 616
4.5.4 Largeur de ligne . . . 617
4.5.5 Notation proportionnelle . . . 617
4.6 Réduction du nombre de pages de la partition . . . 624
4.6.1 Mise en évidence de l'espacement . . . 624
4.6.2 Modification de l'espacement . . . 625

5 Modification des réglages prédéfinis . . . 628
5.1 Contextes d'interprétation . . . 628
5.1.1 Tout savoir sur les contextes . . . 628
Définitions de la sortie – hiérarchie des contextes . . . 628
Score – le père de tous les contextes . . . 629
Contextes de haut niveau – les systèmes . . . 629
Contextes de niveau intermédiaire – les portées . . . 629
Contextes de bas niveau – les voix . . . 630
5.1.2 Création et référencement d'un contexte . . . 630
5.1.3 Conservation d'un contexte . . . 634
5.1.4 Modification des greffons de contexte . . . 636
5.1.5 Modification des réglages par défaut d'un contexte . . . 638
Modification de tous les contextes d'un même type . . . 639
Modification d'un contexte particulier . . . 641
Ordre de préséance . . . 643
5.1.6 Définition de nouveaux contextes . . . 644
5.1.7 Ordonnancement des contextes . . . 646
5.2 En quoi consiste la référence des propriétés internes . . . 648
5.2.1 Navigation dans les références du programme . . . 648
5.2.2 Interfaces de rendu . . . 649
5.2.3 Détermination de la propriété d'un objet graphique (grob) . . . 650
5.2.4 Conventions de nommage . . . 651
5.3 Modification de propriétés . . . 651
5.3.1 Vue d'ensemble de la modification des propriétés . . . 652
5.3.2 La commande de fixation `\set` . . . 652
5.3.3 La commande de dérogation `\override` . . . 654
5.3.4 La commande d'affinage `\tweak` . . . 656
5.3.5 `\set` ou `\override` . . . 658
5.3.6 La commande de décalage `\offset` . . . 659
5.3.7 Modification de listes associatives . . . 664
5.4 Propriétés et contextes utiles . . . 666
5.4.1 Modes de saisie . . . 666
5.4.2 Direction et positionnement . . . 667
Indicateurs de position d'une articulation . . . 667
La propriété `direction` . . . 668
5.4.3 Distances et unités de mesure . . . 669
5.4.4 Dimensions . . . 669
5.4.5 Propriétés des symboles de la portée . . . 670
5.4.6 Extenseurs et prolongateurs . . . 670
Utilisation de `spanner-interface` . . . 670
Utilisation de `line-spanner-interface` . . . 673

5.4.7 Visibilité des objets ... 675
Suppression des stencils ... 675
Transparence des objets ... 676
Blanchiment des objets ... 676
Utilisation de break-visibility ... 677
Considérations spécifiques ... 679
5.4.8 Styles de ligne ... 682
5.4.9 Rotation des objets ... 683
Rotation des objets de mise en forme ... 683
Rotation des étiquettes ... 683
5.5 Retouches avancées ... 684
5.5.1 Alignement des objets ... 684
Détermination directe de `X-offset` et `Y-offset` ... 685
Utilisation de `side-position-interface` ... 685
Utilisation de `self-alignment-interface` ... 686
Utilisation de `break-aligned-interface` ... 687
5.5.2 Regroupement vertical d'objets graphiques ... 689
5.5.3 Modification des stencils ... 689
5.5.4 Modification de l'allure des éléments ... 690
Modification des liaisons ... 690
5.5.5 Modification de bandeaux avec rupture ... 694
Utilisation de `\alterBroken` ... 694
5.5.6 Conteneurs requalifiants ... 696
5.6 Utilisation de fonctions musicales ... 698
5.6.1 Syntaxe d'une fonction de substitution ... 698
5.6.2 Exemples de fonction de substitution ... 699

Annexe A Tables du manuel de notation ... 702

A.1 Table des noms d'accord ... 702
A.2 Modificateurs d'accord usuels ... 703
A.3 Accordages prédéfinis ... 706
A.4 Diagrammes d'accord prédéfinis ... 707
Diagrammes pour guitare ... 707
Diagrammes pour ukulele ... 709
Diagrammes pour mandoline ... 711
A.5 Formats de papier prédéfinis ... 713
A.6 Instruments MIDI ... 715
A.7 Liste des couleurs ... 716
A.8 La fonte Emmentaler ... 718
Glyphes de clef ... 719
Glyphes de métrique ... 719
Glyphes de chiffre ... 719
Glyphes d'altération ... 720
Glyphes de tête de note par défaut ... 721
Glyphes de tête de note spéciale ... 721
Glyphes de tête de note à forme variable ... 722
Glyphes de silence ... 725
Glyphes de crochet de croche ... 726
Glyphes de point ... 727
Glyphes de nuance ... 727
Glyphes de script ... 727
Glyphes de flèche ... 730
Glyphes d'extrémité d'accolade ... 730
Glyphes de pédale ... 730

Glyphes d'accordéon ... 731
Glyphes de liaison ... 731
Glyphes de style vaticana ... 731
Glyphes de style medicaea ... 732
Glyphes de style Hufnagel ... 733
Glyphes de style mensural ... 733
Glyphes de style néomensural ... 737
Glyphes de style Petrucci ... 738
Glyphes de style Solesmes ... 739
Glyphes de style kiévien ... 739
A.9 Styles de tête de note ... 740
A.10 Styles de clef ... 740
Clefs standards ... 741
Clefs pour portée de percussions ... 742
Clefs pour tablatures ... 742
Clefs de musique ancienne ... 742
Grégorien ... 742
Mensural ... 743
Kiévien ... 745
A.11 Commandes pour *markup* ... 746
A.11.1 Font ... 746
A.11.2 Align ... 756
A.11.3 Graphic ... 771
A.11.4 Music ... 779
A.11.5 Instrument Specific Markup ... 785
A.11.6 Accordion Registers ... 789
A.11.7 Other ... 794
A.12 Commandes pour liste de *markups* ... 801
A.13 Liste des caractères spéciaux ... 803
A.14 Liste des signes d'articulation ... 805
Scripts d'articulation ... 805
Scripts d'ornement ... 805
Scripts de point d'orgue et point d'arrêt ... 806
Scripts spécifiques à certains instruments ... 806
Scripts de reprise et de répétition ... 806
Scripts pour musique ancienne ... 807
A.15 Notes utilisées en percussion ... 807
A.16 Glossaire technique ... 810
alist (liste associative) ... 810
callback (rappel) ... 810
closure (clôture) ... 810
glyphe ... 810
grob (objet graphique) ... 810
inaltérable ... 811
interface ... 811
lexer (analyseur lexical) ... 811
altérable ... 811
output-def (définition de sortie) ... 812
parser (analyseur syntaxique) ... 812
variable de l'analyseur grammatical ... 812
prob (objet de propriété) ... 812
smob (objet Scheme) ... 813
stencil ... 813
A.17 Liste des propriétés de contexte ... 813

A.18 Propriétés de mise en forme ... 827
A.19 Fonctions musicales prédéfinies ... 848
A.20 Identificateurs de modification de contexte ... 860
A.21 Types de prédicats prédéfinis ... 860
R5RS primary predicates ... 860
R5RS secondary predicates ... 861
Guile predicates ... 861
LilyPond scheme predicates ... 861
LilyPond exported predicates ... 862
A.22 Fonctions Scheme ... 863

Annexe B Aide-mémoire ... 889

Annexe C GNU Free Documentation License ... 893
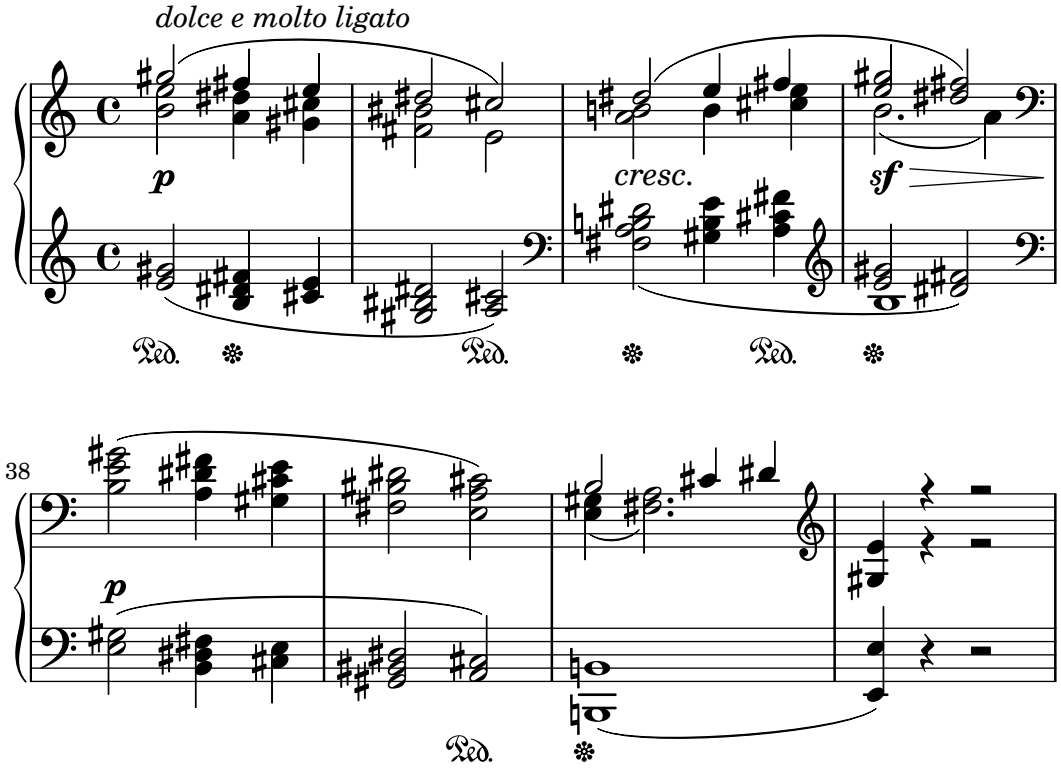
Annexe D Index des commandes LilyPond ... 900

Annexe E Index de LilyPond ... 910

1 Notation musicale générale

Ce chapitre explique comment créer la notation musicale standard.

1.1 Hauteurs



Cette section détaille la façon d'indiquer la hauteur des notes, sous trois aspects : la saisie des hauteurs, la modification des hauteurs et les options de gravure.
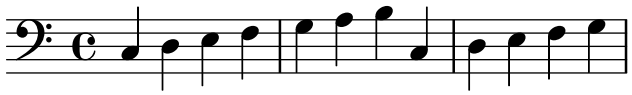
1.1.1 Écriture des hauteurs de note

Cette section explique la manière d'indiquer les hauteurs de note. Deux modes permettent d'indiquer l'octave des notes : le mode absolu, et le mode relatif. Ce dernier est le plus pratique lors de la saisie d'un fichier source au clavier de l'ordinateur.

Hauteurs avec octave absolue
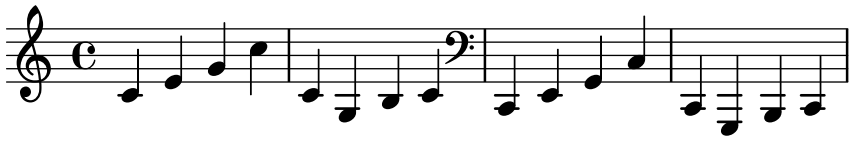
La hauteur s'écrit – à moins de préciser une autre langue – avec la notation batave, en utilisant les lettres de `a` à `g`. Les notes `c` (do) et `b` (si) sont écrites une octave sous le do central.

```
{
  \clef bass
  c4 d e f
  g4 a b c
  d4 e f g
}
```
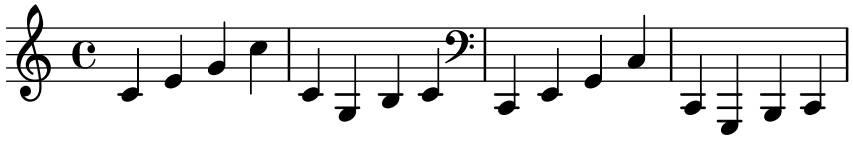


L'octave peut être précisée sous forme d'une série d'apostrophes ' ou d'une série de virgules ,. Chaque ' hausse la note d'une octave ; chaque , baisse la note d'une octave.

```
{
  \clef treble
  c'4 e' g' c''
  c'4 g b c'
  \clef bass
  c,4 e, g, c
  c,4 g,, b,, c,
}
```



Les indications d'octave communes peuvent ne se mentionner qu'une fois, en faisant suivre l'instruction \fixed, placée avant la musique, d'une hauteur de référence. Les hauteurs d'une section \fixed ne nécessitent des ' ou , que lorsqu'elles se trouvent au-dessus ou au-dessous de l'octave de la hauteur de référence.

```
{
  \fixed c' {
    \clef treble
    c4 e g c'
    c4 g, b, c
  }
  \clef bass
  \fixed c, {
    c4 e g c'
    c4 g, b, c
  }
}
```



Les hauteurs d'une expression musicale venant après un \fixed ne seront en rien affectées par un éventuel \relative qui la contiendrait.

Voir aussi

Glossaire musicologique : Section "Noms des notes" dans *Glossaire*.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Octaves relatives

Le mode d'écriture en octave absolue requiert d'indiquer l'octave de chaque note. Pour le mode d'écriture en octave relative, par contre, l'octave d'une note est déterminée par rapport à la note précédente : modifier l'octave d'une note aura des répercussions sur toutes les notes à venir.

Une musique peut être déclarée explicitement comme étant en notation relative à l'aide de la commande \relative :

\relative *hauteur_de_référence expression_musicale*

En mode relatif, chaque note est considérée comme étant le plus proche possible de celle qui la précède. L'octave des notes mentionnées dans *expression_musicale* va être calculée de la manière suivante :

- Si aucun signe de changement d'octave n'est utilisé, l'intervalle de base entre la note actuelle et la précédente sera toujours au plus d'une quarte. Cet intervalle est déterminé sans tenir compte des altérations.
- Un signe de changement d'octave `'` ou `,` peut être ajouté pour hausser ou baisser la note d'une octave par rapport à la hauteur calculée sans spécification.
- Ces signes de changement d'octave peuvent être multipliés. Par exemple, `''` ou `,,` ajouteront une octave supplémentaire.
- La première hauteur de *`expression_musicale`* est déterminée relativement à *`hauteur_de_référence`*. Cette *`hauteur_de_référence`* s'exprime en octave absolue ; plusieurs options s'offrent à vous :

  une octave de do (`c`)
  : Un `c'` identifiant le do placé entre les portées d'un piano, il est de fait aisé de déterminer d'autres octaves de `c`. Pour une musique qui commencerait par un sol dièse (`gis`) au-dessus du do suraigu (`c'''`), vous écririez quelque chose comme `\relative c''' { gis' ... }`
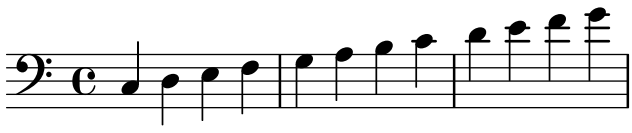
  une octave de la première note de l'expression
  : Écrire `\relative gis'' { gis ... }` permet de déterminer facilement la hauteur absolue de la première note de l'expression.

  pas de hauteur de référence explicite
  : La fromulation `\relative { gis''' ... }`) peut se voir comme une version abrégée de l'option précédente : la première note de l'expression est écrite en octave absolue. Cette option est équivalente à prendre un `f` comme hauteur de référence.

  La documentation de LilyPond utilise en règle générale la dernière option.
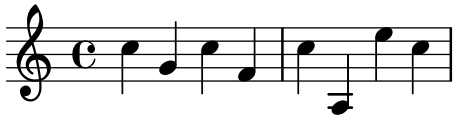
Voici le mode `\relative` en action.

```
\relative {
  \clef bass
  c d e f
  g a b c
  d e f g
}
```
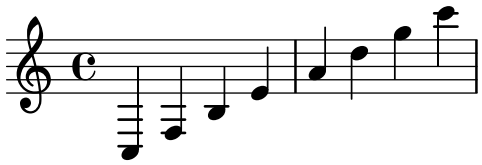


On utilise les signes de changement d'octave pour les intervalles dépassant la quarte.

```
\relative {
  c'' g c f,
  c' a, e'' c
}
```
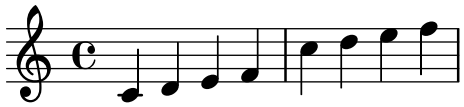


Bien que ne comportant aucun signe de changement d'octave, une séquence de notes peut tout à fait couvrir un intervalle important.

```
\relative {
  c f b e
  a d g c
}
```



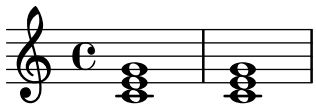
Lorsque plusieurs blocs `\relative` sont imbriqués, le bloc `\relative` inclus dispose de sa propre haueur de référence indépendamment de celui qui l'englobe.

```
\relative {
  c' d e f
  \relative {
    c'' d e f
  }
}
```



`\relative` est sans effet sur un bloc `\chordmode`.
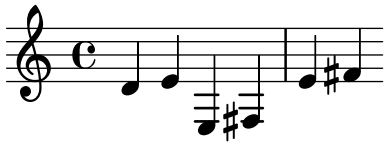
```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` n'est pas permis au sein d'un bloc `\chordmode`.
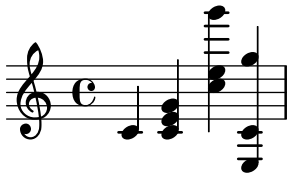
Pour utiliser le mode d'octave relative dans de la musique transposée, une clause `\relative` additionnelle doit être placée au sein du bloc `\transpose`.

```
\relative {
  d' e
  \transpose f g {
    d e
    \relative {
      d' e
    }
  }
}
```
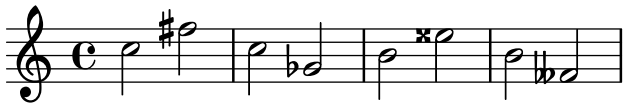


Si l'expression précédente est un accord, c'est la première note de l'accord qui détermine l'emplacement de la première note du prochain accord. À l'intérieur de l'accord, les notes sont placées relativement à celle qui précède. Examinez avec attention l'exemple suivant, et tout particulièrement le positionnement des do.

```
\relative {
  c'
  <c e g>
  <c' e g'>
  <c, e, g''>
}
```



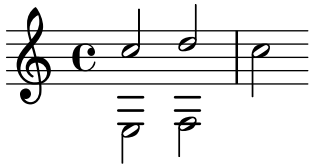
Comme nous l'avons vu, l'octaviation est déterminée sans tenir compte des altérations. Ainsi un mi double-dièse qui suit un si naturel sera placé au-dessus de celui-ci, alors qu'un fa double-bémol se retrouvera en dessous. En d'autres termes, une quarte doublement augmentée demeure considérée comme un intervalle plus petit qu'une quinte diminuée, bien que la quarte doublement augmentée soit de sept demi-tons et la quinte diminuée de seulement six demi-tons.

```
\relative {
  c''2 fis
  c2 ges
  b2 eisis
  b2 feses
}
```



Dans certaines situation complexes, il peut être souhaitable de revenir à une hauteur déterminée sans tenir compte de ce qui se passait auparavant, à l'aide d'un `\resetRelativeOctave` :

```
\relative {
  <<
    { c''2 d }
    \\
    { e,,2 f }
  >>
  \resetRelativeOctave c''
  c2
}
```



Voir aussi

Glossaire musicologique : Section “quinte” dans *Glossaire*, Section “intervalle” dans *Glossaire*, Section “nom des notes” dans *Glossaire*.

Manuel de notation : [Vérifications d’octave], page 10.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.
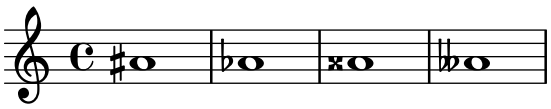
Référence des propriétés internes : Section “RelativeOctaveMusic” dans *Référence des propriétés internes*.

Altérations

> **Note :** Les nouveaux utilisateurs sont parfois déroutés par la gestion des altérations et de l’armure. Pour LilyPond, un nom de note spécifie une hauteur ; l’armure et la clef ne feront que déterminer comment ces hauteurs seront retranscrites. Un simple `c` signifie tout bonnement « do naturel » quelles que soient l’armure et la clef en vigueur. Pour plus d’information, reportez-vous au chapitre Section “Hauteurs et armure” dans *Manuel d’initiation*.
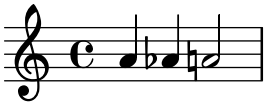
Dans la notation par défaut, un *dièse* est formé en ajoutant `is` après le nom de note, un *bémol* en ajoutant `es`. Les *double-dièses* et *double-bémols* sont obtenus en ajoutant respectivement `isis` ou `eses` au nom de note. Ce sont les noms de note hollandais. Pour les autres langues, consultez [Nom des notes dans d’autres langues], page 8.

```
\relative c'' { ais1 aes aisis aeses }
```



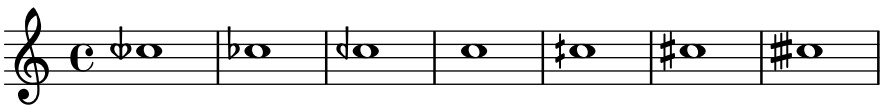
Une hauteur naturelle se saisit comme un simple nom de note, sans suffixe. Un bécarre sera imprimé si besoin est, que ce soit pour annuler les effets d’un précédente altération accidentelle ou pour déroger à l’armure.

```
\relative c'' { a4 aes a2 }
```



Les demi-bémols et demi-dièses s’écrivent en ajoutant respectivement `eh` et `ih`. Voici une série de dos altérés en hauteurs croissantes :

```
\relative c'' { ceseh1 ces ceh c cih cis cisih }
```
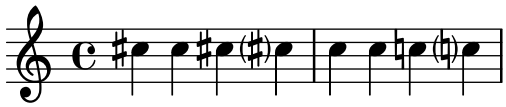


Les micro-intervalles sont aussi exportés dans le fichier MIDI.

Normalement, les altérations sont imprimées automatiquement, mais il se peut que vous vouliez les imprimer manuellement. On peut forcer l’impression d’une altération, dite « de précaution », en ajoutant un point d’exclamation `!` après la hauteur de note. Une altération
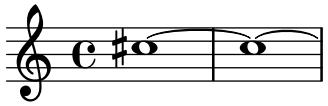
entre parenthèses peut être obtenue en ajoutant un point d'interrogation `?` après la hauteur de note.
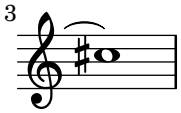
```
\relative c'' { cis cis cis! cis? c c c! c? }
```



Lorsqu'une note est prolongée par une liaison de tenue, l'altération ne sera réimprimée que s'il y a un saut de ligne.

```
\relative c'' {
  cis1 ~ 1 ~
  \break
  cis
}
```
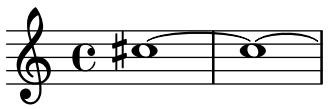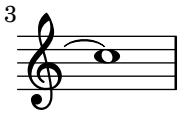




Morceaux choisis

Non répétition de l'altération après saut de ligne sur liaison de prolongation

Cet exemple illustre comment, lorsqu'une note affublée d'une altération accidentelle est prolongée, ne pas répéter cette altération après un saut de ligne.

```
\relative c'' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```
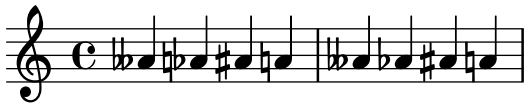




Suppression des bécarres superflus

En accord avec les règles traditionnelles de l'écriture musicale, on grave un bécarre avant un dièse ou un bémol si la note était auparavant affublée d'un double-dièse ou double-bémol. Pour adopter un comportement plus contemporain, la propriété `extraNatural` du contexte `Staff` doit se voir attribuer la valeur `##f` (faux).

```
\relative c'' {
  aeses4 aes ais a
```

```
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



Voir aussi

Glossaire musicologique : Section “dièse” dans *Glossaire*, Section “bémol” dans *Glossaire*, Section “double dièse” dans *Glossaire*, Section “double bémol” dans *Glossaire*, Section “Nom des notes” dans *Glossaire*, Section “quart de ton” dans *Glossaire*.

Manuel d’initiation : Section “Hauteurs et armure” dans *Manuel d’initiation*.

Manuel de notation : [Altérations accidentelles automatiques], page 29, [Altérations suggérées (*musica ficta*)], page 477, [Nom des notes dans d’autres langues], page 8.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Références des propriétés internes : Section “Accidental_engraver” dans *Référence des propriétés internes*, Section “Accidental” dans *Référence des propriétés internes*, Section “AccidentalCautionary” dans *Référence des propriétés internes*, Section “accidental-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements
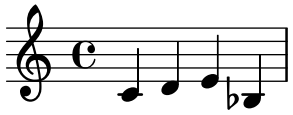
Il n’y a pas de standard universellement accepté pour noter le bémol et demi (qui abaisse la hauteur trois quarts de ton), le symbole de LilyPond n’est donc conforme à aucun standard.

Nom des notes dans d’autres langues

Vous disposez de jeux prédéfinis de noms de note et altérations pour plusieurs autres langues. Pour les utiliser, il suffit de déclarer, en début de fichier, la langue que vous utilisez. Voici comment, par exemple, utiliser l’italien pour votre saisie :

```
\language "italiano"

\relative {
  do' re mi sib
}
```



Les langues disponibles ainsi que les noms de note utilisés sont les suivantes :

Lange	**Nom des notes**
nederlands	c d e f g a bes b
català ou catalan	do re mi fa sol la sib si
deutsch	c d e f g a b h
english	c d e f g a bf/b-flat b
español ou espanol	do re mi fa sol la sib si
français	do ré/re mi fa sol la sib si

italiano	do re mi fa sol la sib si
norsk	c d e f g a b h
português or portugues	do re mi fa sol la sib si
suomi	c d e f g a b h
svenska	c d e f g a b h
vlaams	do re mi fa sol la sib si
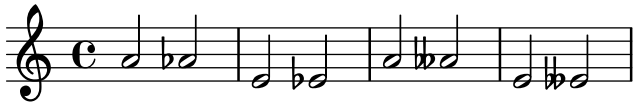
et les suffixes d'altération correspondants :

Langue	dièse	bémol	double dièse	double bémol
nederlands	is	es	isis	eses
català ou catalan	d/s	b	dd/ss	bb
deutsch	is	es	isis	eses
english	s/-sharp	f/-flat	ss/x/-sharpsharp	ff/-flatflat
español ou espanol	s	b	ss/x	bb
français	d	b	dd/x	bb
italiano	d	b	dd	bb
norsk	iss/is	ess/es	ississ/isis	essess/eses
português or portugues	s	b	ss	bb
suomi	is	es	isis	eses
svenska	iss	ess	ississ	essess
vlaams	k	b	kk	bb

Notez qu'en hollandais, en allemand, en norvégien et en finnois, un *la* altéré de *bémol* se note `aes` et se contracte en `as` ; pour le hollandais et le norvégien, LilyPond accepte cependant les deux formes. Il en va de même pour `es` et `ees`, `aeses` et `ases`, ainsi que pour `eeses` et `eses`.

En allemand et en finnois, LilyPond fournit la forme plus couramment utilisée de `asas` pour `ases`.

```
\relative c'' { a2 as e es a ases e eses }
```



Certaines musiques utilisent des microtonalités, pour lesquelles les altérations sont des fractions de dièse ou bémol « normaux ». Le tableau suivant répertorie les noms de note en quart de ton, tels que définis dans les fichiers linguistiques. Les préfixes *semi-* et *sesqui-* correspondent au *demi-* et *trois demis*.
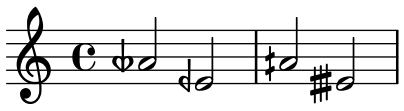
Langue	semi-dièse	semi-bémol	sesqui-dièse	sesqui-bémol
nederlands	ih	eh	isih	eseh
català or catalan	qd/qs	qb	tqd/tqs	tqb
deutsch	ih	eh	isih	eseh
english	qs	qf	tqs	tqf
español or espanol	cs	cb	tcs	tcb
français	sd	sb	dsd	bsb

italiano	sd	sb	dsd	bsb
norsk	ih	eh	issih/isih	esseh/eseh
português or portugues	sqt	bqt	stqt	btqt
suomi	ih	eh	isih	eseh
svenska	ih	eh	issih	esseh
vlaams	hk	hb	khk	bhb

En allemand, les contractions de microtonalités sont identiques à celles des hauteurs normales indiquées ci-dessus.

```
\language "deutsch"

\relative c'' { asah2 eh aih eisih }
```



La plupart des langues dont nous venons de parler correspondent à la musique classique occidentale au tempérament égal – le concept de *Common Practice Period* en anglais. LilyPond prend néanmoins en charge d'autres systèmes de notation, comme indiqué au chapitre Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Voir aussi

Glossaire musicologique : Section “Nom des notes” dans *Glossaire*, Section “Common Practice Period” dans *Glossaire*.

Manuel de notation : Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Fichiers d'initialisation : `scm/define-note-names.scm`.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

1.1.2 Modification de plusieurs hauteurs
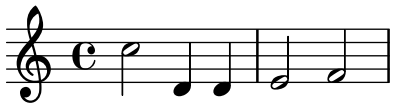
Cette partie traite de la manière de modifier les hauteurs de note.

Vérifications d'octave

Les tests d'octave rendent la correction d'erreurs d'octave plus facile dans le mode d'octave `relative` – un `,` ou un `'` oublié, ça n'arrive pas qu'aux autres !
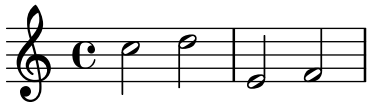
Une note peut être suivie de `=`*apostrophes/virgules* pour indiquer à quelle **octave absolue** elle devrait être. Dans l'exemple suivant, le premier `d` générera un avertissement, puisqu'on attend un `d''` – intervalle inférieur à la quarte – mais qu'on obtient un `d'`. Sur la partition, l'octave sera corrigée pour donner un `d'` et la prochaine note sera calculée en fonction de ce `d'` et non de `d''`.

```
\relative {
  c''2 d='4 d
  e2 f
}
```



Il existe aussi une vérification d'octave qui ne produit pas de musique imprimée, ayant pour syntaxe `\octaveCheck` *`hauteur_référence`* – *`hauteur_référence`* étant spécifiée en mode absolu. Cette commande vérifie que l'intervalle entre la note qui précède et *`hauteur_référence`* est inférieur à la quinte comme il se doit en mode relatif. Dans le cas contraire, un message sera émis. Bien que la note précédente ne sera pas modifiée, les notes suivantes seront positionnées relativement à la valeur corrigée.
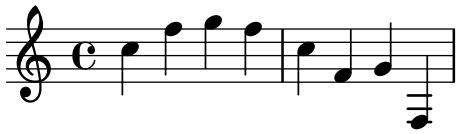
```
\relative {
  c''2 d
  \octaveCheck c'
  e2 f
}
```



Dans les deux mesures qui suivent, les premier et troisième `\octaveCheck` échouent, mais le deuxième est concluant.

```
\relative {
  c''4 f g f

  c4
  \octaveCheck c'
  f
  \octaveCheck c'
  g
  \octaveCheck c'
  f
}
```



Voir aussi

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "RelativeOctaveCheck" dans *Référence des propriétés internes*.

Transposition

Une expression musicale peut être transposée avec `\transpose`. En voici la syntaxe :
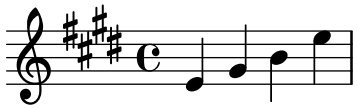
`\transpose` *`note_de_départ note_d_arrivée expression_musicale`*

Cela signifie que *`expression_musicale`* est transposé de l'intervalle séparant *`note_de_départ`* et *`note_d_arrivée`* : toute note dont la hauteur était `note_de_départ` est changée en `note_d_arrivée` ; les autres notes seront changées selon le même intervalle. Les deux hauteurs s'expriment en octave absolue.

> **Note :** La musique contenue dans un bloc `\transpose` est en octaves absolues, sauf à inclure dans ce même bloc une clause `\relative`.
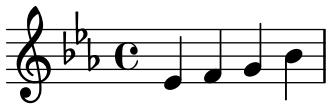
Prenons comme exemple une pièce écrite en ré majeur. Si cette pièce est un peu trop basse pour l'interprète, elle peut être transposée en mi majeur. Vous noterez que l'armure est automatiquement modifiée.

```
\transpose d e {
  \relative {
    \key d \major
    d'4 fis a d
  }
}
```



Regardons maintenant une partie écrite pour violon – un instrument en ut. Si cette partie doit être jouée par une clarinette en la (écrite à la tierce mineure supérieure, un do écrit donnant un la réel), la transposition suivante créera la partie appropriée.
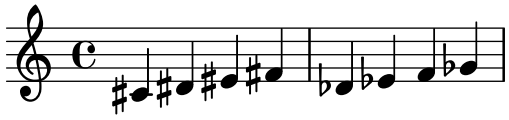
```
\transpose a c' {
  \relative {
    \key c \major
    c'4 d e g
  }
}
```



La présence de `\key c \major` s'explique par le fait que, bien que les notes soient effectivement transposées, l'armure ne sera imprimée que dans la mesure où elle est explicitement mentionnée.

`\transpose` fait la distinction entre les notes enharmoniques : `\transpose c cis` et `\transpose c des` transposeront la pièce un demi-ton plus haut, au détail près que la première version écrira des dièses et la deuxième des bémols.

```
music = \relative { c' d e f }
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



On peut aussi utiliser `\transpose` pour entrer des notes écrites pour un instrument transpositeur. Normalement, les hauteurs dans LilyPond sont écrites en ut, c'est à dire en sons réels, mais elles peuvent être écrites dans un autre ton. Prenons l'exemple d'un morceau pour trompette en si bémol commençant sur un ré à l'oreille ; on pourrait écrire

```
musiqueEnSiBemol = { e4 ... }
\transpose c bes, \musiqueEnSiBemol
```

Pour imprimer cette musique en fa – et de ce fait produire une partie de cor au lieu d'un conducteur en notes réelles – on utilisera un deuxième `\transpose` :

```
musiqueEnSiBemol = { e4 ... }
\transpose f c' { \transpose c bes, \musiqueEnSiBemol }
```

Pour plus d'information à ce sujet, consultez [Instruments transpositeurs], page 28.

Morceaux choisis

Transposition et réduction du nombre d'altérations accidentelles

Cet exemple, grâce à un peu de code Scheme, donne la priorité aux enharmoniques afin de limiter le nombre d'altérations supplémentaires. La règle applicable est :

- Les altérations doubles sont supprimées
- Si dièse -> Do
- Mi dièse -> Fa
- Do bémol -> Si
- Fa bémol -> Mi

Cette façon de procéder aboutit à plus d'enharmoniques naturelles.

```
#(define (naturalize-pitch p)
   (let ((o (ly:pitch-octave p))
         (a (* 4 (ly:pitch-alteration p)))
         ;; alteration, a, in quarter tone steps,
         ;; for historical reasons
         (n (ly:pitch-notename p)))
     (cond
      ((and (> a 1) (or (eqv? n 6) (eqv? n 2)))
       (set! a (- a 2))
       (set! n (+ n 1)))
      ((and (< a -1) (or (eqv? n 0) (eqv? n 3)))
       (set! a (+ a 2))
       (set! n (- n 1))))
     (cond
      ((> a 2) (set! a (- a 4)) (set! n (+ n 1)))
      ((< a -2) (set! a (+ a 4)) (set! n (- n 1))))
     (if (< n 0) (begin (set! o (- o 1)) (set! n (+ n 7))))
     (if (> n 6) (begin (set! o (+ o 1)) (set! n (- n 7))))
     (ly:make-pitch o n (/ a 4))))
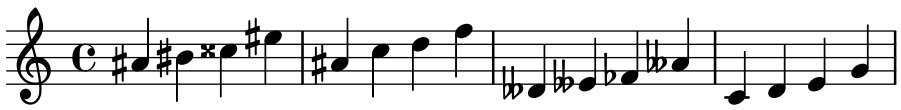
#(define (naturalize music)
   (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
     (if (pair? es)
         (ly:music-set-property!
          music 'elements
          (map naturalize es)))
     (if (ly:music? e)
         (ly:music-set-property!
          music 'element
          (naturalize e)))
     (if (ly:pitch? p)
```

```
          (begin
            (set! p (naturalize-pitch p))
            (ly:music-set-property! music 'pitch p)))
      music))

naturalizeMusic =
#(define-music-function (m)
   (ly:music?)
   (naturalize m))

music = \relative c' { c4 d e g }

\score {
  \new Staff {
    \transpose c ais { \music }
    \naturalizeMusic \transpose c ais { \music }
    \transpose c deses { \music }
    \naturalizeMusic \transpose c deses { \music }
  }
  \layout { }
}
```



Voir aussi

Manuel de notation : [Instruments transpositeurs], page 28, [Inversion], page 14, [Octaves relatives], page 2, [Rétrogradation], page 15, [Transformations modales], page 15.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriété internes : Section “TransposedMusic” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Si vous voulez utiliser en même temps `\transpose` et `\relative`, vous devez mettre `\transpose` en dehors de `\relative`, puisque `\relative` n’aura aucun effet sur la musique apparaissant dans un `\transpose`.

La fonction `\transpose` ne permet pas d’imprimer des altérations triples ; elle les remplacera par un « équivalent enharmonique » – par exemple ré bémol au lieu de mi triple bémol.

Inversion
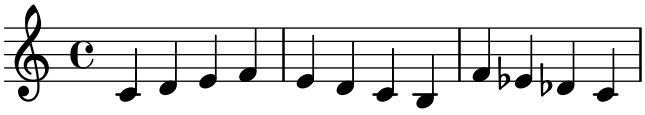
Une expression musicale peut s’inverser et être transposée à l’aide de l’instruction

`\inversion` *`hauteur-pivot hauteur-arrivée expression_musicale`*

L’*`expression_musicale`* sera alors inversée, intervalle par intervalle, puis transposée de telle sorte que *`hauteur-pivot`* devienne *`hauteur-arrivée`*.

```
music = \relative { c' d e f }
\new Staff {
  \music
  \inversion d' d' \music
  \inversion d' ees' \music
```

```
}
```



Note : Le motif à inverser doit être exprimé en hauteur absolue, à moins d'avoir été préalablement inclus explicitement dans un bloc `\relative`.

Voir aussi
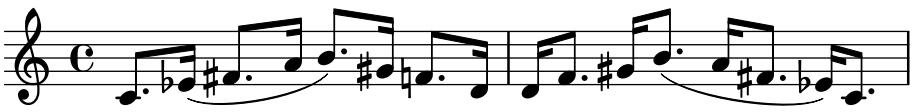
Manuel de notation : [Rétrogradation], page 15, [Transformations modales], page 15, [Transposition], page 11.

Rétrogradation

Une expression musicale peut se renverser et se présenter sous forme rétrograde :

```
music = \relative { c'8. ees16( fis8. a16 b8.) gis16 f8. d16 }

\new Staff {
  \music
  \retrograde \music
}
```



Problèmes connus et avertissements

La fonction `\retrograde` est un outil plutôt simpliste. Dans la mesure où de nombreux événements se reflètent au lieu d'être échangés, les ajustements et indicateurs de positionnement à l'entame d'un objet étendu devront être répétés à leur terminaison : `^(` devra se terminer par `^)`, tout `\<` ou `\cresc` devra se terminer par un `\!` ou un `\endcresc` et tout `\>` ou `\decr` devra se terminer par un `\enddecr`. Les dérogations ou commandes modifiant les propriétés sur la durée peuvent avoir des effets surprenants.

Voir aussi

Manuel de notation : [Inversion], page 14, [Transformations modales], page 15, [Transposition], page 11.

Transformations modales

Dans une composition basée sur une gamme, un même motif est transformé à plusieurs reprises et selon des schémas différents. Il peut être *transposé* pour partir de différents points de la gamme ou bien être *inversé* à partir d'une note pivot dans la gamme. Il peut aussi être renversé pour produire une rétrogradation.

Note : Toute note qui ne ferait pas partie de la gamme en question ne sera pas transformée.

Transposition modale
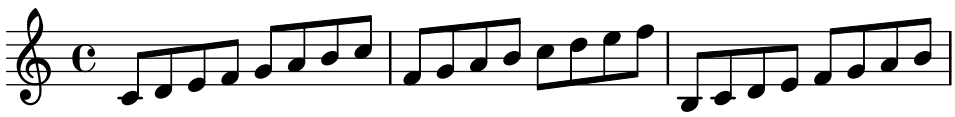
Un motif peut se transposer selon une gamme donnée :

`\modalTranspose` *`hauteur-départ hauteur-arrivée gamme motif`*

Les notes de *motif* seront décalées à l'intérieur de la *gamme* selon leur degré, déterminé par l'intervalle entre *hauteur-départ* et *hauteur-arrivée* :

```
diatonicScale = \relative { c' d e f g a b }
motif = \relative { c'8 d e f g a b c }
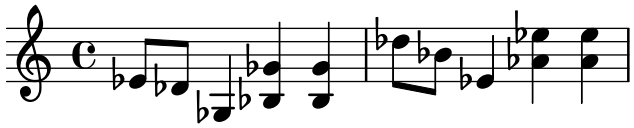
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```



Il est ainsi possible de déterminer une gamme ascendante, quels qu'en soient l'amplitude et les différents intervalles :

```
pentatonicScale = \relative { ges aes bes des ees }
motif = \relative { ees'8 des ges,4 <ges' bes,> <ges bes,> }
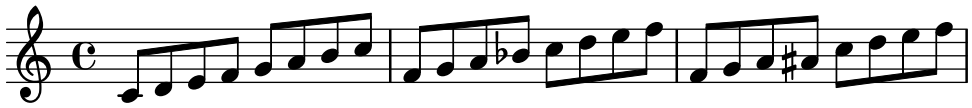
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```



L'utilisation de `\modalTranspose` avec une gamme chromatique produit les mêmes effets qu'un `\transpose`, à ceci près que les notes seront alors prédéterminées :

```
chromaticScale = \relative { c' cis d dis e f fis g gis a ais b }
motif = \relative { c'8 d e f g a b c }

\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
```



Inversion modale

Un motif peut s'inverser selon une certaine gamme et à partir d'un pivot déterminé, puis transposé, le tout en une seule opération :
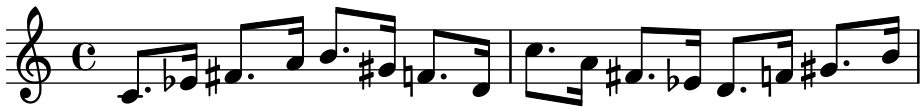
`\modalInversion` *`hauteur-pivot hauteur-arrivée gamme motif`*

Les notes de *motif* se retrouvent au même degré par rapport à *hauteur-pivot* dans la *gamme*, toutefois dans le sens opposé, puis décalées dans cette même *gamme* de l'intervalle séparant *hauteur-départ* et *hauteur-arrivée*.

Il est donc possible de simplement inverser à partir d'une des notes de la gamme en donnant la même valeur à *hauteur-départ* et *hauteur-arrivée* :

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }
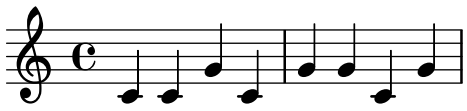
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Pour permuter deux notes de la gamme, il suffit donc d'inverser à partir de l'une des notes et de transposer d'un degré de la gamme. Les deux notes spécifiées peuvent s'interpréter comme étant les bornes du pivot.

```
scale = \relative { c' g' }
motive = \relative { c' c g' c, }
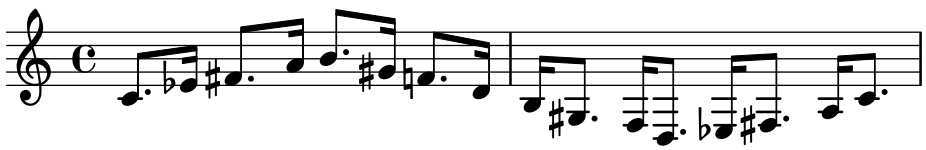
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



L'opération conjointe d'une inversion et d'une rétrogradation produit une rétrogradation inversée :

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
}
```



Voir aussi

Manuel de notation : [Inversion], page 14, [Rétrogradation], page 15, [Transposition], page 11.
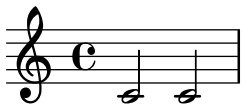
1.1.3 Gravure des hauteurs

Nous allons voir dans cette partie comment influer sur la gravure des hauteurs.

Clefs
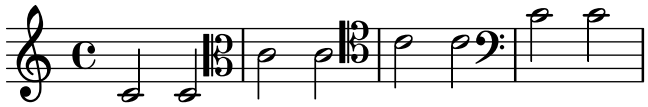
La clef indique quelles lignes de la portée correspondent à quelles hauteurs. En l'absence de commande explicite, LilyPond utilise par défaut la clef de sol.

```
c'2 c'
```



La clef se modifie à l'aide de la commande `\clef` suivie d'un nom approprié. Pour chacun des exemples suivants est indiquée la position du do médium.

```
\clef treble
c'2 c'
\clef alto
c'2 c'
\clef tenor
c'2 c'
\clef bass
c'2 c'
```



Les différents noms possibles sont répertoriés à l'annexe Section A.10 [Styles de clef], page 740.
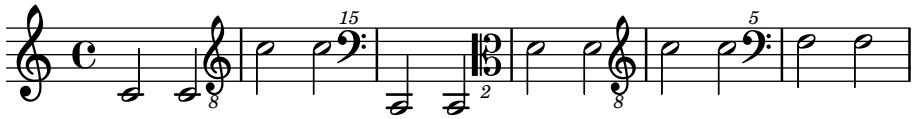
Des clefs spéciales, telles que celles rencontrées en musique ancienne, sont abordées dans [Clefs anciennes], page 473, et [Clefs grégoriennes], page 480. La musique requérant des clefs de tablature est traitée dans [Tablatures par défaut], page 371, et [Tablatures personnalisées], page 389.

Les citations peuvent demander une modification de clef à l'aide des commandes `\cueClef` et `\cueDuringWithClef` – voir [Mise en forme d'une citation], page 224.

En ajoutant `_8` ou `^8` au nom de la clef, celle-ci est transposée à l'octave respectivement inférieure ou supérieure, et `_15` ou `^15` la transpose de deux octaves. D'autres nombres entiers peuvent être utilisés selon les besoins. L'argument *clefname* doit être mis entre guillemets lorsqu'il contient des caractères supplémentaires. Par exemple,
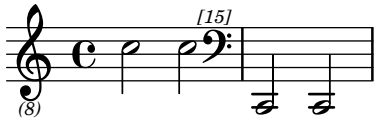
```
\clef treble
c'2 c'
\clef "treble_8"
c'2 c'
\clef "bass^15"
c'2 c'
\clef "alto_2"
c'2 c'
\clef "G_8"
```

```
c'2 c'
\clef "F^5"
c'2 c'
```



Une indication d'octaviation optionnelle s'obtient en entourant l'argument numérique par des parenthèses ou des crochets :

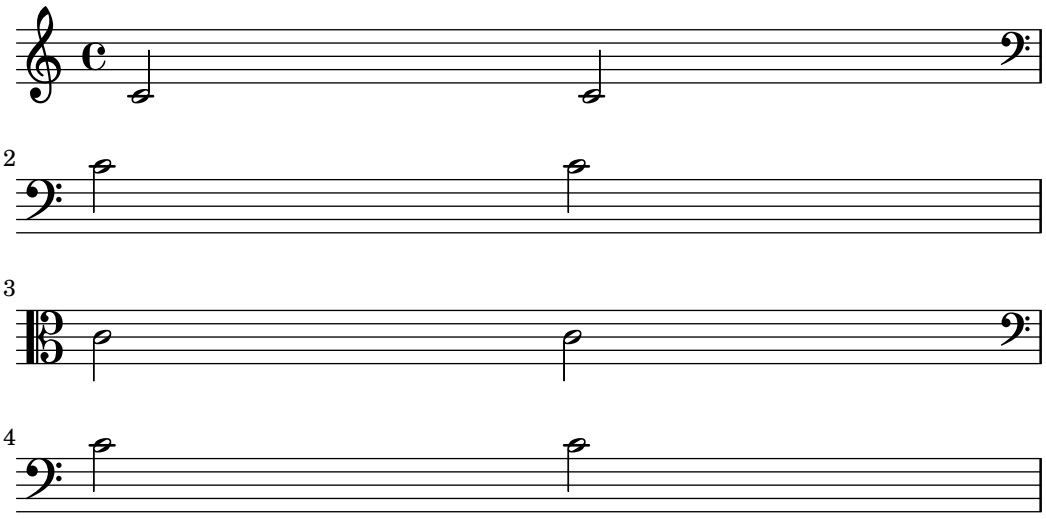
```
\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'
```



Les hauteurs seront affichées comme si l'argument numérique n'avait pas été encadré de parenthèses ou crochets.

Lorsqu'un changement de clef intervient en même temps qu'un saut de ligne, la nouvelle clef est imprimée à la fois en fin de ligne et au début de la suivante. Vous pouvez toujours supprimer cette « clef de précaution ».
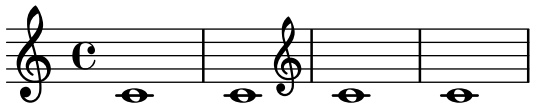
```
\clef treble { c'2 c' } \break
\clef bass { c'2 c' } \break
\clef alto
  \set Staff.explicitClefVisibility = #end-of-line-invisible
  { c'2 c' } \break
  \unset Staff.explicitClefVisibility
\clef bass { c'2 c' } \break
```



Lorsqu'une clef a déjà été imprimée et qu'aucune autre clef n'a depuis été imprimée, LilyPond ignorera toute réitération de la commande `\clef`. Forcer la réimpression de la clef s'obtient à l'aide de la commande `\set Staff.forceClef = ##t`.

```
\clef treble
```
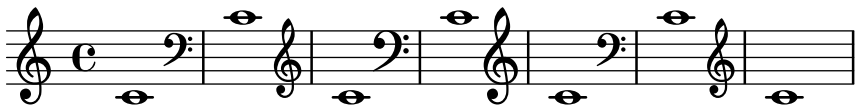
```
c'1
\clef treble
c'1
\set Staff.forceClef = ##t
c'1
\clef treble
c'1
```



Pour être plus précis, la commande `\clef` n'a pas pour fonction d'imprimer une clef ; elle détermine ou modifie une propriété attachée au graveur de clefs (le `Clef_engraver`), qui décide de son propre chef quand doit être affichée une clef dans la portée en cours. La propriété `forceClef` ne vient que forcer la décision de réimprimer une fois la clef en un point donné.

Le symbole imprimé lors d'un changement de clef est plus petit que la clef initiale. La taille peut toutefois être ajustée.

```
\clef "treble"
c'1
\clef "bass"
c'1
\clef "treble"
c'1
\override Staff.Clef.full-size-change = ##t
\clef "bass"
c'1
\clef "treble"
c'1
\revert Staff.Clef.full-size-change
\clef "bass"
c'1
\clef "treble"
c'1
```



Morceaux choisis

Affinage des propriétés d'une clef

Modifier le glyphe, la position de la clef ou son octaviation ne changeront pas la position des notes ; il faut pour y parvenir modifier aussi la position du do médium. La redéfinition préalable de `middleCClefPosition` permet de placer l'armure sur les bonnes lignes. Le positionnement est relatif à la ligne médiane, un nombre positif faisant monter, un nombre négatif abaissant.

Par exemple, la commande `\clef "treble_8"` équivaut à définir `clefGlyph`, `clefPosition` – qui contrôle la position verticale de la clef – `middleCPosition` et `clefOctavation`. Une nouvelle clef apparaîtra dès lors que l'une de ces propriétés, à l'exception de `middleCPosition`, aura été modifiée.

Les exemples qui suivent illustrent les différentes possibilités de définir ces propriétés manuellement. Sur la première ligne, la position relative des notes par rapport aux clefs est préservée, ce qui n'est pas le cas pour la deuxième ligne.

```
{
  % The default treble clef
  \key f \major
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  \set Staff.middleCPosition = #6
  \set Staff.middleCClefPosition = #6
  \key g \major
  c'1
  % The baritone clef
  \set Staff.clefGlyph = #"clefs.C"
  \set Staff.clefPosition = #4
  \set Staff.middleCPosition = #4
  \set Staff.middleCClefPosition = #4
  \key f \major
  c'1
  % The standard choral tenor clef
  \set Staff.clefGlyph = #"clefs.G"
  \set Staff.clefPosition = #-2
  \set Staff.clefTransposition = #-7
  \set Staff.middleCPosition = #1
  \set Staff.middleCClefPosition = #1
  \key f \major
  c'1
  % A non-standard clef
  \set Staff.clefPosition = #0
  \set Staff.clefTransposition = #0
  \set Staff.middleCPosition = #-4
  \set Staff.middleCClefPosition = #-4
  \key g \major
  c'1 \break

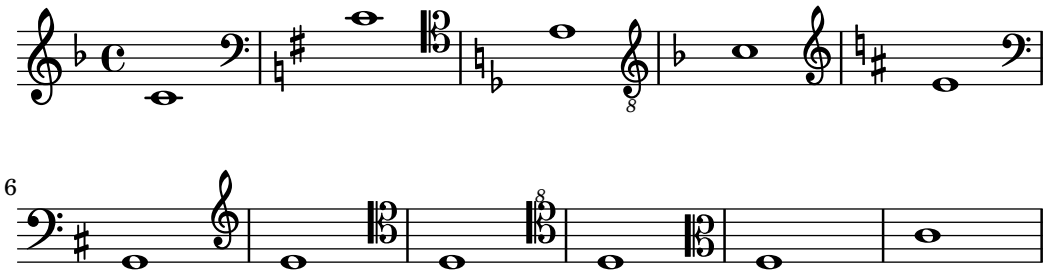
  % The following clef changes do not preserve
  % the normal relationship between notes, key signatures
  % and clefs:

  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  c'1
  \set Staff.clefGlyph = #"clefs.G"
  c'1
  \set Staff.clefGlyph = #"clefs.C"
  c'1
  \set Staff.clefTransposition = #7
  c'1
  \set Staff.clefTransposition = #0
```

```
  \set Staff.clefPosition = #0
  c'1

  % Return to the normal clef:

  \set Staff.middleCPosition = #0
  c'1
}
```



Voir aussi

Manuel de notation : [Clefs anciennes], page 473, [Clefs grégoriennes], page 480, [Mise en forme d'une citation], page 224, Section 2.9 [Notations anciennes], page 468, [Tablatures par défaut], page 371, [Tablatures personnalisées], page 389.

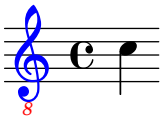
Fichiers d'initialisation : `scm/parser-clef.scm`.

Morceaux choisis: Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Clef_engraver" dans *Référence des propriétés internes*, Section "Clef" dans *Référence des propriétés internes*, Section "ClefModifier" dans *Référence des propriétés internes*, Section "clef-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L'indicateur d'octaviation attaché à la clef est un objet graphique en lui même. Par voie de conséquence, tout `\override` affectant l'objet `Clef` devra être manuellement répercuté sur l'objet `ClefModifier`.



Armure

> **Note :** Les nouveaux utilisateurs sont parfois déroutés par la gestion des altérations et de l'armure. Pour LilyPond, une hauteur n'est que du matériau brut ; l'armure et la clef ne feront que déterminer comment ce matériau sera retranscrit. Un simple `c` signifie tout bonnement « do naturel » quelles que soient l'armure et la clef en question. Pour plus d'information, reportez-vous au chapitre Section "Hauteurs et armure" dans *Manuel d'initiation*.
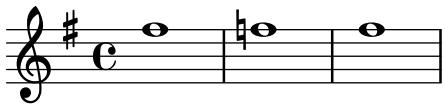
L'armure indique la tonalité dans laquelle la pièce doit être jouée. Elle comprend un ensemble d'altérations (dièses ou bémols) à la clef, c'est-à-dire au début de la portée. Elle peut varier en cours de morceau.

On définit ou modifie l'armure avec la commande `\key` :

`\key` *`hauteur mode`*

Ici, *`mode`* doit être `\major` ou `\minor` afin d'avoir respectivement *hauteur*-majeur ou *hauteur*-mineur. Vous pouvez aussi avoir recours aux modes anciens que sont `\ionian`, `\locrian`, `\aeolian`, `\mixolydian`, `\lydian`, `\phrygian` et `\dorian`.
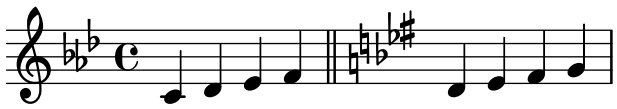
```
\relative {
  \key g \major
  fis''1
  f
  fis
}
```



Rien n'empèche de définir d'autres modes, en listant l'altération de chacun des degrés de la gamme en partant du do.

```
freygish = #`((0 . ,NATURAL) (1 . ,FLAT) (2 . ,NATURAL)
    (3 . ,NATURAL) (4 . ,NATURAL) (5 . ,FLAT) (6 . ,FLAT))
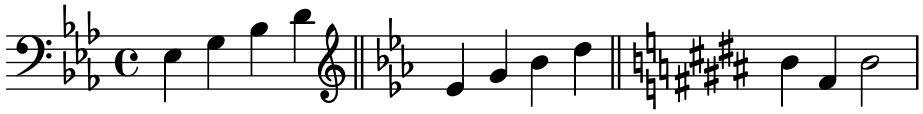
\relative {
  \key c \freygish c'4 des e f
  \bar "||" \key d \freygish d es fis g
}
```



Les altérations à la clef peuvent s'imprimer à des octaves différents de leur position traditionnelle ou à plusieurs octaves, à l'aide des propriétés `flat-positions` et `sharp-positions` de l'objet `KeySignature`. Les entrées fournies à ces propriétés définissent l'amplitude des positions sur la portée où les altérations seront imprimées. Dans le cas où l'entrée est constituée d'une position unique, les altérations seront placées à l'intérieur de l'octave finissant à cette position sur la portée.

```
\override Staff.KeySignature.flat-positions = #'((-5 . 5))
\override Staff.KeyCancellation.flat-positions = #'((-5 . 5))
\clef bass \key es \major es g bes d'
\clef treble \bar "||" \key es \major es' g' bes' d''

\override Staff.KeySignature.sharp-positions = #'(2)
\bar "||" \key b \major b' fis' b'2
```
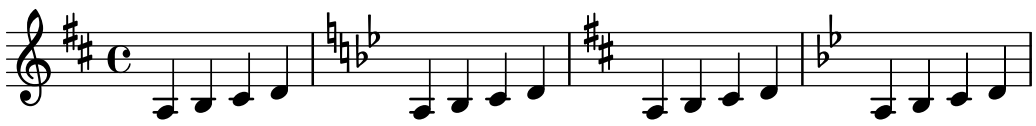


Morceaux choisis

Suppression des bécarres superflus lors d'un changement de tonalité

Après un changement de tonalité, un bécarre est imprimé pour annuler toute altération précédente. Ce comportement s'annule en désactivant la propriété `printKeyCancellation` du contexte `Staff`.

```
\relative c' {
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
  \set Staff.printKeyCancellation = ##f
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
}
```



Armures inhabituelles

La commande `\key` détermine la propriété `keyAlterations` d'un contexte `Staff`. Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété.
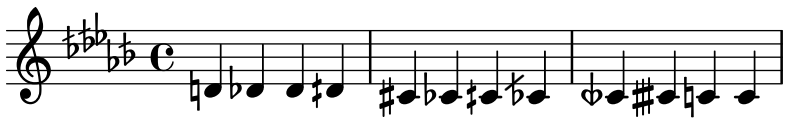
Il s'agit en l'occurrence de définir une liste :

```
\set Staff.keyAlterations =
  #`(((octave . pas) . altération) ((octave . pas) . altération)
  @dots{})
```

dans laquelle, et pour chaque élément, `octave` spécifie l'octave (0 pour celle allant du do médium au si supérieur), `pas` la note dans cette octave (0 pour do et 6 pour si), et `altération` sera `,SHARP` ou `,FLAT` ou `,DOUBLE-SHARP`, etc. (attention à la virgule en préfixe).

Une formulation abrégée – `(pas . altération)` – signifie que l'altération de l'élément en question sera valide quelle que soit l'octave. En ce qui concerne les gammes microtonales dans lesquelles un « dièse » n'est pas d'un centième, `altération` se réfère à un deux-centième de ton entier.

```
\include "arabic.ly"
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                 (1 . ,SEMI-FLAT)
                                 (2 . ,FLAT)
                                 (5 . ,FLAT)
                                 (6 . ,SEMI-FLAT))
%\set Staff.extraNatural = ##f
  re reb \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dodsd do do |
}
```



Voir aussi

Glossaire musicologique : Section “mode d’église” dans *Glossaire*, Section “scordatura” dans *Glossaire*.

Manuel d’initiation : Section “Hauteurs et armure” dans *Manuel d’initiation*.

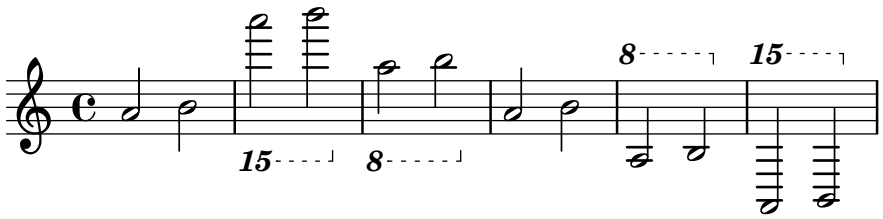
Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Key_engraver” dans *Référence des propriétés internes*, Section “Key_performer” dans *Référence des propriétés internes*, Section “KeyCancellation” dans *Référence des propriétés internes*, Section “KeyChangeEvent” dans *Référence des propriétés internes*, Section “KeySignature” dans *Référence des propriétés internes*, Section “key-signature-interface” dans *Référence des propriétés internes*.

Marques d’octaviation

Les marques d’octaviation, *Ottava*, permettent d’introduire une transposition spécifique d’une octave pour la portée en cours. C’est la fonction `ottava` qui s’en charge.

```
\relative c'' {
  a2 b
  \ottava #-2
  a2 b
  \ottava #-1
  a2 b
  \ottava #0
  a2 b
  \ottava #1
  a2 b
  \ottava #2
  a2 b
}
```
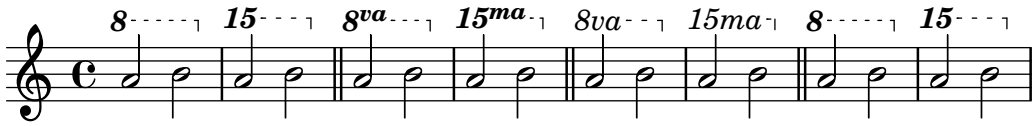


Par défaut s’imprimera un simple nombre en début de crochet. Ce réglage est modifiable afin d’obtenir un nombre ordinal dont l’abréviation peut se positionner en petite lettre haute ou en caractère normal (réglage par défaut auparavant) ; la graisse par défaut de ces caractères est elle aussi modifiable – voir [Sélection de la fonte et de la taille], page 265.

L’exemple suivant illustre différentes options, ainsi que le moyen de retrouver le comportement par défaut.

```
\relative c'' {
  \ottava #1
  a'2 b
  \ottava #2
  a'2 b
  \bar "||"
  \set Staff.ottavationMarkups = #ottavation-ordinals
```

```
  \ottava #1
  a,2 b
  \ottava #2
  a'2 b
  \bar "||"
  \override Staff.OttavaBracket.font-series = #'medium
  \set Staff.ottavationMarkups = #ottavation-simple-ordinals
  \ottava #1
  a,2 b
  \ottava #2
  a'2 b
  \bar "||"
  \revert Staff.OttavaBracket.font-series
  \set Staff.ottavationMarkups = #ottavation-numbers
  \ottava #1
  a,2 b
  \ottava #2
  a'2 b
}
```



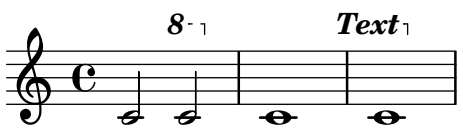
Morceaux choisis

Modification du texte des marques d'octaviation

En interne, la fonction `\ottava` détermine les propriétés `ottavation` (par ex. en `"8va"` ou `"8vb"`) et `middleCPosition`. Vous pouvez modifier le texte d'une marque d'octaviation en définissant `ottavation` après avoir fait appel à `ottava`.

Un texte bref est particulièrement utile lorsque l'octaviation est courte.

```
{
  c'2
  \ottava #1
  \set Staff.ottavation = #"8"
  c''2
  \ottava #0
  c'1
  \ottava #1
  \set Staff.ottavation = #"Text"
  c''1
}
```

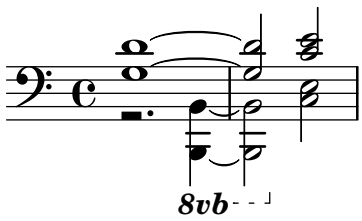


Ajout d'une indication d'octave pour une seule voix

Lorsque plusieurs voix cohabitent sur une même portée, déterminer l'octaviation d'une voix affectera la position des notes de toutes les voix, jusqu'à la fin du crochet d'octaviation. Si l'octaviation ne doit s'appliquer qu'à une seule voix, les positionnements du do central (propriété

`middleCPosition`) et du crochet d'octaviation peuvent s'indiquer explicitement. Dans l'exemple suivant, le `middleCPosition` qui a normalement une valeur de 6 en clef de fa – soit six crans au-dessus de la ligne médiane – est déterminé à 7 crans au-dessus (une octave) pour la durée de l'*ottava*.

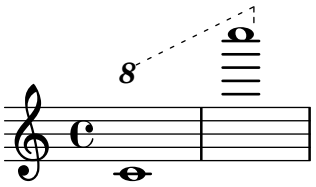
```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
    {
      r2.
      \set Staff.ottavation = #"8vb"
      \once \override Staff.OttavaBracket.direction = #DOWN
      \set Voice.middleCPosition = #(+ 6 7)
      <b,,, b,,>4 ~ |
      q2
      \unset Staff.ottavation
      \unset Voice.middleCPosition
      <c e>2
    }
  >>
}
```



Modification de la pente de l'extension d'octaviation

Il est possible d'adapter la pente d'une indication d'octaviation.

```
\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0) ; Change the integer here
                (attach-dir . ,LEFT)
                (padding . 0)
                (stencil-align-dir-y . ,CENTER)))
       (right . ((Y . 5) ; Change the integer here
                 (padding . 0)
                 (attach-dir . ,RIGHT)
                 (text . ,(make-draw-dashed-line-markup
                           (cons 0 -1.2))))))
  \override Staff.OttavaBracket.left-bound-info =
     #ly:line-spanner::calc-left-bound-info-and-text
  \override Staff.OttavaBracket.right-bound-info =
     #ly:line-spanner::calc-right-bound-info
  \ottava #1
  c1
  c'''1
}
```



Voir aussi

Glossaire musicologique : Section “octaviation” dans *Glossaire*.

Manuel de notation : [Sélection de la fonte et de la taille], page 265.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Ottava_spanner_engraver” dans *Référence des propriétés internes*, Section “OttavaBracket” dans *Référence des propriétés internes*, Section “ottava-bracket-interface” dans *Référence des propriétés internes*.

Instruments transpositeurs
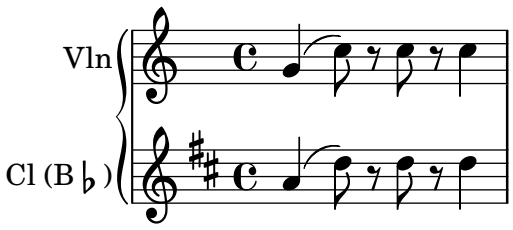
Lorsque l’on saisit une partition d’ensemble incluant des instruments transpositeurs, certaines parties peuvent être dans une autre tonalité que la *tonalité de concert*. Il faudra en pareil cas indiquer la tonalité spécifique de ces *instruments transpositeurs*, sous peine de fichier MIDI erroné et de citations incorrectes. Pour plus de détails sur les citations, consultez le chapitre [Citation d’autres voix], page 221.

`\transposition` *`hauteur`*

La hauteur donnée en argument à `\transposition` doit correspondre à la note entendue lorsqu’un `do` écrit sur la portée est joué par l’instrument transpositeur. Cette hauteur doit être mentionnée en *mode absolu*. Par exemple, lorsque vous saisissez une partition en notes réelles, toutes les voix devraient être en ut ; si un instrument joue un ton au dessus, il faudra lui ajouter un `\transposition d'`. La commande `\transposition` s’utilise **si, et seulement si** les notes à saisir **ne sont pas** dans la tonalité de concert.

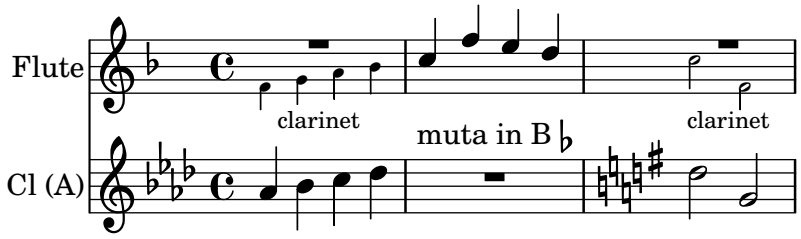
Voici un fragment pour violon et clarinette en si bémol (*B-flat*) pour lequel les parties respectives ont été recopiées à partir du conducteur. Les deux instruments sont à l’unisson.

```
\new GrandStaff <<
  \new Staff = "violin" \with {
    instrumentName = "Vln"
    midiInstrument = "violin"
   }
  \relative c'' {
    % not strictly necessary, but a good reminder
    \transposition c'
    \key c \major
    g4( c8) r c r c4
  }
  \new Staff = "clarinet" \with {
    instrumentName = \markup { Cl (B\flat) }
    midiInstrument = "clarinet"
  }
  \relative c'' {
    \transposition bes
    \key d \major
    a4( d8) r d r d4
  }
>>
```



La `\transposition` peut évoluer au cours d'un morceau. Un clarinettiste peut être amené à jongler avec une clarinette en la et une autre en si bémol.

```
flute = \relative c'' {
  \key f \major
  \cueDuring "clarinet" #DOWN {
    R1 _\markup\tiny "clarinet"
    c4 f e d
    R1 _\markup\tiny "clarinet"
  }
}
clarinet = \relative c'' {
  \key aes \major
  \transposition a
  aes4 bes c des
  R1^\markup { muta in B\flat }
  \key g \major
  \transposition bes
  d2 g,
}
\addQuote "clarinet" \clarinet
<<
  \new Staff \with { instrumentName = "Flute" }
    \flute
  \new Staff \with { instrumentName = "Cl (A)" }
    \clarinet
>>
```



Voir aussi

Glossaire musicologique : Section "tonalité de concert" dans *Glossaire*, Section "instrument transpositeur" dans *Glossaire*.

Manuel de notation : [Citation d'autres voix], page 221, [Transposition], page 11.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Altérations accidentelles automatiques

LilyPond dispose d'une fonction chargée de regrouper les règles suivant lesquelles s'impriment les altérations. Elle s'invoque de la manière suivante :

```
\new Staff <<
  \accidentalStyle voice
```

```
  { ... }
>>
```

La règle de gestion des altérations s'applique par défaut au contexte `Staff` en cours, exception faite des styles `piano` et `piano-cautionary` comme nous allons le voir. Cette fonction accepte un éventuel argument supplémentaire chargé de spécifier le champ d'action de la règle à suivre. À titre d'exemple, il faudra utiliser, pour que toutes les portées d'un même système – contexte `StaffGroup` – soient soumises à la même règle :

```
\accidentalStyle StaffGroup.voice
```

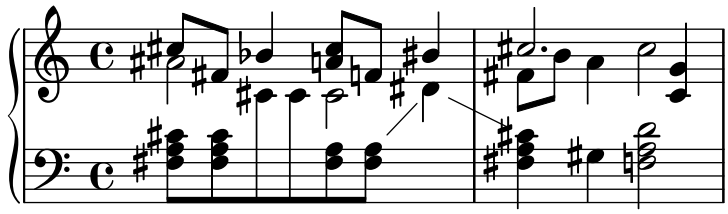
Nous vous présentons ci-après les différentes règles d'altération prises en charge. Pour les besoins de la démonstration, nous partirons de l'exemple suivant :

```
musicA = {
  <<
    \relative {
      cis''8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\
    \relative {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative {
      <fis a cis>8[ <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
      <fis, a> <fis a>]
      \showStaffSwitch
      \change Staff = up
      dis'4 |
      \change Staff = down
      <fis, a cis>4 gis <f a d>2 |
    }
  }
}

\new PianoStaff {
  <<
    \new Staff = "up" {
      \accidentalStyle default
      \musicA
    }
    \new Staff = "down" {
      \accidentalStyle default
      \musicB
```

```
    }
  >>
}
```

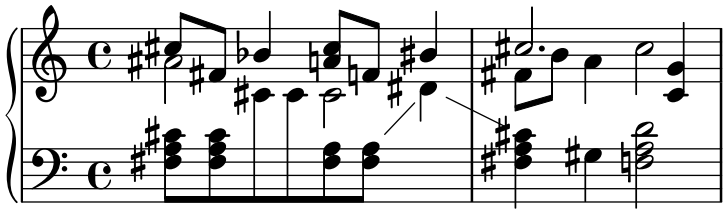


Notez bien que pour appliquer le même style aux deux portées, seules les dernières lignes de cet exemple nous intéressent.

```
\new PianoStaff {
  <<
    \new Staff = "haut" {
      %%% voici la ligne à modifier en conséquence :
      \accidentalStyle Score.default
      \musicA
    }
    \new Staff = "bas" {
      \musicB
    }
  >>
}
```

`default`

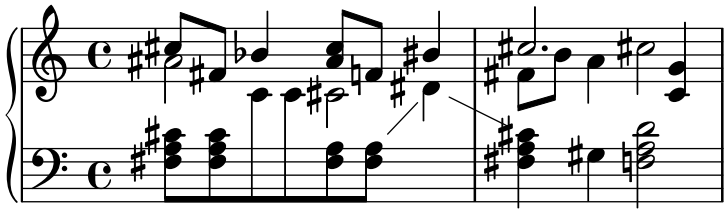
C'est la règle d'impression par défaut, qui se rapporte à l'usage en vigueur au XVIII^{e} siècle : les altérations accidentelles sont valables toute une mesure, et uniquement à leur propre octave. C'est la raison pour laquelle il n'y a pas de bécarre avant le *si* de la deuxième mesure, ni avant le dernier *do*.



`voice`

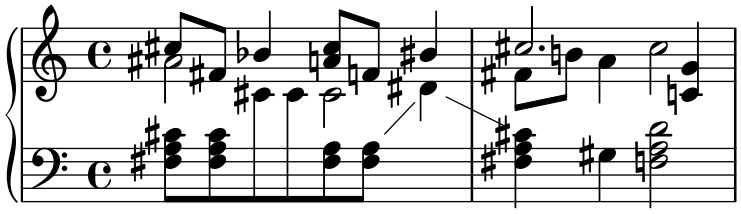
En principe, LilyPond se souvient de toutes les altérations présentes sur la portée (contexte `Staff`). Avec cette règle, cependant, les altérations sont indépendantes pour chacune des voix tout en obéissant à la règle `default`.

Les altérations d'une voix sont de fait ignorées dans les autres voix, ce qui peut donner lieu à un résultat malencontreux. Dans l'exemple suivant, il est difficile de dire si le deuxième *la* est dièse ou naturel. La règle `voice` n'est donc à envisager que dans le cas de voix devant être lues par des musiciens différents. S'il s'agit d'un « conducteur », ou d'une portée destinée à un seul musicien, il vaut mieux utiliser `modern` ou `modern-cautionary`.



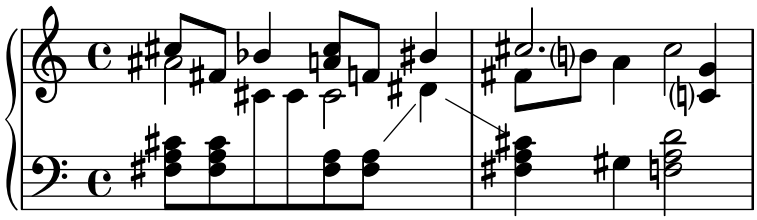
`modern`

Cette règle est la plus courante au XX^{e} siècle. Certains bécarres ne sont pas imprimés, comme il était d'usage lorsqu'une note diésée suit cette même note flanquée d'un double dièse, ou bien un bémol un double bémol. Le style `modern` suit la même règle que le style `default`, avec deux additions afin de lever les ambiguïtés : lorsqu'une note non altérée apparaît à une octave différente, ou bien dans la mesure suivante, des bécarres de précaution sont ajoutés. Dans l'exemple suivant, notez ainsi les deux bécarres dans la deuxième mesure de la main droite.



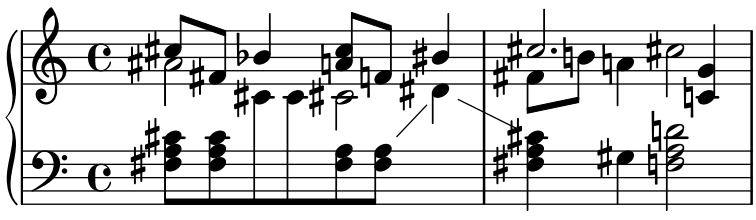
`modern-cautionary`

Cette règle est équivalente à `modern`, mais les bécarres de précaution (absents dans la règle `default`) sont imprimés entre parenthèses. Ils peuvent aussi adopter une taille différente, au moyen de la propriété `font-size` de l'objet `AccidentalSuggestion`.



`modern-voice`

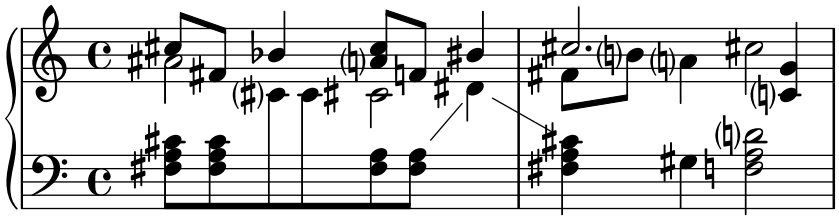
Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix, mais les autres voix d'un même contexte `Staff` en *tiennent compte* cette fois. C'est pourquoi le *la* de la dernière mesure est affublé d'un bécarre bien qu'il y en ait déjà eu un dans la mesure précédente, et que le *ré* de la main gauche en ait un alors que le dièse qu'il avait auparavant concernait la main droite.



`modern-voice-cautionary`

Cette règle est similaire à la précédente, mais les altérations de précaution (celles que n'aurait pas ajoutées `voice`), sont imprimées de façon particulière. On retrouve

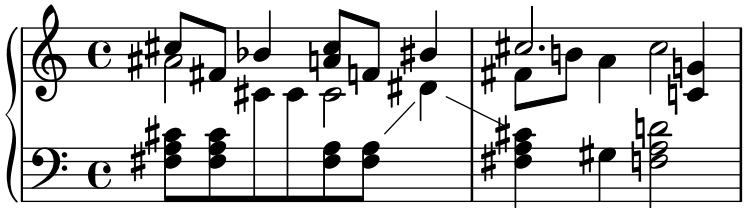
donc toutes les altérations qu'imprimerait `default`, mais certaines sont considérées comme étant « de précaution ».



`piano`

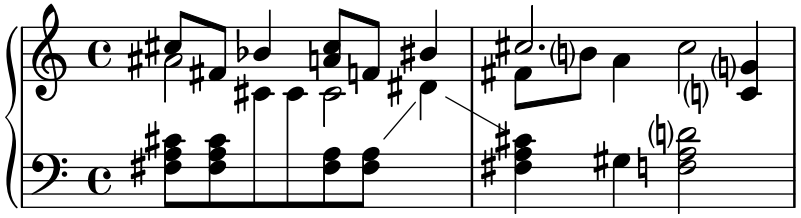
Cette règle est communément employée pour les partitions de piano au XX^{e} siècle. Très similaire à `modern` de par son comportement, elle s'en distingue en ce que les altérations tiennent compte des autres portées du contexte `GrandStaff` ou `PianoStaff`.

Cette règle s'applique par défaut dans un `GrandStaff` et dans un `PianoStaff`.



`piano-cautionary`

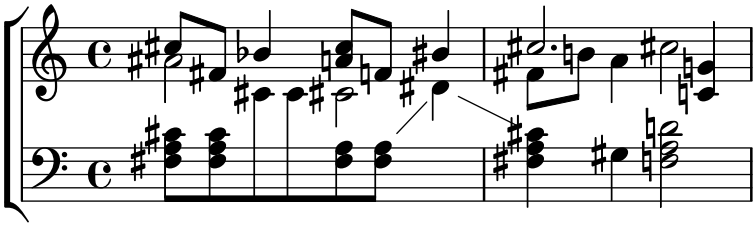
Identique au style `piano`, mais les altérations de précaution sont imprimées différemment.



`choral`

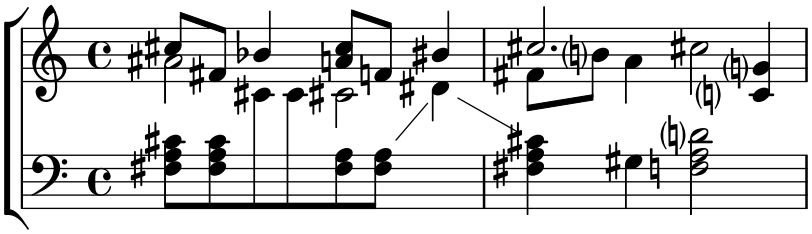
Cette règle est une combinaison des styles `modern-voice` et `piano`. Les altérations accidentelles sont indiquées aussi bien pour un chanteur qui suit seulement sa voix, que pour un lecteur suivant toutes les voix d'un `ChoirStaff`.

Ce style d'altération s'applique, par défaut, au `ChoirStaff` en cours.



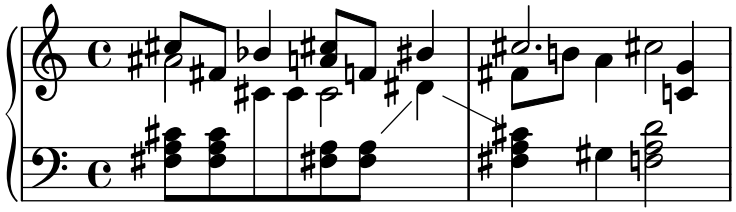
`choral-cautionary`

Identique au style `choral`, mais les altérations de précaution sont imprimées différemment.



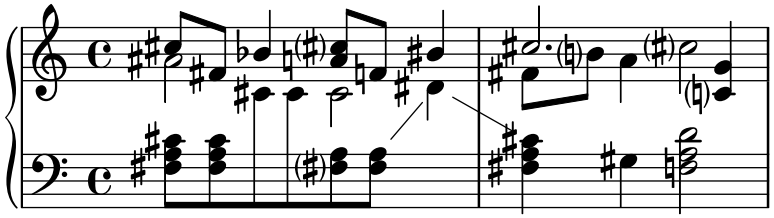
`neo-modern`

Cette règle suit les pratiques de la musique contemporaine : les altérations accidentelles apparaissent comme dans le style `modern`, à ceci près qu'elles sont répétées dans la même mesure – sauf si elles concernent deux notes consécutives.



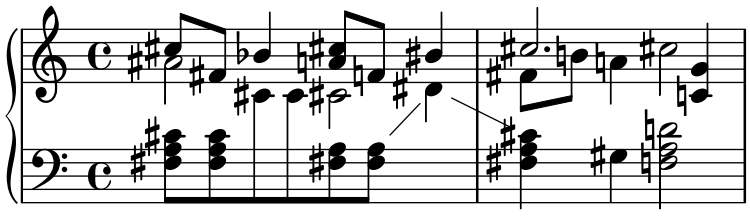
`neo-modern-cautionary`

Identique au style `neo-modern`, mais les altérations de précaution sont imprimées entre parenthèses. Elles peuvent aussi adopter une taille différente, au moyen de la propriété `font-size` de l'objet `AccidentalSuggestion`.



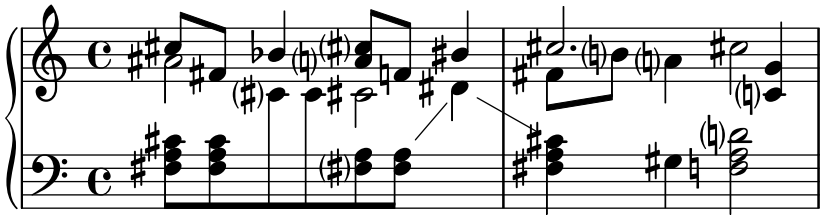
`neo-modern-voice`

Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix comme avec le style `neo-modern` mais les autres voix dans le même contexte `Staff` en tiennent aussi compte.



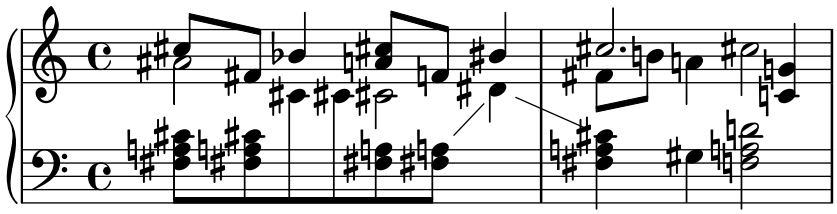
`neo-modern-voice-cautionary`

Cette règle est identique à `neo-modern-voice`, mais les altérations de précaution sont imprimées soit entre parenthèses (par défaut), soit en plus petit.



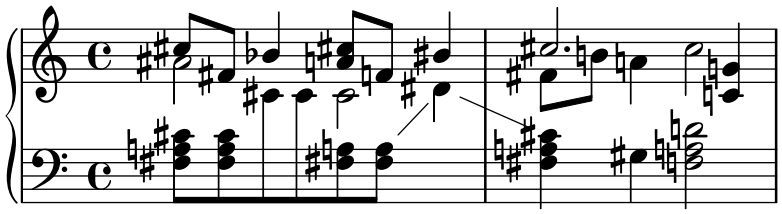
`dodecaphonic`

Cette règle reproduit ce que certains compositeurs du début du XX^{e} siècle ont introduit dans leur désir d'abolir la distinction entre les notes naturelles ou non. Ainsi, **chaque** note est affublée d'une altération, même si elle est naturelle.



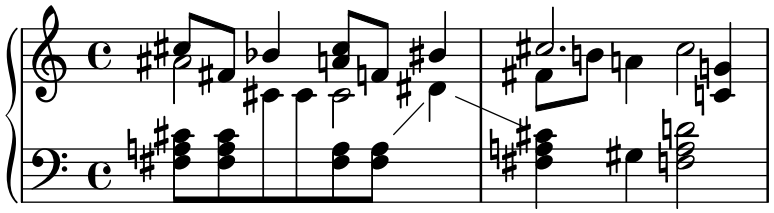
`dodecaphonic-no-repeat`

Comme dans le cas du style `dodecaphonic`, chaque note est par défaut affublée d'une altération. Celle-ci sera toutefois omise lorsque la même hauteur est immédiatement répétée dans la même portée.
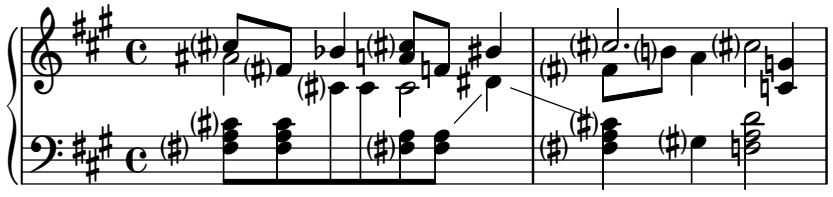


`dodecaphonic-first`

Comme dans le cas du style `dodecaphonic`, chaque note est par défaut affublée d'une altération. Cette altération n'apparaîtra que pour la première occurrence dans la mesure et seront répétés en cas d'octave différente.



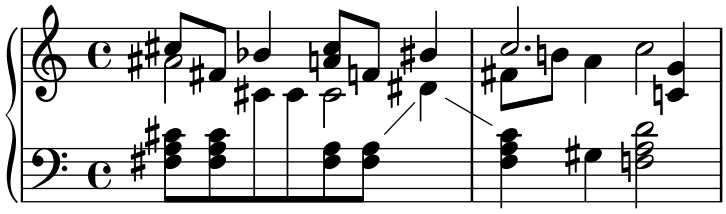
`teaching`

Cette règle est à usage pédagogique : l'impression d'une simple gamme fera apparaître une altération de précaution pour toute note altérée. Les altérations accidentelles sont imprimées selon le style `modern`, et une altération de précaution est ajoutée pour chaque dièse ou bémol à la clef – sauf dans le cas de notes consécutives.



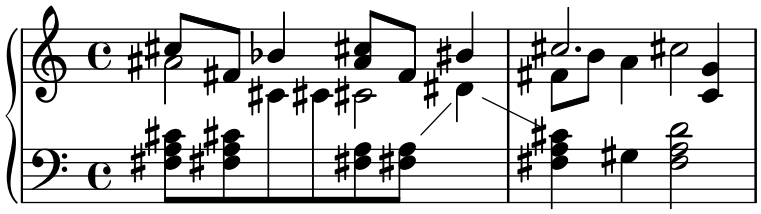
`no-reset`

C'est la même règle que `default`, mais l'effet des altérations accidentelles ne cesse jamais, même dans les mesures suivantes.



`forget`

Tout le contraire de `no-reset` : l'effet des altérations cesse aussitôt ; toutes les altérations, quelle que soit leur place dans la mesure, sont de ce fait imprimées en fonction de l'éventuelle armure.



Voir aussi

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

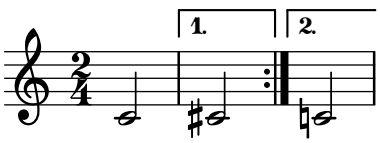
Référence des propriétés internes : Section "Accidental" dans *Référence des propriétés internes*, Section "Accidental_engraver" dans *Référence des propriétés internes*, Section "GrandStaff" dans *Référence des propriétés internes* et Section "PianoStaff" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*, Section "AccidentalSuggestion" dans *Référence des propriétés internes*, Section "AccidentalPlacement" dans *Référence des propriétés internes*. Section "accidental-suggestion-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les notes simultanées sont considérées comme des évènements séquentiels. Ceci implique que, dans un accord, les altérations accidentelles seront imprimées comme si les notes de l'accord apparaissaient l'une après l'autre, en fonction de l'ordre dans lequel elles ont été saisies – ce qui peut poser problème lorsqu'au sein d'un accord certaines altérations dépendent les unes des autres. Ce problème est à résoudre manuellement, en insérant des `!` et des `?` après les notes concernées, tel '`<f! fis!>`'.

L'absence d'altération de précaution est déterminée par l'examen de la mesure précédente.

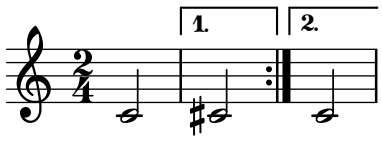
Dans le cas d'une reprise, la logique voudrait que la présence d'altération de précaution d'une alternativee soit basée sur la dernière mesure *jouée* plutôt que la dernière *imprimée*. Dans l'exemple qui suit, vous conviendrez que le do de la seconde alternative ne nécessite pas son bécarre.



L'astuce suivante, qui définit temporairement le recours au style `forget`, permet d'obtenir quelque chose de présentable.

```
forget = #(define-music-function (music) (ly:music?) #{
  \accidentalStyle forget
  #music
  \accidentalStyle modern
```

```
#})
{
  \accidentalStyle modern
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
    \volta 1 { cis' }
    \volta 2 { \forget c' }
  }
}
```
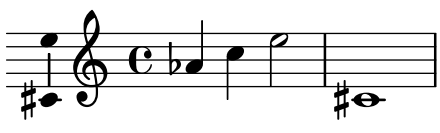


Ambitus

L'*ambitus* est l'amplitude des hauteurs d'une voix donnée dans une partition. Ce terme peut aussi désigner la tessiture qu'un instrument est capable d'atteindre. Souvent, cet ambitus est imprimé au début des partitions vocales, afin que les exécutants puissent voir au premier coup d'œil s'ils sont en mesure de tenir la partie en question.

Pour exprimer l'ambitus d'une pièce, on indique avant la clef deux têtes de note représentant la hauteur la plus basse et la plus haute. Les éventuelles altérations accidentelles seront automatiquement ajoutées.

```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\relative {
  aes' c e2
  cis,1
}
```
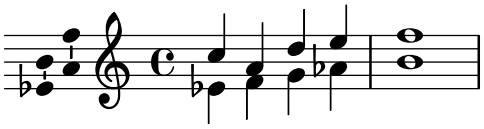


Morceaux choisis

Un ambitus par voix

L'`ambitus` peut être individualisé par voix. Il faut en pareil cas éviter qu'ils se chevauchent.

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = #2.0
```
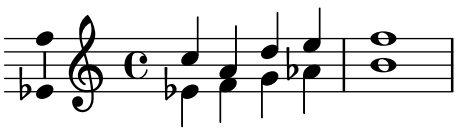
```
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Ambitus sur plusieurs voix

Si plusieurs voix se trouvent sur une même portée, on peut attribuer le graveur `Ambitus_engraver` au contexte `Staff` afin d'obtenir l'ambitus sur toutes les voix cumulées, non d'une seule des voix actives.

```
\new Staff \with {
  \consists "Ambitus_engraver"
  }
<<
  \new Voice \relative c'' {
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Réglage de l'affichage d'un ambitus

L'affichage d'un *ambitus* peut s'affiner pour répondre à vos préférences en matière d'esthétique.

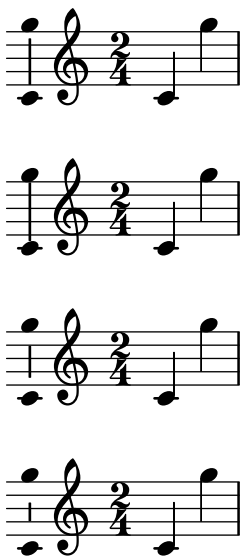
```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}
```

```
\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1.5
  c'4 g''
}
```
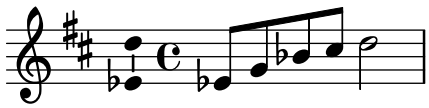


Ambitus après l'armure

L'ambitus se place par défaut à gauche de la clef. La fonction `\ambitusAfter` permet cependant de modifier ce positionnement ; sa syntaxe est `\ambitusAfter` *`grob-interface`* – voir Section "Graphical Object Interfaces" dans *Référence des propriétés internes* pour une liste des valeurs de *`grob-interface`* possibles. L'un des cas d'usage est d'insérer l'ambitus entre l'armure et la métrique.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
```

```
  \key d \major
  es'8 g bes cis d2
}
```



Voir aussi

Glossaire musicologique : Section “ambitus” dans *Glossaire*.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Ambitus_engraver” dans *Référence des propriétés internes*, Section “Voice” dans *Référence des propriétés internes*, Section “Staff” dans *Référence des propriétés internes*, Section “Ambitus” dans *Référence des propriétés internes*, Section “AmbitusAccidental” dans *Référence des propriétés internes*, Section “AmbitusLine” dans *Référence des propriétés internes*, Section “AmbitusNoteHead” dans *Référence des propriétés internes*, Section “ambitus-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

LilyPond ne gère pas les collisions entre plusieurs ambitus présents sur une même portée.
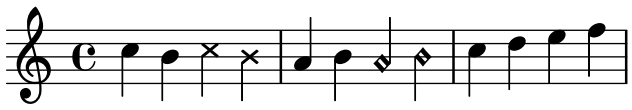
1.1.4 Têtes de note

Nous allons voir dans ce chapitre comment modifier l’aspect des têtes de note.

Têtes de note spécifiques

L’apparence des têtes de note peut évoluer au cours de la partition :

```
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
  \override NoteHead.style = #'harmonic
  a b
  \revert NoteHead.style
  c4 d e f
}
```

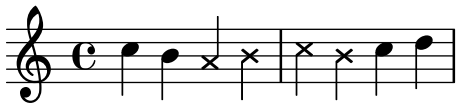


Pour une liste exhaustive des styles de tête de note, consultez Section A.9 [Styles de tête de note], page 740.

Certains instruments utilisent des têtes de note différentes à des fins spécifiques – des croix (style `cross`) pour le *parlato* des chanteurs ou les notes étouffées des guitares :

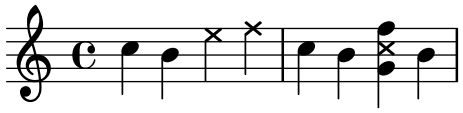
```
\relative {
  c''4 b
  \xNotesOn
   a b c4 b
```

```
  \xNotesOff
  c4 d
}
```



Cette commande opère aussi bien sur des notes isolées qu'au sein d'un accord, dans une portée traditionnelle ou dans un contexte de tablature :
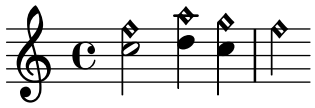
```
\relative {
  c''4 b
  \xNote { e f }
  c b < g \xNote c f > b
}
```



Vous pouvez utiliser, en lieu et place de `\xNote`, `\xNotesOn` et `\xNotesOff`, les commandes `\deadNote`, `\deadNotesOn` et `\deadNotesOff`.

Il existe un raccourci pour les notes en losange :

```
\relative c'' {
  <c f\harmonic>2 <d a'\harmonic>4 <c g'\harmonic> f\harmonic
}
```



Commandes prédéfinies

`\harmonic`, `\xNotesOn`, `\xNotesOff`, `\xNote`.

Voir aussi

Manuel de notation : [Indication des harmoniques et notes étouffées], page 419, [Notes en accords], page 173, Section A.9 [Styles de tête de note], page 740.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

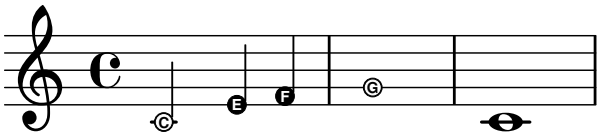
Référence des propriétés internes : Section "note-event" dans *Référence des propriétés internes*, Section "Note_heads_engraver" dans *Référence des propriétés internes*, Section "Ledger_line_engraver" dans *Référence des propriétés internes*, Section "NoteHead" dans *Référence des propriétés internes*. Section "LedgerLineSpanner" dans *Référence des propriétés internes*, Section "note-head-interface" dans *Référence des propriétés internes*, Section "ledger-line-spanner-interface" dans *Référence des propriétés internes*.

Têtes de note avec nom de note

Les notes « easy play » comportent le nom de la note à l'intérieur de la tête. On l'utilise dans des partitions pour débutants. L'impression doit être de plus grande taille, afin que les lettres soient lisibles. Voir à ce propos Section 4.2.2 [Définition de la taille de portée], page 585.

```
#(set-global-staff-size 26)
```

```
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



Commandes prédéfinies

`\easyHeadsOn`, `\easyHeadsOff`.

Morceaux choisis

Easy play – chiffres en lieu et place des lettres

En mode « easy play », les têtes de note utilisent la propriété `note-names` attachée à l'objet `NoteHead` pour déterminer ce qui apparaîtra dans la tête. Intervenir sur cette propriété permet d'imprimer un chiffre correspondant au degré dans la gamme.

La création d'un graveur dédié permet de traiter toutes les notes.

```
#(define Ez_numbers_engraver
   (make-engraver
    (acknowledgers
     ((note-head-interface engraver grob source-engraver)
      (let* ((context (ly:translator-context engraver))
             (tonic-pitch (ly:context-property context 'tonic))
             (tonic-name (ly:pitch-notename tonic-pitch))
             (grob-pitch
              (ly:event-property (event-cause grob) 'pitch))
             (grob-name (ly:pitch-notename grob-pitch))
             (delta (modulo (- grob-name tonic-name) 7))
             (note-names
              (make-vector 7 (number->string (1+ delta)))))
        (ly:grob-set-property! grob 'note-names note-names))))))

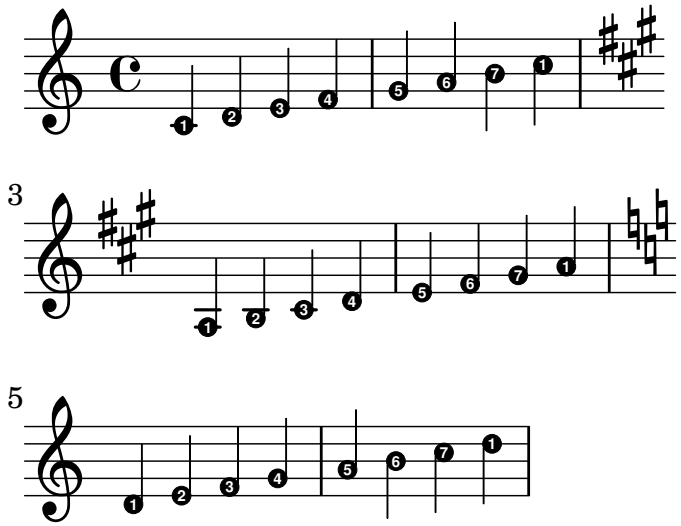
#(set-global-staff-size 26)

\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break
```

```
  \key a \major
  a,4 b cis d
  e4 fis gis a \break

  \key d \dorian
  d,4 e f g
  a4 b c d
}
```



Voir aussi

Manuel de notation : Section 4.2.2 [Définition de la taille de portée], page 585.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.
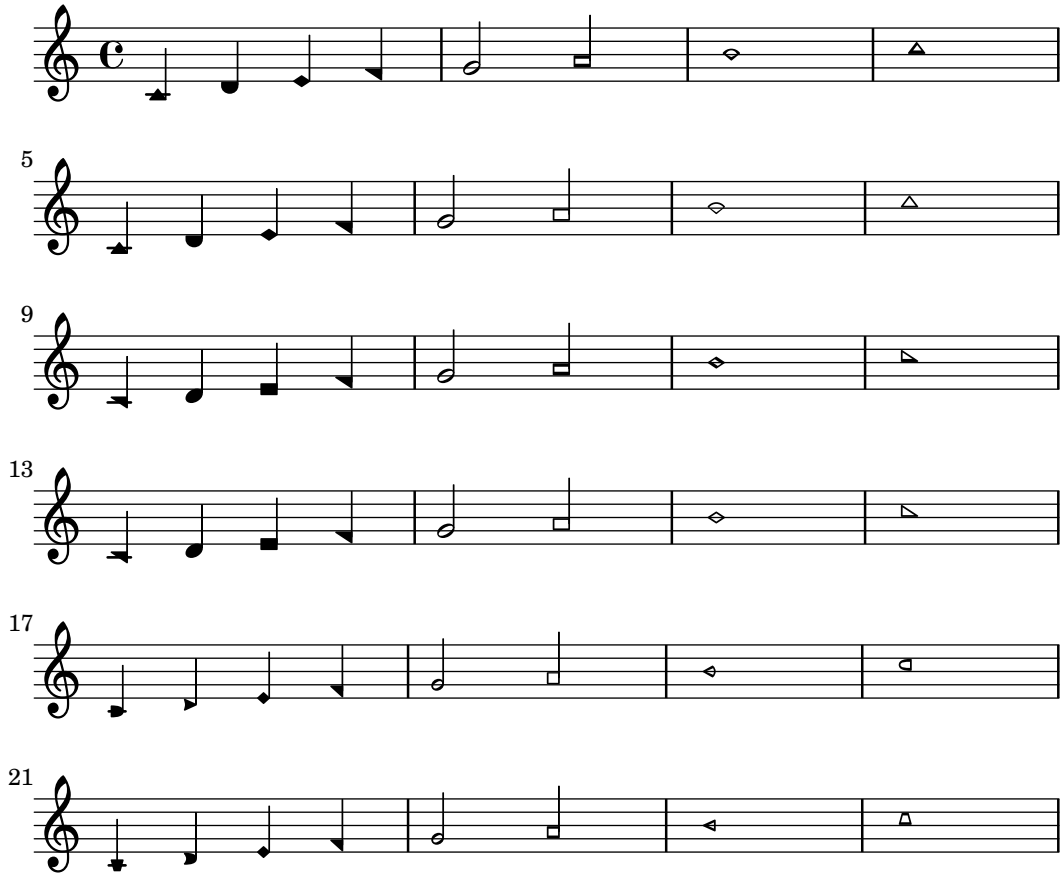
Référence des propriétés internes : Section “note-event” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “NoteHead” dans *Référence des propriétés internes*, Section “note-head-interface” dans *Référence des propriétés internes*.

Têtes de note à forme variable

En notation profilée, le profil d’une tête de note correspond à la fonction harmonique de cette note dans la gamme. Ce style de notation était très en vogue dans les recueils de chansons américains du XIX^{e} siècle. Voici comment procéder :
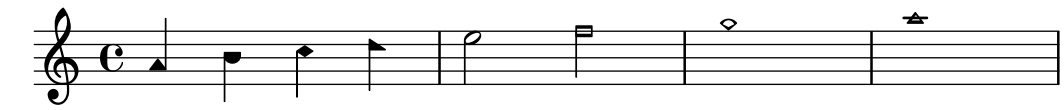
```
\relative c'' {
  \aikenHeads
  c, d e f g2 a b1 c \break
  \aikenThinHeads
  c,4 d e f g2 a b1 c \break
  \sacredHarpHeads
  c,4 d e f g2 a b1 c \break
  \southernHarmonyHeads
  c,4 d e f g2 a b1 c \break
  \funkHeads
  c,4 d e f g2 a b1 c \break
  \walkerHeads
  c,4 d e f g2 a b1 c \break
```
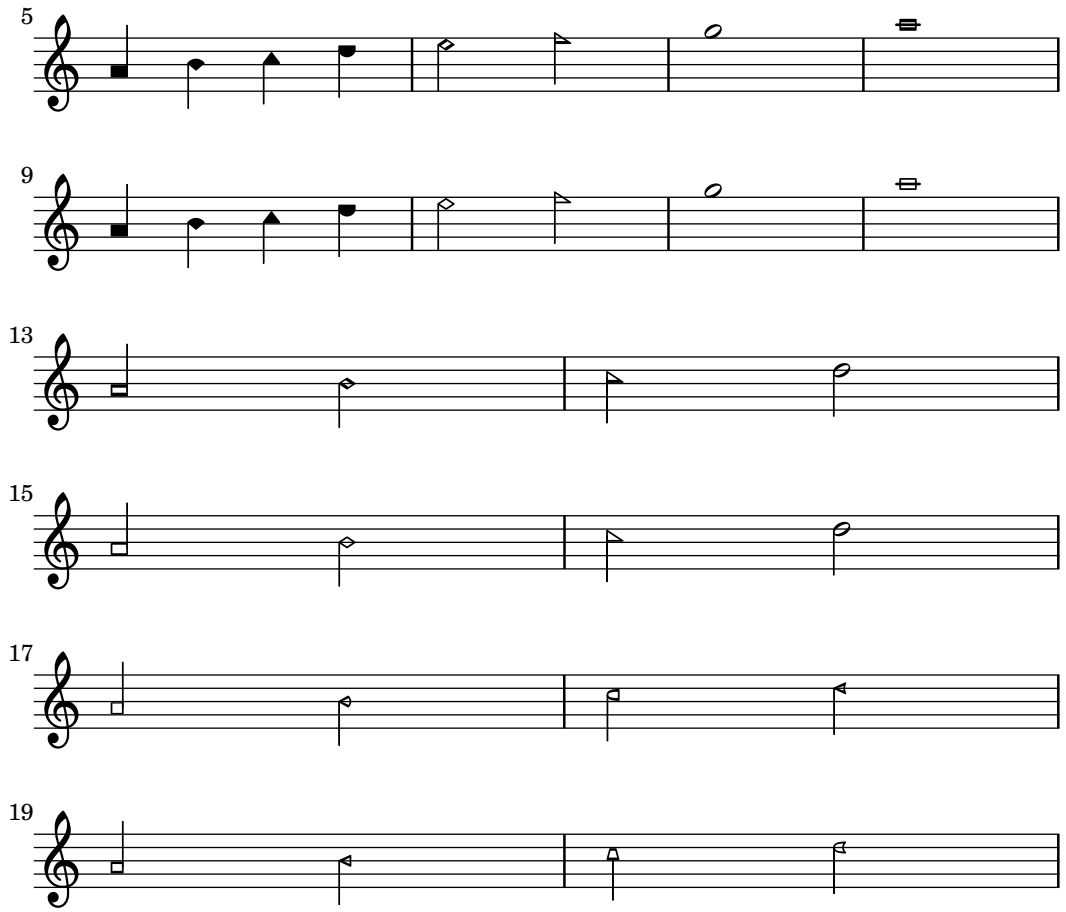
```
}
```



Les profils sont déterminés par la hauteur dans la gamme, le premier degré étant défini par la commande `\key`. Pour une tonalité mineure, les degrés sont déterminés par rapport au relatif majeur :

```
\relative c'' {
  \key a \minor
  \aikenHeads
  a b c d e2 f g1 a \break
  \aikenHeadsMinor
  a,4 b c d e2 f g1 a \break
  \aikenThinHeadsMinor
  a,4 b c d e2 f g1 a \break
  \sacredHarpHeadsMinor
  a,2 b c d \break
  \southernHarmonyHeadsMinor
  a2 b c d \break
  \funkHeadsMinor
  a2 b c d \break
  \walkerHeadsMinor
  a2 b c d \break
}
```





Commandes prédéfinies

`\aikenHeads`, `\aikenHeadsMinor`, `\aikenThinHeads`, `\aikenThinHeadsMinor`, `\funkHeads`, `\funkHeadsMinor`, `\sacredHarpHeads`, `\sacredHarpHeadsMinor`, `\southernHarmonyHeads`, `\southernHarmonyHeadsMinor`, `\walkerHeads`, `\walkerHeadsMinor`.
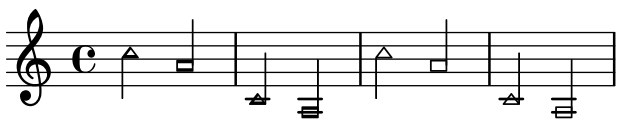
Morceaux choisis

Variante fine des notes profilées Aiken

Les notes profilées Aiken, lorsqu'elles sont blanches, deviennent difficile à distinguer avec des tailles de portée réduites, notamment en présence de ligne supplémentaire. Perdant du blanc sur leur intérieur les fait alors ressembler à des noires.

```
\score {
  {
    \aikenHeads
    c''2 a' c' a

    % Switch to thin-variant noteheads
    \set shapeNoteStyles = ##(doThin reThin miThin
                              faThin sol laThin tiThin)
    c'' a' c' a
  }
}
% END EXAMPLE
```



Profilage des notes selon leur degré dans la gamme

La propriété `shapeNoteStyles` permet d'affecter un profil particulier à chaque degré de la gamme – à partir de l'armure ou de la propriété `tonic`. Ses valeurs sont constituées d'une liste de symboles, qu'il s'agisse de formes géométriques (`triangle`, `cross` ou `xcircle`) ou basés sur la tradition des graveurs américains (avec quelques noms de note latins).

LilyPond dispose de deux raccourcis, `\aikenHeads` et `\sacredHarpHeads`, permettant de reproduire d'anciens recueils de chansons américaines.
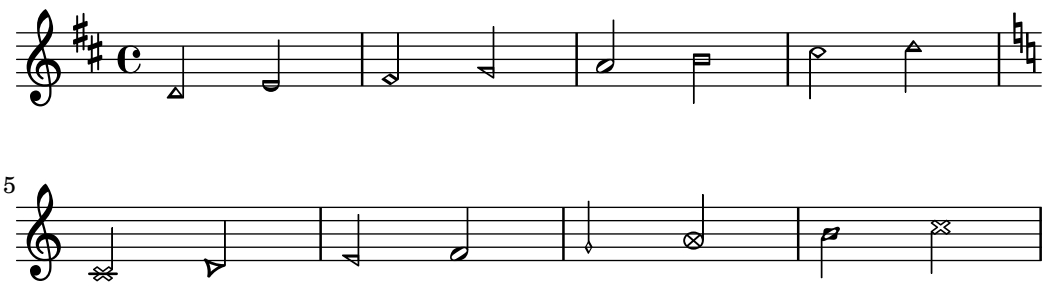
L'exemple suivant montre plusieurs manières de profiler les têtes de note, ainsi que la capacité de transposer tout en respectant la fonction harmonique de chaque note dans la gamme.

```
fragment = {
  \key c \major
  c2 d
  e2 f
  g2 a
  b2 c
}

\new Staff {
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = ##(do re mi fa
                              #f la ti)
    \fragment
  }

  \break

  \relative c' {
    \set shapeNoteStyles = ##(cross triangle fa #f
                              mensural xcircle diamond)
    \fragment
  }
}
```



Pour une liste exhaustive des styles de tête de note, consultez Section A.9 [Styles de tête de note], page 740.

Voir aussi

Manuel de notation : Section A.9 [Styles de tête de note], page 740.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “note-event” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “NoteHead” dans *Référence des propriétés internes*, Section “note-head-interface” dans *Référence des propriétés internes*.

Improvisation

L’improvisation peut quelquefois s’indiquer à l’aide de notes de forme allongée (*slash*). L’interprète jouera alors les notes qu’il veut, en respectant toutefois le rythme affiché. Ces têtes de notes sont créées ainsi :

```
\new Voice \with {
  \consists "Pitch_squash_engraver"
} \relative {
  e''8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  2 ~ 8 f4 f8 ~
  2
  \improvisationOff
  a16( bes) a8 g e
}
```



Commandes prédéfinies

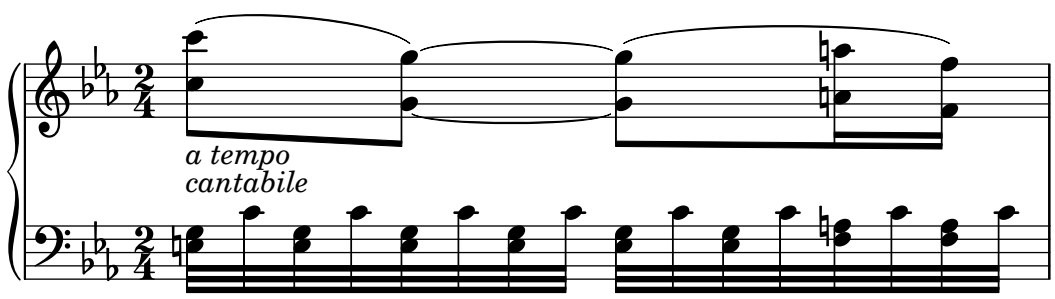
`\improvisationOn`, `\improvisationOff`.
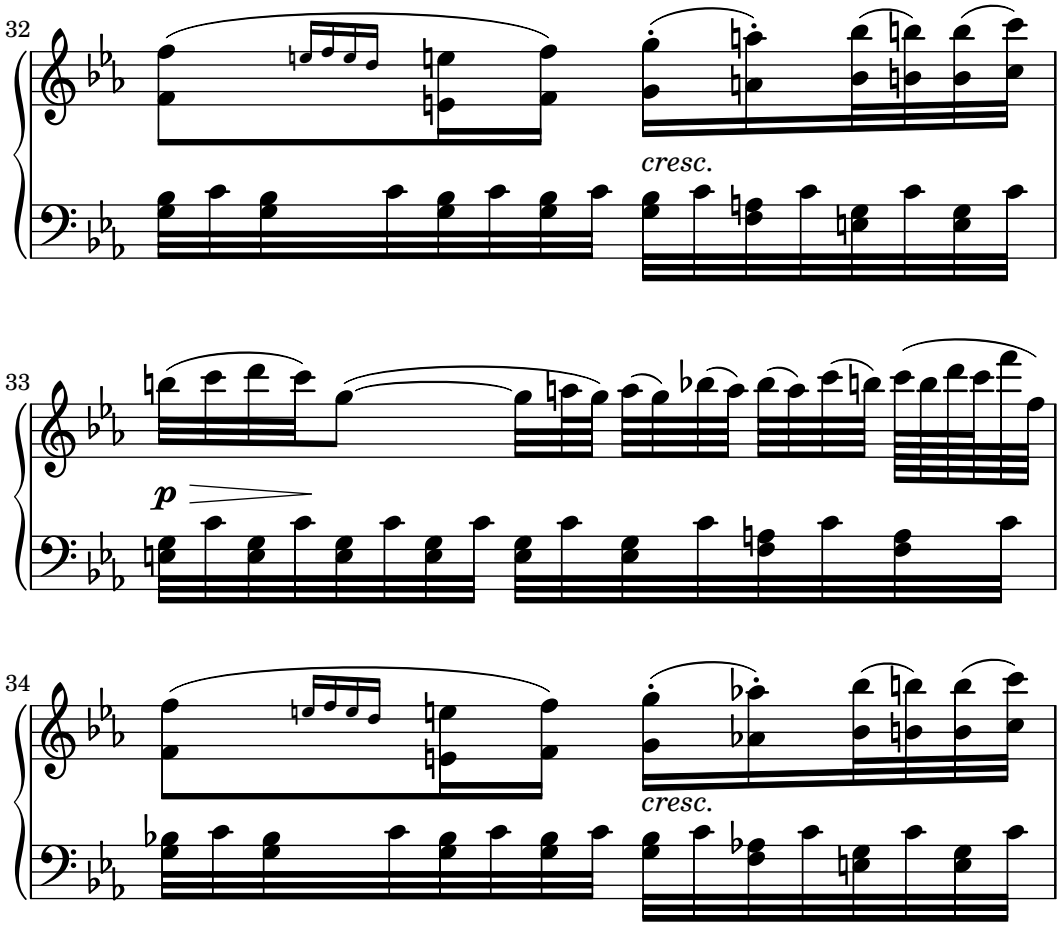
Voir aussi

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Pitch_squash_engraver” dans *Référence des propriétés internes*, Section “Voice” dans *Référence des propriétés internes*, Section “RhythmicStaff” dans *Référence des propriétés internes*.

1.2 Rythme





Cette section traite du rythme : durées, silences, barres de ligature et de mesure.
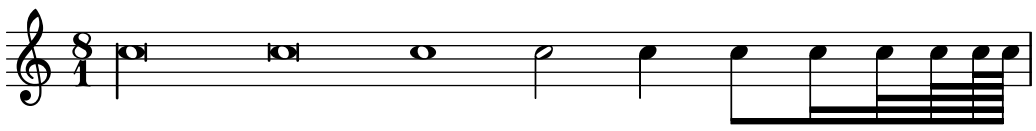
1.2.1 Écriture du rythme

Durées

Dans les modes de notes, d'accords et de paroles, les durées sont écrites avec des chiffres et des points : les durées sont indiquées par leur valeur fractionnaire par rapport à la durée d'une ronde. Une noire, par exemple, qui équivaut à un 1/4 de ronde – *quarter note* en anglais – s'écrit `4`, alors qu'une blanche – *half-note*, 1/2 ronde – s'écrit `2`. Des durées plus courtes que la quintuple croche – 1/128 de ronde – sont possibles, à condition de les ligaturer – voir [Ligatures], page 470.

Pour des notes plus longues qu'une ronde, vous devrez utiliser les commandes `\longa` pour une longue, et `\breve` pour une brève, aussi appelée carrée. Une note dont la durée est de quatre brèves s'obtient par la commande `\maxima` ; celle-ci n'est toutefois disponible que dans le cadre de la notation ancienne. Pour plus de détails, voir Section 2.9 [Notations anciennes], page 468.
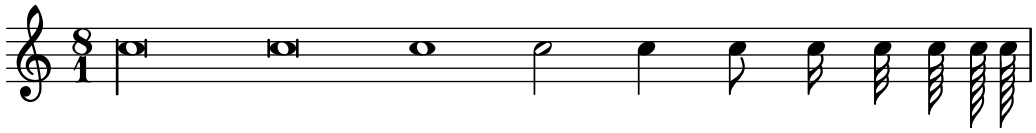
```
\relative {
  \time 8/1
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```



Voici ces mêmes durées sans la fonction de ligature automatique.
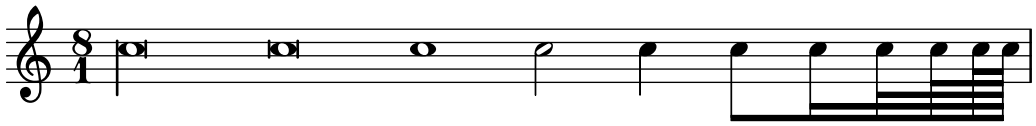
```
\relative {
```

```
  \time 8/1
  \autoBeamOff
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```
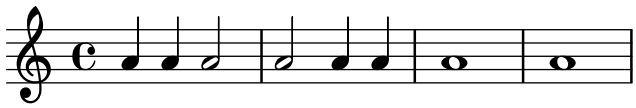


Lorsque une note ou un accord est suivi d'une succession de durées individuelles, celles-ci adopteront la ou les dernières hauteurs mentionnées.

```
\relative {
  \time 8/1
  c'' \longa \breve 1 2
  4 8 16 32 64 128 128
}
```
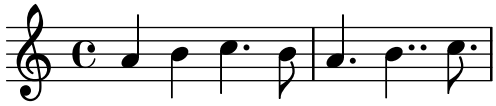


Si la durée d'une note n'est pas précisée, elle est alors assimilée à la durée de la note précédente. La valeur par défaut pour la première note est la noire (4).

```
\relative { a' a a2 a a4 a a1 a }
```



Pour obtenir des notes pointées, ajoutez simplement un point (.) au chiffre. Les notes doublement pointées sont créées de la même façon.

```
\relative { a'4 b c4. b8 a4. b4.. c8. }
```



Les points sont normalement haussés pour éviter les lignes de portée, sauf dans certaines polyphonies. Des commandes prédéfinies permettent de manuellement forcer un positionnement particulier, comme indiqué au chapitre Section 5.4.2 [Direction et positionnement], page 667.

Certaines durées ne peuvent s'obtenir à partir de chiffres et de points, mais uniquement en « liant » deux ou plusieurs notes entre elles. Voir [Liaisons de prolongation], page 57, à ce sujet.

Quant à la manière de spécifier la durée des syllabes ou bien d'aligner des paroles par rapport aux notes, reportez vous au chapitre Section 2.1 [Musique vocale], page 287.

Espacer les notes selon leur durée relative est tout à fait possible. Pour plus de détails à ce sujet et sur les autres réglages propres à cette forme de notation, reportez vous à Section 4.5.5 [Notation proportionnelle], page 617.

Commandes prédéfinies
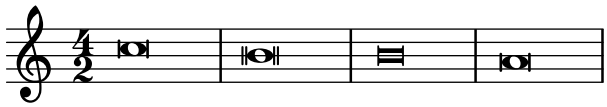
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Morceaux choisis

Brève alternative, avec deux barres verticales

Voici comment obtenir une brève – aussi appelée note carée – flanquée de deux barres verticales, au lieu d'une comme habituellement.
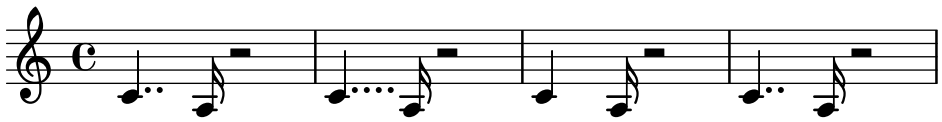
```
\relative c'' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}
```



Spécification du nombre de points d'augmentation d'une note

Le nombre de points d'augmentation affectés à une note en particulier peut se modifier indépendamment des points placés après la note.

```
\relative c' {
  c4.. a16 r2 |
  \override Dots.dot-count = #4
  c4.. a16 r2 |
  \override Dots.dot-count = #0
  c4.. a16 r2 |
  \revert Dots.dot-count
  c4.. a16 r2 |
}
```



Voir aussi

Glossaire musicologique : Section "breve" dans *Glossaire*, Section "longa" dans *Glossaire*, Section "maxima" dans *Glossaire*, Section "valeur des notes" dans *Glossaire*, Section "Noms de durée (notes et silences)" dans *Glossaire*.

Manuel de notation : [Barres de ligature automatiques], page 87, Section 1.2.2 [Écriture des silences], page 61, Section 1.2.1 [Écriture du rythme], page 48, [Hampes], page 245, [Liaisons de prolongation], page 57, [Ligatures], page 470, Section 2.1 [Musique vocale], page 287, Section 2.9 [Notations anciennes], page 468, Section 4.5.5 [Notation proportionnelle], page 617.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Dots" dans *Référence des propriétés internes*, Section "DotColumn" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Il n'existe pas à proprement parler de limite (inférieure ou supérieure) en terme de durée d'un silence. Cependant, le nombre de glyphes disponibles ne couvre que les silences allant de 1 024^{e} de pause (256^{e} de soupir) à la maxime (valant huit pauses).
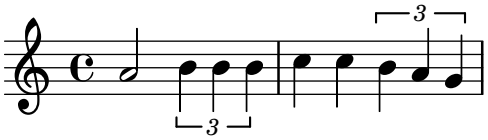
N-olets

Les n-olets – triolets, quintolets, etc. – sont obtenus en multipliant la vitesse d'une expression musicale par une fraction.

```
\tuplet fraction { expression_musicale }
```
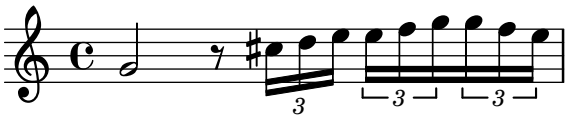
Le numérateur de cette fraction sera imprimé au-dessus ou au-dessous des notes, parfois avec un crochet. Le n-olet le plus courant est le triolet, dans lequel trois notes occupent la durée de deux.

```
\relative {
  a'2 \tuplet 3/2 { b4 4 4 }
  c4 c \tuplet 3/2 { b4 a g }
}
```



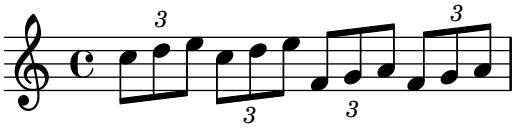
Dans le cas d'une succession de n-olets, saisir la commande `\tuplet` pour chacun des n-olets devient vite fastidieux. LilyPond vous permet de stipuler la durée de base d'un n-olet juste avant l'expression musicale, de telle sorte que les n-olets seront formés automatiquement :

```
\relative {
  g'2 r8 \tuplet 3/2 8 { cis16 d e e f g g f e }
}
```
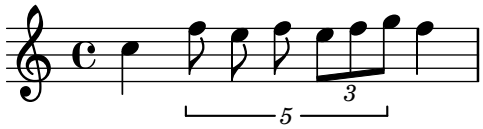


Des commandes prédéfinies permettent de déroger au positionnement automatique du crochet en surplomb ou au-dessous des notes :

```
\relative {
  \tupletUp \tuplet 3/2 { c''8 d e }
  \tupletNeutral \tuplet 3/2 { c8 d e }
  \tupletDown \tuplet 3/2 { f,8 g a }
  \tupletNeutral \tuplet 3/2 { f8 g a }
}
```



Les n-olets peuvent être imbriqués ; par exemple,
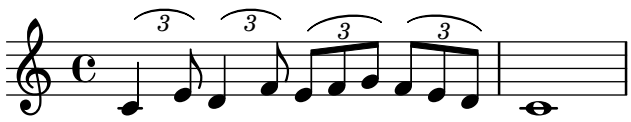
```
\autoBeamOff
c4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4 |
```



Lorsque, dans une imbrication, les n-olets débutent au même instant, il vous faut recourir à la commande `\tweak` – voir Section 5.3.4 [La commande d'affinage \tweak], page 656.

Le crochet de n-olet peut être remplacé par une liaison, à l'instar des éditions anciennes :

```
\relative {
  \tuplet 3/2 4 {
    \override TupletBracket.tuplet-slur = ##t
    c'4 e8 d4 f8
    \override TupletBracket.bracket-visibility = ##t
    e f g f e d
  } c1
}
```



Un crochet sera imprimé en l'absence de ligature couvrant toute sa longueur. Dans certains cas cependant, par exemple en présence d'une liaison comme ci-dessus, il peut être préférable de modifier ce comportement à l'aide de la propriété `bracket-visibility` comme indiqué dans les exemples qui suivent.

De manière générale, aussi bien les objets `TupletBracket` que `TupletNumber` peuvent se masquer comme indiqué dans Section 5.4.7 [Visibilité des objets], page 675 ; on peut néanmoins interférer sur la durée des notes sans imprimer de crochet, comme indiqué au chapitre [Changement d'échelle des durées], page 56.

Commandes prédéfinies

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

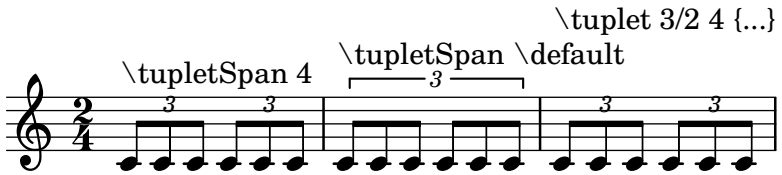
Morceaux choisis

Plusieurs triolets avec une seule commande `\tuplet`

La propriété `tupletSpannerDuration` spécifie la longueur voulue de chaque crochet. Avec elle, vous pouvez faire plusieurs n-olets en ne tapant `\tuplet` qu'une fois, ce qui évite une longue saisie.

Il existe différents moyens de définir `tupletSpannerDuration`. La commande `\tupletSpan` lui affecte une durée arbitraire qui sera réinitialisée dès l'intervention d'une durée à `\default`. Vous pouvez aussi opter pour fournir un argument supplémentaire à la commande `\tuplet`.

```
\relative c' {
  \time 2/4
  \tupletSpan 4
  \tuplet 3/2 { c8^"\\tupletSpan 4" c c c c c }
  \tupletSpan \default
  \tuplet 3/2 { c8^"\\tupletSpan \\default" c c c c c }
  \tuplet 3/2 4 { c8^"\\tuplet 3/2 4 {...}" c c c c c }
}
```

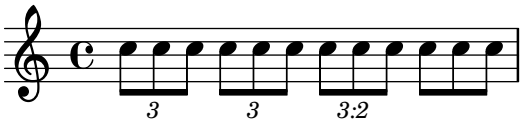


Modifier l'apparence du chiffre de n-olet

LilyPond imprime par défaut le numérateur de la fraction fournie en argument à la commande `\tuplet` du côté du crochet de n-olet.

Il est toutefois possible d'imprimer la fraction entière *num*:*den*, voire de ne rien imprimer du tout.

```
\relative c'' {
  \tuplet 3/2 { c8 c c }
  \tuplet 3/2 { c8 c c }
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```

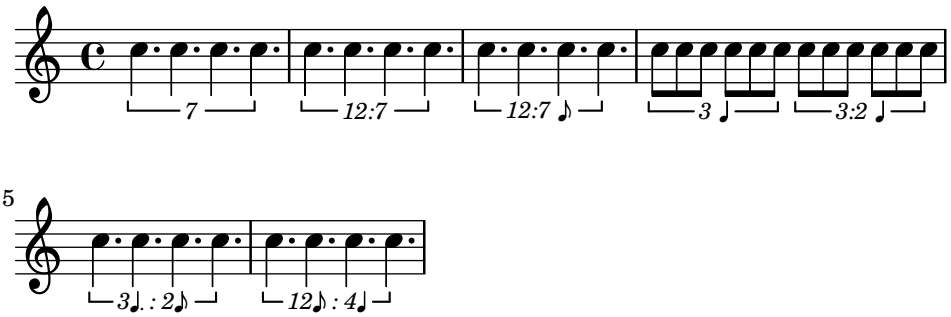


N-olets au chiffrage inhabituel

LilyPond sait aussi gérer des n-olets dont le chiffrage imprimé ne correspond pas exactement à la fraction de mesure à laquelle ils se réfèrent, tout comme ceux auxquels une valeur de note vient en complément du chiffre.

```
\relative c'' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7)
      (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::fraction-with-notes
      (ly:make-duration 2 1) (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-fraction-with-notes 12
```

```
      (ly:make-duration 3 0) 4 (ly:make-duration 2 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
}
```



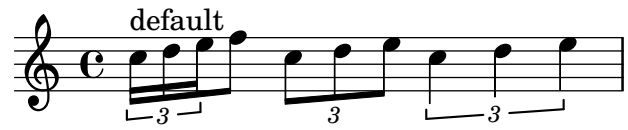
Contrôle de l'impression des crochets de n-olet
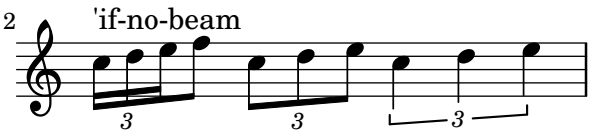
Selon la tradition, les crochets indicateurs de n-olet sont toujours imprimés, sauf dans le cas où ils seraient de la même longueur qu'une ligature.
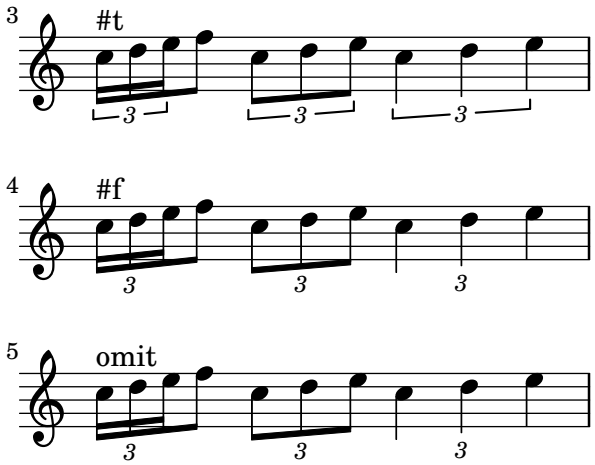
LilyPond permet, au travers de la propriété `bracket-visibility`, de contôler précisément leur affichage : déterminée à `#t`, ils seront toujours imprimés ; `#f` permet de ne jamais les imprimer – donc omettre l'objet `TupletBracket` –, et `#'if-no-beam` les imprimera en l'absence de ligature (comportement par défaut).

```
music = \relative c'' {
  \tuplet 3/2 { c16[ d e } f8]
  \tuplet 3/2 { c8 d e }
  \tuplet 3/2 { c4 d e }
}

\new Voice {
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music s4^"#f" >>
    \omit TupletBracket
    << \music s4^"omit" >>
  }
}
```



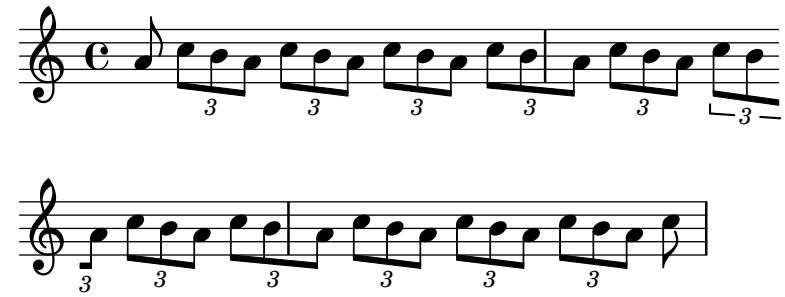




Saut de ligne au milieu d'un n-olet avec ligature

Cet exemple peu académique démontre comment il est possible d'insérer un saut de ligne dans un n-olet portant une ligature. Ces ligatures doivent toutefois être explicites.

```
\layout {
  \context {
    \Voice
    % Permit line breaks within tuplets
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks
    \override Beam.breakable = ##t
  }
}
\relative c'' {
  a8
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  % Insert a manual line break within a tuplet
  \tuplet 3/2 { c[ b \bar "" \break a] }
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  c8
}
```



Voir aussi

Glossaire musicologique : Section "triolet" dans *Glossaire*, Section "n-olet" dans *Glossaire*, Section "polymétrie" dans *Glossaire*.

Manuel d'initiation : Section "Méthodes de retouche" dans *Manuel d'initiation*.

Manuel de notation : [Changement d'échelle des durées], page 56, Section 5.4.2 [Direction et positionnement], page 667, [Gestion du temps], page 125, Section 5.3.4 [La commande d'affinage `\tweak`], page 656, [Notation polymétrique], page 80, Section 5.4.7 [Visibilité des objets], page 675.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section “TupletBracket” dans *Référence des propriétés internes*, Section “TupletNumber” dans *Référence des propriétés internes*, Section “TimeScaledMusic” dans *Référence des propriétés internes*.
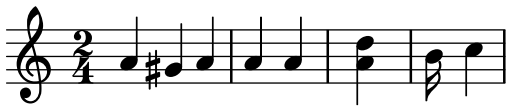
Changement d’échelle des durées

La durée des notes, silences ou accords peut se modifier en lui adjoignant une fraction *N/D*, donnant « *N/D » – ou « *N » si *D=1*. Ce facteur peut aussi s’exprimer à l’aide d’une expression Scheme évaluant à un nombre ou un temps musical comme `*#(ly:music-length musique)`. Cette solution est pratique pour échelonner à une durée de ‘`1`’ et laisser une note ou un silence multimesure s’étirer sur une longueur dérivée d’une variable musicale.

L’ajout de ce facteur ne modifiera en rien l’apparence des notes ou silences produits, mais affectera le positionnement de l’objet dans la mesure, ainsi que le rendu MIDI. Cette fraction peut elle-même être multipliée, ce qui donne quelque chose du style `*M*N/D`. Ce facteur d’échelonnement est partie intégrante de la durée : en l’absence de durée explicite à la note suivante, cette durée échelonnée est considérée comme valeur par défaut.

Dans l’exemple suivant, les trois premières notes prennent exactement deux temps, mais aucun triolet n’est imprimé.

```
\relative {
  \time 2/4
  % Alter durations to triplets
  a'4*2/3 gis a
  % Normal durations
  a4 a
  % Double the duration of chord
  <a d>4*2
  % Duration of quarter, appears like sixteenth
  b16*4 c4
}
```
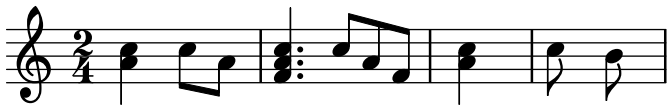


La durée d’un silence invisible ou saut de notes (*skip*) peut elle aussi être affectée d’un multiplicateur. Cette technique permet tout simplement de sauter plusieurs mesures, comme par exemple un `s1*23`.

Il est tout à fait possible d’échelonner des fragments musicaux plus ou moins longs à l’aide d’une simple fraction, comme si chaque note, accord ou silence était affecté de ce même quotient. L’apparence de cette musique ne sera en rien modifiée ; seule la durée des notes est multipliée en interne par le facteur d’échelle donné – généralement *numérateur/dénominateur*. Voici un exemple illustrant la manière de comprimer ou étirer de la musique :

```
\relative {
  \time 2/4
  % Durée normale
  <c'' a>4 c8 a
  % Musique échelonnée à 2/3
  \scaleDurations 2/3 {
    <c a f>4. c8 a f
  }
  % Musique échelonnée par 2
  \scaleDurations 2 {
```

```
    <c' a>4 c8 b
  }
}
```



Cette technique est tout à fait appropriée à la notation polymétrique – voir [Notation polymétrique], page 80.

Voir aussi

Manuel de notation : [N-olets], page 51, [Notation polymétrique], page 80, [Silences invisibles], page 63.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Problèmes connus et avertissements

Le calcul de la position au sein d'une mesure doit prendre en considération tous les facteurs d'échelonnement appliqués aux notes de cette mesure ainsi que tous les reliquats des mesures précédentes. Ce calcul utilise des nombres rationnels. Dès lors qu'un calcul recontrera un numérateur ou dénominateur intermédiaire d'une valeur supérieure à 2^30, LilyPond s'arrêtera à ce point précis sans pour autant signaler d'erreur.

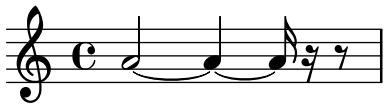
Liaisons de prolongation

Une liaison de tenue (ou de prolongation) relie deux notes adjacentes de même hauteur. Dans les faits, elle prolonge la durée d'une note.

> **Note :** Une liaison de tenue ne doit pas être confondue avec une liaison d'**articulation** ou de **phrasé**. Une liaison de tenue est un moyen parmi d'autres pour prolonger la durée d'une note, tout comme les points.

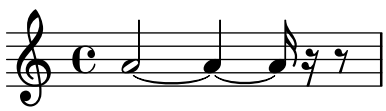
Une liaison de tenue s'indique au moyen d'un tilde (~) qui vient s'adjoindre à la première note de chacune des paires de notes à lier. Ceci indique que la note en question sera liée à la suivante, qui doit être de la même hauteur.

```
{ a'2~ 4~ 16 r r8 }
```



Les liaisons de tenue peuvent interpréter la « dernière hauteur explicite » d'une succession de durées :
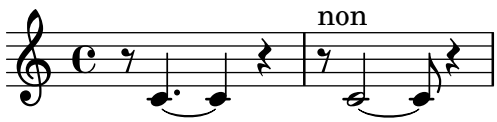
```
{ a'2~ 4~ 16 r r8 }
```



Les liaisons de tenue sont utilisées soit lorsque la note dépasse de la mesure, soit quand les points ne suffisent pas à donner la bonne durée. Lorsque l'on utilise ces liaisons, les valeurs rythmiques les plus longues doivent s'aligner sur les subdivisions de la mesure, comme ici :

```
\relative {
```

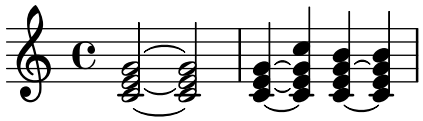
```
  r8 c'4.~ 4 r4 |
  r8^"non" c2~ 8 r4
}
```



Lorsque l'on doit lier de nombreuses notes sur plusieurs mesures, il devient plus facile d'avoir recours à la division automatique des notes – voir [Découpage automatique des notes], page 83. Ce procédé divise automatiquement les notes trop longues, et les lie par-delà les barres de mesure.

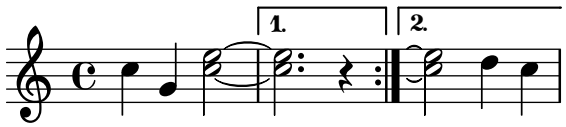
Quand une liaison de tenue se trouve entre deux accords, toutes les notes de même hauteur entre ces deux accords sont reliées. S'il n'y en a aucune, aucune liaison n'est créée. Il est également possible de lier partiellement deux accords, en mettant les liaisons à l'intérieur des accords.

```
\relative c' {
  <c e g>2~ 2 |
  <c e g>4~ <c e g c>
    <c~ e g~ b> <c e g b> |
}
```
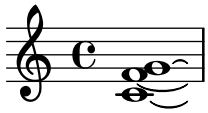


Lorsqu'une mesure d'alternative après une reprise commence sur une note liée, la liaison doit être répétée, comme ici :

```
\relative {
  \repeat volta 2 { c'' g <c e>2~ }
  \alternative {
    % La note tenue est liée normalement
    \volta 1 { <c e>2. r4 }
    % la note qui suit est pourvue d'une liaison de reprise
    \volta 2 { <c e>2\repeatTie d4 c }
  }
}
```



Les liaisons « Laissez vibrer » (*L.v.*) sont utilisées pour le piano, la harpe et certains instruments de percussion. Elles indiquent à l'instrumentiste de laisser sonner la note ou l'accord au lieu de l'étouffer. Elles s'indiquent de la manière suivante :
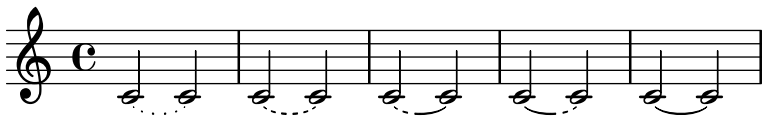
```
<c' f' g'>1\laissezVibrer
```



Le positionnement vertical des liaisons de prolongation peut être affiné – voir à ce sujet les « commandes prédéfinies » et, pour de plus amples détails, Section 5.4.2 [Direction et positionnement], page 667.
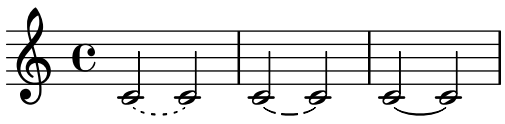
Les liaisons de prolongation peuvent s'imprimer sous la forme de ligne continue, discontinue ou pointillée.

```
\relative c' {
  \tieDotted
  c2~ 2
  \tieDashed
  c2~ 2
  \tieHalfDashed
  c2~ 2
  \tieHalfSolid
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Il est même possible d'en personnaliser l'allure :

```
\relative c' {
  \tieDashPattern #0.3 #0.75
  c2~ 2
  \tieDashPattern #0.7 #1.5
  c2~ 2
  \tieSolid
  c2~ 2
}
```
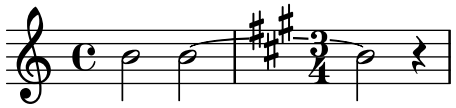


Que ce soit pour une tenue ou un phrasé, le motif d'une ligne discontinue formant une liaison se définit de la même manière. Pour de plus amples détails à ce sujet, reportez vous au chapitre [Liaisons d'articulation], page 139.

Dans le cas où une liaison est recouverte par d'autres éléments de la portée, une adaptation des propriétés *whiteout* et *layer* permet d'obtenir une meilleure lisibilité.

```
\relative {
  \override Tie.layer = #-2
  \override Staff.TimeSignature.layer = #-1
  \override Staff.KeySignature.layer = #-1
  \override Staff.TimeSignature.whiteout = ##t
  \override Staff.KeySignature.whiteout = ##t
  b'2 b~
  \time 3/4
  \key a \major
```

```
  b r4
}
```



Commandes prédéfinies

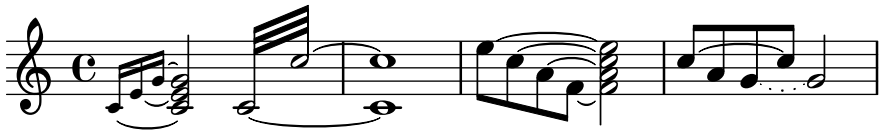
`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

Morceaux choisis

Liaison de tenue et arpège

Les liaisons de tenue servent parfois à rendre un accord arpégé. Dans ce cas, les notes liées ne sont pas toutes consécutives. Il faut alors assigner à la propriété `tieWaitForNote` la valeur `#t` (*true* pour « vrai »). Cette même méthode peut servir, par exemple, à lier un trémolo à un accord.

```
\relative c' {
  \set tieWaitForNote = ##t
  \grace { c16[ ~ e ~ g] ~ } <c, e g>2
  \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
  e8 ~ c ~ a ~ f ~ <e' c a f>2
  \tieUp
  c8 ~ a
  \tieDown
  \tieDotted
  g8 ~ c g2
}
```
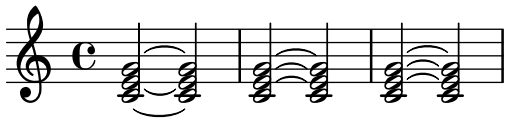


Dessin à main levée de liaisons de tenue

Il est possible de graver manuellement les liaisons de tenue, en modifiant la propriété `tie-configuration`. Pour chaque paire, le premier nombre indique la distance à la portée, en espaces de portée, et le second la direction (1 pour haut, −1 pour bas).

Notez bien que LilyPond fait la distinction, au niveau du premier nombre, entre valeur exacte et valeur inexacte. Dans le cas d'une valeur exacte – autrement dit un entier ou une fraction comme `(/ 4 5)` – celle-ci servira de position verticale brute, ensuite afinée par LilyPond de sorte à éviter les lignes de la portée. Dans le cas d'une valeur inexacte, tel un nombre à virgule flottante, c'est elle qui servira à positionner verticalement, sans ajustement.

```
\relative c' {
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0 . 1) (-2 . 1) (-4 . 1))
  <c e g>2~ <c e g>
```

```
}
```



Voir aussi

Glossaire musicologique : Section “liaison de tenue” dans *Glossaire*, Section “laissez vibrer” dans *Glossaire*.

Manuel de notation : [Découpage automatique des notes], page 83, [Liaisons d’articulation], page 139.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “LaissezVibrerTie” dans *Référence des propriétés internes*, Section “LaissezVibrerTieColumn” dans *Référence des propriétés internes*, Section “TieColumn” dans *Référence des propriétés internes*, Section “Tie” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un changement de portée, lorsqu’une liaison de tenue est active, ne peut produire une liaison oblique.

Un changement de clef ou d’octave pendant une liaison de tenue produit un résultat indéfini. Dans ces cas là, il est préférable d’utiliser un *legato*.
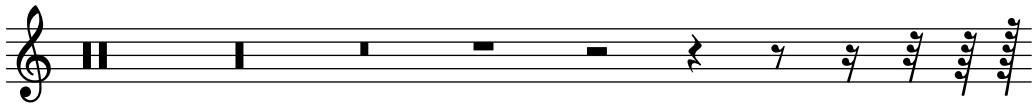
1.2.2 Écriture des silences

On saisit les silences dans une expression musicale tout comme les notes.

Silences

Les silences sont écrits comme des notes avec le nom de note `r` – premier caractère du mot *rest*. Les durées supérieures à la pause s’indiquent à l’aide de commandes prédéfinies :
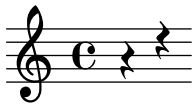
```
\new Staff {
  % These two lines are just to prettify this example
  \time 16/1
  \omit Staff.TimeSignature
  % Print a maxima rest, equal to four breves
  r\maxima
  % Print a longa rest, equal to two breves
  r\longa
  % Print a breve rest
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Les pauses d’une mesure complète, qui sont placées au centre de la mesure, doivent être entrées comme des mesures de silence. Elles peuvent être utilisées pour une seule mesure comme pour plusieurs, et leur utilisation est expliquée à la rubrique [Silences valant une mesure], page 65.

Pour spécifier explicitement la position verticale d'un silence, écrivez une note suivie de `\rest`. Un silence de même durée sera placé à la position où serait imprimée la note. Cela rend plus facile la mise en place de musique polyphonique, puisque le formateur automatique de collision des silences laissera ces silences tranquilles.

```
\relative { a'4\rest d4\rest }
```



Morceaux choisis

Styles de silences

Les silences peuvent être gravés selon différents styles.

```
\new Staff \relative c {
  \omit Score.TimeSignature
  \cadenzaOn

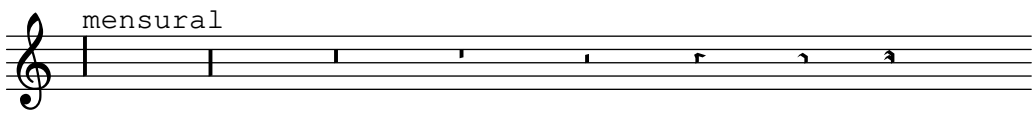
  \override Staff.Rest.style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

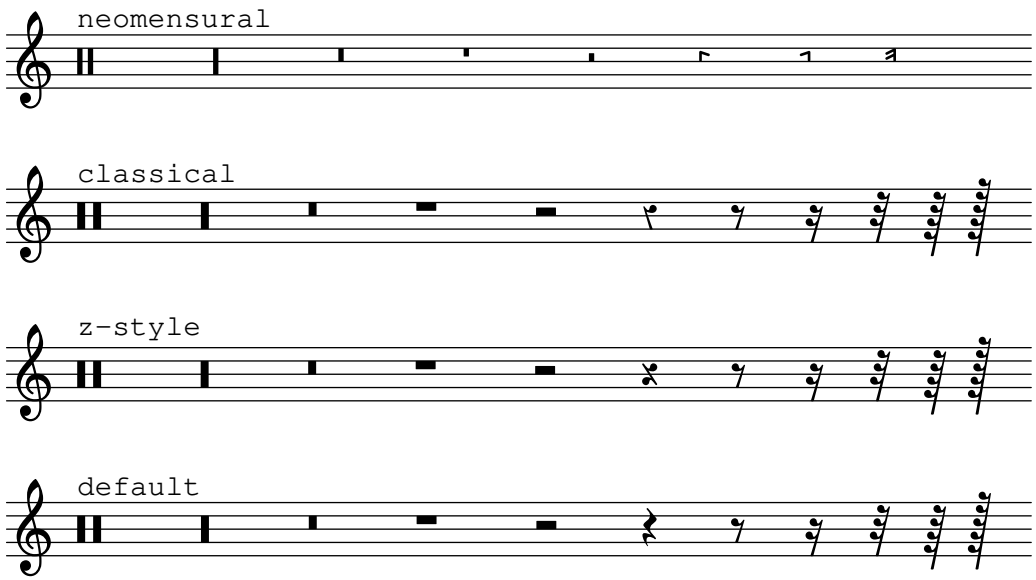
  \override Staff.Rest.style = #'neomensural
  r\maxima^\markup \typewriter { neomensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

  \override Staff.Rest.style = #'classical
  r\maxima^\markup \typewriter { classical }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
  \bar ""
  \break

  \override Staff.Rest.style = #'z
  r\maxima^\markup \typewriter { z-style }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
  \bar ""
  \break

  \override Staff.Rest.style = #'default
  r\maxima^\markup \typewriter { default }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
}
```





Voir aussi

Glossaire musicologique : Section “breve” dans *Glossaire*, Section “longa” dans *Glossaire*, Section “maxima” dans *Glossaire*.

Manuel de notation : [Silences valant une mesure], page 65.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Rest” dans *Référence des propriétés internes*.
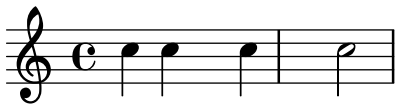
Problèmes connus et avertissements

Il n’existe pas à proprement parler de limite (inférieure ou supérieure) en terme de durée d’un silence. Cependant, le nombre de glyphes disponibles ne couvre que les silences allant du 1 024^{e} de pause (256^{e} de soupir) à la maxime (équivalant à huit pauses).

Silences invisibles

Un silence invisible – que l’on pourrait appeler un « saut » – peut être entré comme une note avec le nom de note `s` ou avec `\skip` *`durée`* :
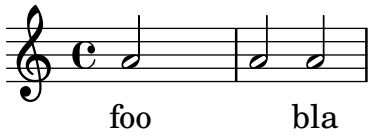
```
\relative c'' {
  c4 c s c |
  s2 c |
}
```



La syntaxe `s` est seulement disponible pour les modes d’entrée de notes et d’accords. Dans les autres situations, pour l’entrée de paroles par exemple, vous devrez utiliser la commande `\skip`, qui requiert une durée explicite ; cette durée ne sera pas prise en considération dès lors que les paroles suivent le rythme des notes de la mélodie à laquelle vous les aurez associées à l’aide des commandes `\addlyrics` ou `\lyricsto`.
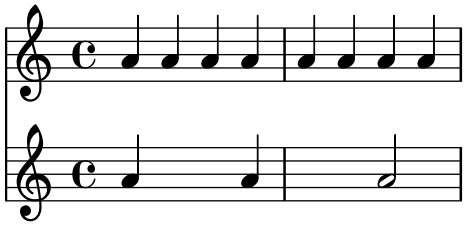
```
<<
  {
    a'2 \skip2 a'2 a'2
  }
  \new Lyrics {
    \lyricmode {
```

```
      foo2 \skip 1 bla2
    }
  }
>>
```



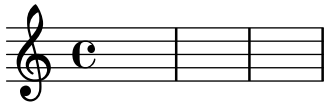
Gardez à l'esprit que `\skip` est une commande, et de ce fait n'affectera en rien la durée des notes qui suivent, contrairement à un `s`.

```
<<
  {
    \repeat unfold 8 { a'4 }
  }
  {
    a'4 \skip 2 a' |
    s2 a'
  }
>>
```



La commande de saut génère simplement une case musicale vide. Le code de saut `s` crée tout de même les contextes `Staff` et `Voice` lorsque nécessaire, à l'instar des notes ou des silences :

```
{ s1 s s }
```



Un `\skip` ne fait que sauter du temps musical ; il ne produit rien du tout, pas même un symbole transparent.

```
% This is valid input, but does nothing
{ \skip 1 \skip1 \skip 1 }
```

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : [Dictée à trous], page 241, Section 5.4.7 [Visibilité des objets], page 675.

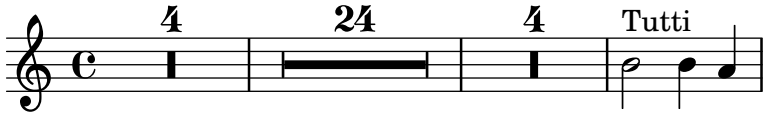
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SkipMusic" dans *Référence des propriétés internes*.

Silences valant une mesure

Un silence valant une ou plusieurs mesures entières s'entre avec un `R` majuscule. Sa durée s'indique comme pour n'importe quelle note, y compris un éventuel multiplicateur comme indiqué dans [Changement d'échelle des durées], page 56.

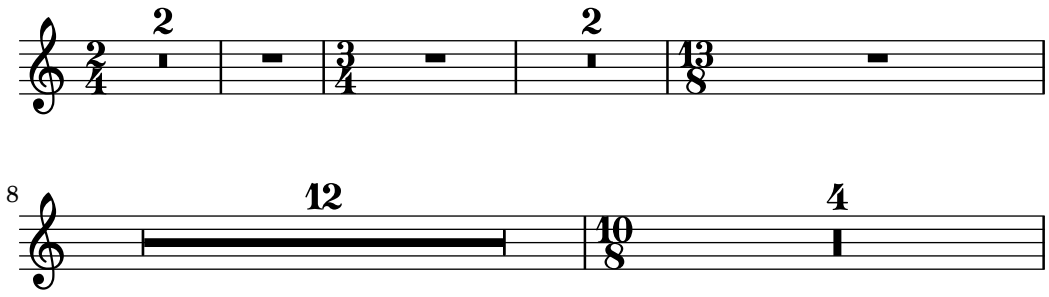
```
% Rest measures contracted to single measure
\compressMMRests {
  R1*4
  R1*24
  R1*4
  b'2^"Tutti" b'4 a'4
}
```



Cet exmple illustre aussi la manière de condenser une succession de mesures vides, comme indiqué dans [Compression de mesures vides], page 230.
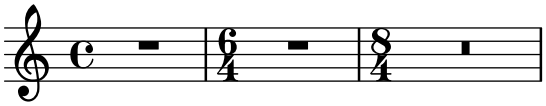
Ceci ne peut être utile que pour une mesure complètement vide. Sa durée doit donc correspondre à la longueur de la mesure telle que définie par la métrique. C'est la raison pour laquelle on utilisera aussi des points d'augmentation ou des fractions :

```
\compressMMRests {
  \time 2/4
  R1 | R2 |
  \time 3/4
  R2. | R2.*2 |
  \time 13/8
  R1*13/8 | R1*13/8*12 |
  \time 10/8
  R4*5*4 |
}
```



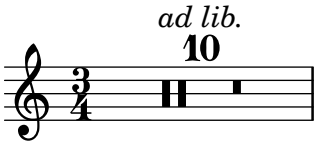
Un `R` qui s'étend sur une seule mesure s'imprime tantôt comme une pause, tantôt comme une brève – ou « bâton de pause » – qui sera centrée sur la mesure quelle qu'en soit la métrique :

```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```



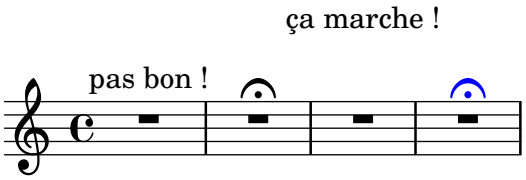
Vous pouvez aussi ajouter du texte à un silence multimesure en utilisant la syntaxe *note*-`markup` (cf. Section 1.8.2 [Mise en forme du texte], page 263).

```
\compressMMRests {
  \time 3/4
  R2.*10^\markup { \italic "ad lib." }
}
```



> **Note :** C'est `MultiMeasureRestText` qui créera le texte, et `MultiMeasureRestScript` pour les articulations, non `TextScript` ou `Script`. Les commandes de dérogation ou de redéfinition doivent s'adresser à l'objet concerné, comme vous pouvez le constater dans l'exemple suivant.

```
% Ceci échouera : il y a erreur quant à l'objet spécifié
\override TextScript.padding = #5
\override Script.color = #blue
R1^"pas bon !"
R1\fermata
% Formulation correcte, qui fonctionnera
\override MultiMeasureRestText.padding = #5
\override MultiMeasureRestScript.color = #blue
R1^"ça marche !"
R1\fermata
```



Un silence multimesure placé directement après une commande `\partial` risque fort de perturber le vérificateur de limites et numéros de mesure.

Commandes prédéfinies

`\textLengthOn`, `\textLengthOff`, `\compressMMRests`.

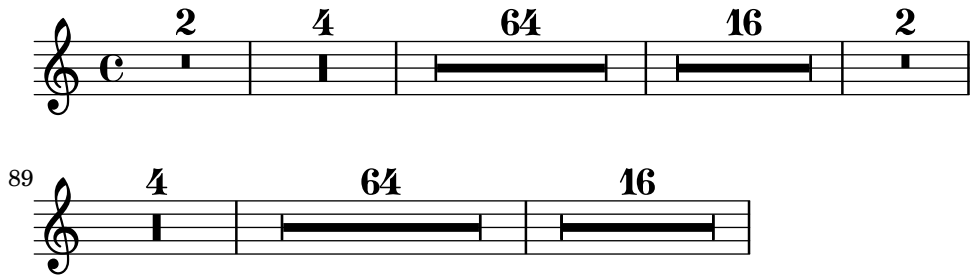
Morceaux choisis

Contrôle de la taille d'un silence multimesure

Les silences multimesure ont une largeur relative à leur durée totale, contrôlée par `MultiMeasureRest.space-increment`. Sa valeur par défaut est fixée à `2.0`.

```
\relative c' {
  \compressEmptyMeasures
  R1*2 R1*4 R1*64 R1*16
  \override Staff.MultiMeasureRest.space-increment = 2.5
```

```
  R1*2 R1*4 R1*64 R1*16
}
```



Positionnement des silences multimesures

Si l'on peut positionner verticalement un silence simple en le rattachant à une note, il n'en va pas de même pour un silence multimesure. Néanmoins, et uniquement dans le cadre de musique polyphonique, les silences multimesures sont positionnés différemment selon qu'ils appartiennent à une voix au numéro pair ou impair. Le positionnement des silences multimesures peut se contrôler ainsi :

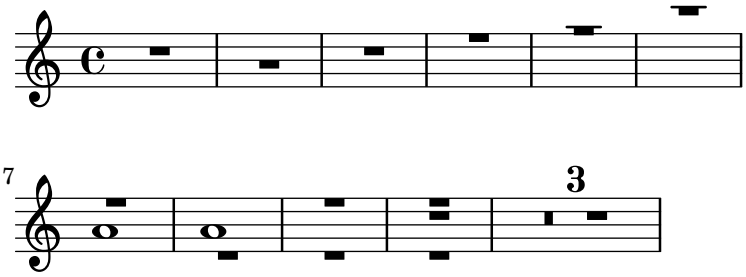
```
\relative c'' {
  % Multi-measure rests by default are set under the fourth line
  R1
  % They can be moved using an override
  \override MultiMeasureRest.staff-position = #-2
  R1
  \override MultiMeasureRest.staff-position = #0
  R1
  \override MultiMeasureRest.staff-position = #2
  R1
  \override MultiMeasureRest.staff-position = #3
  R1
  \override MultiMeasureRest.staff-position = #6
  R1
  \revert MultiMeasureRest.staff-position
  \break

  % In two Voices, odd-numbered voices are under the top line
  << { R1 } \\ { a1 } >>
  % Even-numbered voices are under the bottom line
  << { a1 } \\ { R1 } >>
  % Multi-measure rests in both voices remain separate
  << { R1 } \\ { R1 } >>

  % Separating multi-measure rests in more than two voices
  % requires an override
  << { R1 } \\ { R1 } \\
     \once \override MultiMeasureRest.staff-position = #0
     { R1 }
  >>

  % Using compressed bars in multiple voices requires another override
  % in all voices to avoid multiple instances being printed
  \compressMMRests
```

```
  <<
   \revert MultiMeasureRest.direction
    { R1*3 }
    \\
   \revert MultiMeasureRest.direction
    { R1*3 }
  >>
}
```

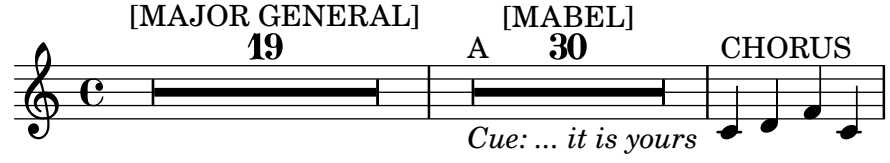


Ajout de texte à un silence multimesure

Lorsque du texte est attaché à un silence multimesure, il sera centré dans la mesure, au-dessus ou en dessous de la portée. Afin d'étirer la mesure dans le cas où ce texte est relativement long, il suffit d'insérer un accord vide auquel on attache le texte en question, avant le silence multimesure.

Le texte attaché à un silence invisible sera aligné sur la gauche de là où serait positionnée la note dans la mesure. Cependant, si la taille de la mesure est déterminée par la longueur du texte, il apparaîtra comme centré.

```
\relative c' {
  \compressMMRests {
    \textLengthOn
    <>^\markup { [MAJOR GENERAL] }
    R1*19
    <>_\markup { \italic { Cue: ... it is yours } }
    <>^\markup { A }
    R1*30^\markup { [MABEL] }
    \textLengthOff
    c4^\markup { CHORUS } d f c
  }
}
```



Voir aussi

Glossaire musicologique : Section "silence multimesures" dans *Glossaire*.

Manuel de notation : [Changement d'échelle des durées], page 56, [Commentaires textuels], page 256, [Compression de mesures vides], page 230, [Durées], page 48, Section 1.8.2 [Mise en forme du texte], page 263, Section 1.8 [Texte], page 253.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section “MultiMeasureRest” dans *Référence des propriétés internes*. Section “MultiMeasureRestNumber” dans *Référence des propriétés internes*, Section “MultiMeasureRestScript” dans *Référence des propriétés internes*, Section “MultiMeasureRestText” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Vous ne pouvez pas utiliser de doigtés (par ex. `R1*10-4`) pour positionner des nombres au dessus d'un silence multimesure, le numéro de doigt (4) risquant de chevaucher le nombre de mesures à compter (10).

Condenser plusieurs silences en un unique silence multimesure ne peut être automatisé.
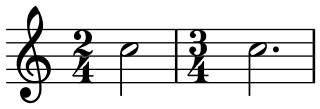
Les silences multimesures peuvent générer des collisions avec d'autres silences.

1.2.3 Gravure du rythme

Métrique

Le chiffre de mesure indique le mètre d'une pièce : une alternance régulière de temps forts et de temps faibles. Il est indiqué par une fraction au début de la portée :

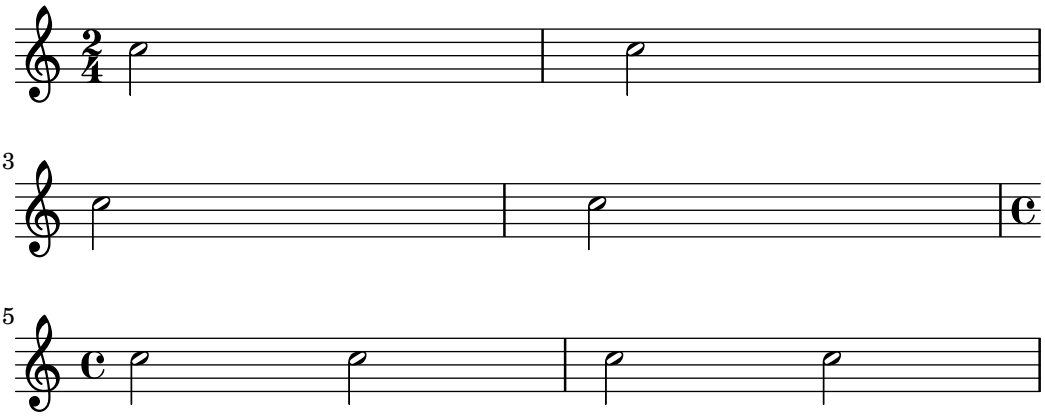
```
\time 2/4 c''2
\time 3/4 c''2.
```



Les changements de métrique en cours de mesure sont abordés dans [Levées], page 77.
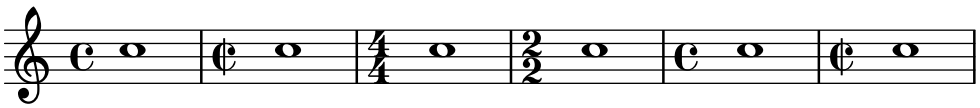
La métrique est imprimée en début de morceau, et à chaque fois qu'elle est modifiée. Si cette modification intervient au niveau d'un saut de ligne, une métrique « de précaution » sera imprimée avant de passer à la ligne suivante. Ce comportement par défaut peut être modifié, comme indiqué au chapitre Section 5.4.7 [Visibilité des objets], page 675.

```
\relative c'' {
  \time 2/4
  c2 c
  \break
  c c
  \break
  \time 4/4
  c c c c
}
```



Le symbole de métrique utilisé pour les mesures à 2/2 et 4/4 peut être changé pour un style numérique :

```
\relative c'' {
  % Default style
  \time 4/4 c1
  \time 2/2 c1
  % Change to numeric style
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  % Revert to default style
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}
```
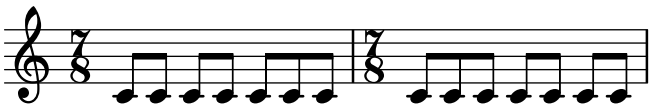


Les métriques anciennes font l'objet d'un Section "chapitre particulier" dans *Manuel de notation*.

En plus de déterminer la métrique qui sera imprimée, la commande `\time` réglera aussi les valeurs par défaut des propriétés `baseMoment`, `beatStructure` et `beamExceptions` correspondant à la métrique. Les valeurs prédéterminées par défaut de ces différentes propriétés sont inscrites dans le fichier `scm/time-signature-settings.scm`.

La valeur par défaut de `beatStructure` peut se voir aménagée dès la commande `\time` à l'aide d'un premier argument :
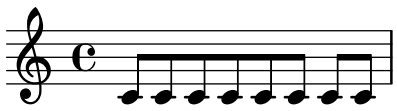
```
\score {
  \new Staff {
    \relative {
      \time 2,2,3 7/8
      \repeat unfold 7 { c'8 } |
      \time 3,2,2 7/8
      \repeat unfold 7 { c8 } |
    }
  }
}
```



Les valeurs par défaut de toutes ces variables associées à la métrique, y compris `baseMoment` et `beamExceptions`, peuvent se définir en même temps. Ces valeurs peuvent se régler indépendamment pour différentes métriques. Les valeurs adaptées ne seront effectives qu'à partir du moment où interviendra une commande `\time` de la valeur de métrique correspondante :

```
\score {
  \relative c' {
    \overrideTimeSignatureSettings
```

```
      4/4        % timeSignatureFraction
      1/4        % baseMomentFraction
      3,1        % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```
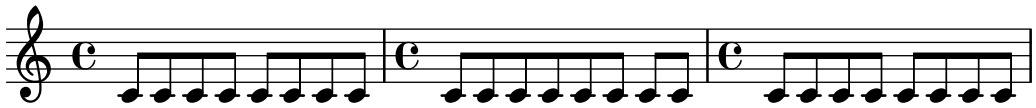


`\overrideTimeSignatureSettings` prend quatre arguments :

1. *`timeSignatureFraction`*, une fraction indiquant la métrique pour laquelle ces valeurs doivent s'appliquer,
2. *`baseMomentFraction`*, une fraction comprenant les numérateur et dénominateur de la fraction indiquant la base de la pulsation,
3. *`beatStructure`*, une liste Scheme indiquant la structure de cette pulsation, en unité de base,
4. *`beamExceptions`*, une liste associative des règles de ligature pour cette métrique, en dehors de celles basées sur le temps comme indiqué à la rubrique [Définition des règles de ligature automatique], page 90.

Vous pouvez revenir à tout moment aux réglages prédéterminés d'une métrique :

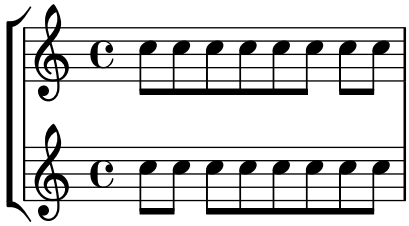
```
\score {
  \relative {
    \repeat unfold 8 { c'8 } |
    \overrideTimeSignatureSettings
      4/4        % timeSignatureFraction
      1/4        % baseMomentFraction
      3,1        % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
    \revertTimeSignatureSettings 4/4
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```



Le fait de déplacer du contexte `Score` au contexte `Staff` à la fois le `Timing_translator` et le `Default_bar_line_engraver` permet d'obtenir des réglages particuliers pour les différentes portées d'un regroupement :

```
\score {
  \new StaffGroup <<
    \new Staff {
```

```
        \overrideTimeSignatureSettings
          4/4        % timeSignatureFraction
          1/4        % baseMomentFraction
          3,1        % beatStructure
          #'()       % beamExceptions
        \time 4/4
        \repeat unfold 8 {c''8}
      }
      \new Staff {
        \overrideTimeSignatureSettings
          4/4        % timeSignatureFraction
          1/4        % baseMomentFraction
          1,3        % beatStructure
          #'()       % beamExceptions
        \time 4/4
        \repeat unfold 8 {c''8}
      }
    >>
    \layout {
      \context {
        \Score
        \remove "Timing_translator"
        \remove "Default_bar_line_engraver"
      }
      \context {
        \Staff
        \consists "Timing_translator"
        \consists "Default_bar_line_engraver"
      }
    }
}
```



Une autre méthode de modification de ces variables liées à la métrique, et qui évite sa réimpression au moment du changement, est indiquée à la rubrique [Définition des règles de ligature automatique], page 90.

Commandes prédéfinies

`\numericTimeSignature`, `\defaultTimeSignature`.
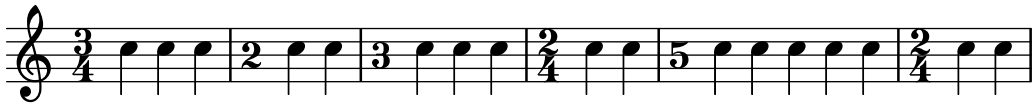
Morceaux choisis

Affichage seulement du numérateur d'une métrique (au lieu d'une fraction)

La métrique est parfois indiquée non pas par une fraction (par ex. 7/4) mais simplement par son numérateur (7 dans ce cas). L'instruction `\override Staff.TimeSignature.style = #'single-digit` permet de déroger au style par défaut de manière permanente – un `\revert`

`Staff.TimeSignature.style` annulera ces modifications. Lorsque cette métrique sous la forme d'un seul chiffre ne se présente qu'une seule fois, il suffit de faire précéder l'instruction `\override` d'un simple `\once`.

```
\relative c'' {
  \time 3/4
  c4 c c
  % Change the style permanently
  \override Staff.TimeSignature.style = #'single-digit
  \time 2/4
  c4 c
  \time 3/4
  c4 c c
  % Revert to default style:
  \revert Staff.TimeSignature.style
  \time 2/4
  c4 c
  % single-digit style only for the next time signature
  \once \override Staff.TimeSignature.style = #'single-digit
  \time 5/4
  c4 c c c c
  \time 2/4
  c4 c
}
```



Voir aussi

Glossaire musicologique : Section "métrique" dans *Glossaire*.

Manuel de notation : [Définition des règles de ligature automatique], page 90, [Gestion du temps], page 125, [Métriques anciennes], page 474.

Installed Files: `scm/time-signature-settings.scm`.
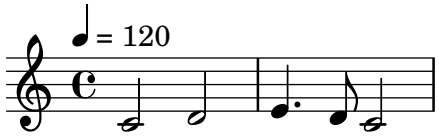
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "TimeSignature" dans *Référence des propriétés internes*, Section "Timing_translator" dans *Référence des propriétés internes*.

Indication métronomique
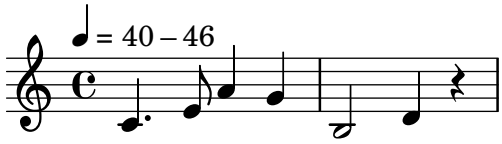
Une indication métronomique s'insère tout simplement comme ceci :

```
\relative {
  \tempo 4 = 120
  c'2 d
  e4. d8 c2
}
```
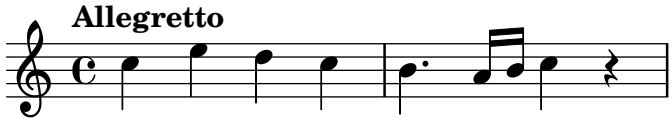


Lorsque le réglage précis du métronome est laissé à l'appréciation de l'exécutant, vous pouvez cependant lui fournir une plage :

```
\relative {
  \tempo 4 = 40 - 46
  c'4. e8 a4 g
  b,2 d4 r
}
```
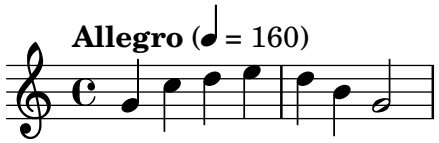


Vous pouvez préférer une indication textuelle :

```
\relative {
  \tempo "Allegretto"
    c''4 e d c
    b4. a16 b c4 r4
}
```
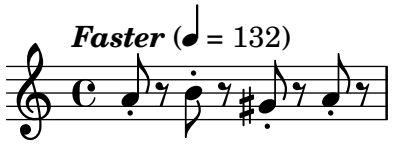


Lorsque vous combinez des indications métronomiques sous forme textuelle et numérique, l'indication numérique sera placée entre parenthèses :

```
\relative {
  \tempo "Allegro" 4 = 160
  g'4 c d e
  d4 b g2
}
```



En matière d'indication textuelle, vous pouvez utiliser n'importe quel objet de type *markup*, comme ici :
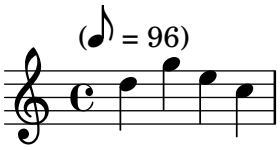
```
\relative {
  \tempo \markup { \italic Faster } 4 = 132
  a'8-. r8 b-. r gis-. r a-. r
}
```



Mentionner une indication textuelle vide vous permet de mettre entre parenthèses l'indication numérique :

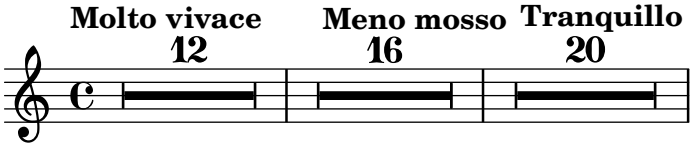
```
\relative {
```

```
  \tempo "" 8 = 96
  d''4 g e c
}
```



Dans le cas d'une partie où l'instrumentiste a de longs moments de silence, les indications de tempo peuvent être fort rapprochées l'une de l'autre. L'instruction `\markLengthOn` permet de préserver suffisamment d'espace horizontal de telle sorte que ces indications ne se chevauchent ; l'instruction `\markLengthOff` restaure le comportement par défaut qui ignore les indications de tempo dans les calculs d'espacement horizontal.

```
\compressMMRests {
  \markLengthOn
  \tempo "Molto vivace"
  R1*12
  \tempo "Meno mosso"
  R1*16
  \markLengthOff
  \tempo "Tranquillo"
  R1*20
}
```



Morceaux choisis

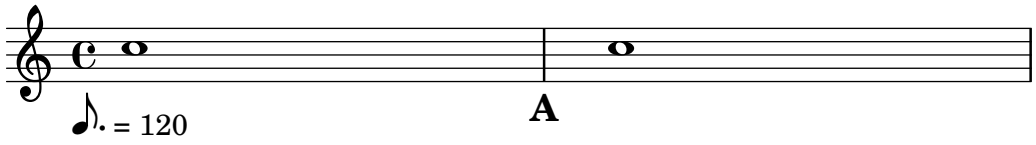
Impression du métronome et des repères sous la portée

Les indications de tempo et les marques de repère s'impriment par défaut au-dessus de la portée. Le fait de régler en conséquence la propriété `direction` des objets `MetronomeMark` et `RehearsalMark` les placera au-dessous de la portée.

```
\layout {
  indent = 0
  ragged-right = ##f
}

{
  % Metronome marks below the staff
  \override Score.MetronomeMark.direction = #DOWN
  \tempo 8. = 120
  c''1
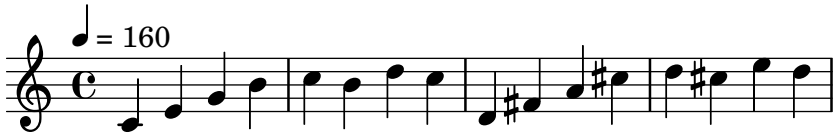
  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
}
```



Changement de tempo sans indication sur la partition

Vous pouvez indiquer un changement de tempo pour le fichier MIDI sans pour autant l'imprimer. Il suffit alors de le rendre invisible aux musiciens.

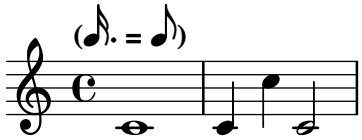
```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```



Création d'une indication métronomique sous forme d'étiquette

Vous pouvez créer des indications de tempo sous la forme d'étiquettes textuelles – des objets *markup* –, notamment des équivalences. Cependant, elles n'apparaîtront pas dans le fichier MIDI.

```
\relative c' {
  \tempo \markup {
    \concat {
      (
      \smaller \general-align #Y #DOWN \note {16.} #1
      " = "
      \smaller \general-align #Y #DOWN \note {8} #1
      )
    }
  }
  c1
  c4 c' c,2
}
```



Pour de plus amples détails, veuillez consulter Section 1.8.2 [Mise en forme du texte], page 263.

Voir aussi

Glossaire musicologique : Section “métronome” dans *Glossaire*, Section “indication métronomique” dans *Glossaire*, Section “indication du tempo” dans *Glossaire*, Section “marque de métronome” dans *Glossaire*.

Manuel de notation : Section 3.5 [Génération de fichiers MIDI], page 556, Section 1.8.2 [Mise en forme du texte], page 263.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “MetronomeMark” dans *Référence des propriétés internes*.
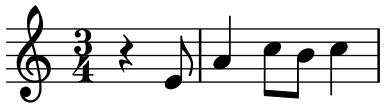
Levées

Les mesures incomplètes, telles que les anacrouses ou levées, doivent être entrées avec la commande `\partial`. La syntaxe de cette commande est

```
\partial durée
```
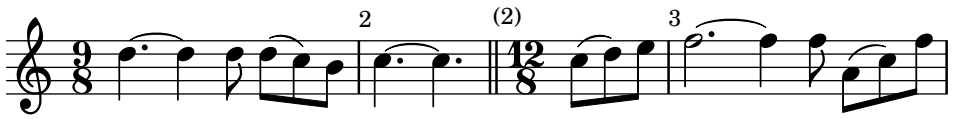
Lorsque `\partial` est utilisé en début de pièce, *durée* égale la valeur rythmique précédant la première mesure.

```
\relative {
  \time 3/4
  \partial 4.
  r4 e'8 | a4 c8 b c4 |
}
```



Lorsque `\partial` est utilisé après le début du morceau, ***durée*** égale la valeur rythmique **restant** dans la mesure en cours. Ceci ne crée pas de nouveau numéro de mesure.
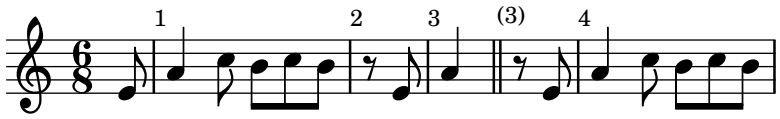
```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
            #end-of-line-invisible
  \time 9/8
  d''4.~ 4 d8 d( c) b | c4.~ 4. \bar "||"
  \time 12/8
  \partial 4.
  c8( d) e | f2.~ 4 f8 a,( c) f |
}
```



La commande `\partial` est **obligatoire** lorsque la métrique change en cours de mesure, mais peut aussi s’utiliser isolément.

```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
            #end-of-line-invisible
  \time 6/8
```

```
  \partial 8
  e'8 | a4 c8 b[ c b] |
  \partial 4
  r8 e,8 | a4 \bar "||"
  \partial 4
  r8 e8 | a4
  c8 b[ c b] |
}
```



La commande `\partial` détermine la propriété `Timing.measurePosition`, nombre rationnel qui indique combien de temps est déjà passé dans la mesure.

Voir aussi

Glossaire musicologique : Section “anacrouse” dans *Glossaire*.

Manuel de notation : [Notes d’ornement], page 119.
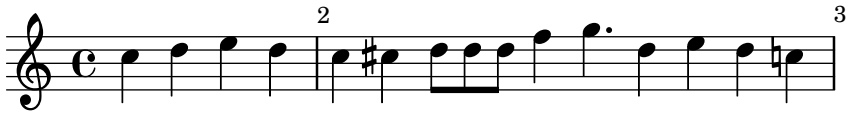
Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “Timing_translator” dans *Référence des propriétés internes*.

Musique sans métrique

Dans un passage tel qu’une cadence, il peut être souhaitable de désactiver la temporisation et tout ce qu’elle entraîne : positionnement des barres et numérotation des mesures, réinitialisation des altérations accidentelles, etc. La musique placée entre `\cadenzaOn` et `\cadenzaOff` ne tient pas compte de la longueur de la mesure.
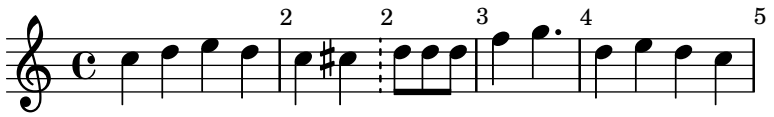
```
\relative c'' {
  % Affiche tous les numéros de mesure
  \override Score.BarNumber.break-visibility = #all-visible
  c4 d e d
  \cadenzaOn
  c4 cis d8[ d d] f4 g4.
  \cadenzaOff
  d4 e d c
}
```



Afin de diviser un passage non mesuré, il suffit de réactiver temporairement la temorisation grâce à l’utilisation de `\partial` pour créer une mesure réduite. La commande `\bar` n’entame pas de nouvelle mesure, même si une barre est imprimée.
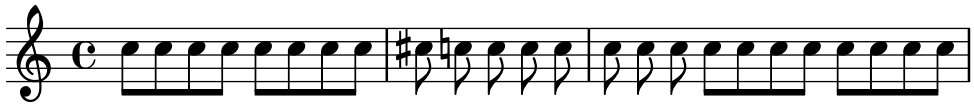
```
cadenzaMeasure = {
  \cadenzaOff
  \partial 1024 s1024
  \cadenzaOn
}
```

```
\relative c'' {
  % Affiche tous les numéros de mesure
  \override Score.BarNumber.break-visibility = #all-visible
  c4 d e d
  \cadenzaOn
  c4 cis \bar "!" d8[ d d] \cadenzaMeasure f4 g4.
  \cadenzaMeasure
  \cadenzaOff
  d4 e d c
}
```



L'instruction `\cadenzaOn` désactive les ligatures automatiques ; elles seront réactivées après un `\cadenzaOff`. Toutes les ligatures devront donc être indiquées de manière explicite tout au long de la cadence – voir [Barres de ligature manuelles], page 99).

```
\relative {
  \repeat unfold 8 { c''8 }
  \cadenzaOn
  cis8 c c c
  \bar"|"
  c8 c c
  \cadenzaOff
  \repeat unfold 8 { c8 }
}
```



Ces commandes prédéfinies affecteront toutes les portées de la partition, même si vous ne les placez que dans un simple contexte `Voice`. Pour éviter ce désagrément, transférez le `Timing_translator` du contexte `Score` au contexte `Staff`, comme indiqué au chapitre [Notation polymétrique], page 80.

Commandes prédéfinies

`\cadenzaOn`, `\cadenzaOff`.

Voir aussi

Glossaire musicologique : Section "cadence" dans *Glossaire*.

Manuel de notation : [Altérations], page 6, [Barres de ligature manuelles], page 99, [Notation polymétrique], page 80, Section 5.4.7 [Visibilité des objets], page 675.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Problèmes connus et avertissements

Sauts de ligne ou de page ne peuvent intervenir qu'au niveau d'une barre de mesure. Si votre musique non mesurée s'étend sur plus d'une ligne, il vous faudra insérer des barres de mesure « invisibles » pour indiquer où des sauts de ligne peuvent prendre place :

```
\bar ""
```

Notation polymétrique

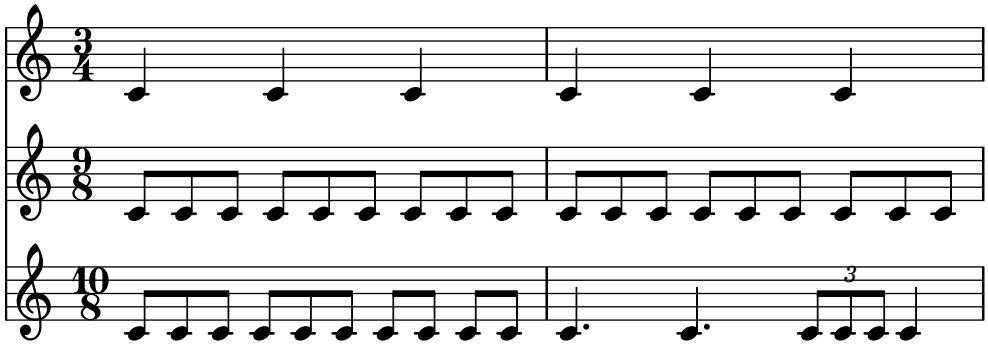
LilyPond gère les métriques composites, aussi bien de manière explicite que de manière détournée – modification de l'indicateur de métrique et échelonnement de la durée des notes.

Métriques différentes et mesures d'égale longueur

Il suffit, pour obtenir cette forme de notation, de tout d'abord affecter une même métrique aux différentes portées. Cette métrique sera ensuite remplacée dans chacune des portées par un quotient fourni en argument à la propriété `timeSignatureFraction`. La durée des notes sera enfin proratisée selon la métrique commune grâce à la fonction `\scaleDurations`.

L'exemple suivant utilise parallèlement des mesures à 3/4, 9/8 et 10/8. Pour la deuxième portée les durées sont multipliées par 2/3 de telle sorte que 2/3 * 9/8 = 3/4 ; pour la troisième elles sont multipliées par 3/5, de telle sorte que 3/5 * 10/8 = 3/4. Les ligatures devront être explicites, la fonction d'échelonnement venant perturber les règles de ligature automatique.

```
\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 9/8
    \scaleDurations 2/3
      \repeat unfold 6 { c8[ c c] }
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 10/8
    \scaleDurations 3/5 {
      \repeat unfold 2 { c8[ c c] }
      \repeat unfold 2 { c8[ c] } |
      c4. c \tuplet 3/2 { c8[ c c] } c4
    }
  }
>>
```



Métriques différentes et mesures de longueur inégale

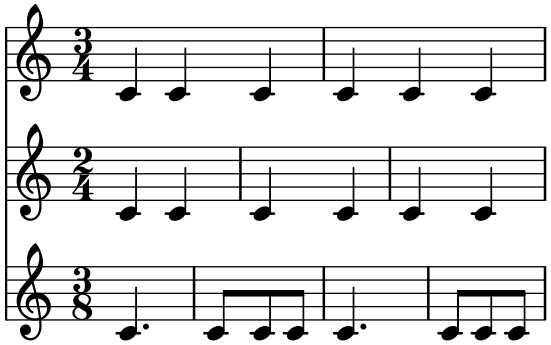
Il arrive aussi que chaque portée ait sa propre métrique. Vous y parviendrez en déplaçant le `Timing_translator` et le `Default_bar_line_engraver` dans le contexte `Staff`.

```
\layout {
```

```
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
}

% Now each staff has its own time signature.

\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \time 2/4
    c4 c |
    c4 c |
    c4 c |
  }
  \new Staff {
    \time 3/8
    c4. |
    c8 c c |
    c4. |
    c8 c c |
  }
>>
```
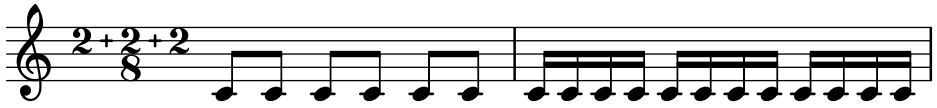


Métriques complexes

Une métrique composite se crée à l'aide de la fonction `\compoundMeter`, en suivant la syntaxe :

```
\compoundMeter #'(liste de listes)
```

La construction la plus simple est constituée d'une seule liste, dans laquelle le *dernier* nombre indique le « dénominateur » de la métrique, les précédents représentent le « numérateur ».

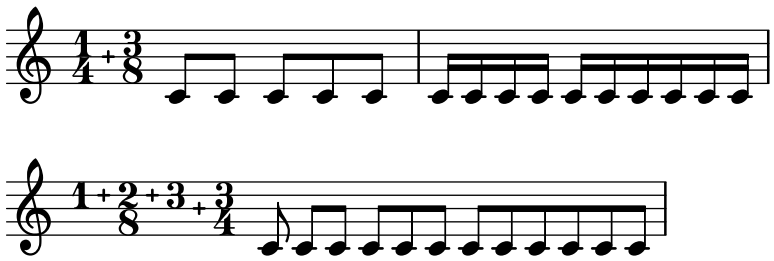
```
\relative {
  \compoundMeter #'((2 2 2 8))
  \repeat unfold 6 c'8 \repeat unfold 12 c16
}
```



Une métrique plus élaborée s'obtient en ajoutant d'autres listes. Bien entendu, les ligatures automatiques s'ajusteront aux différentes valeurs.

```
\relative {
  \compoundMeter #'((1 4) (3 8))
  \repeat unfold 5 c'8 \repeat unfold 10 c16
}

\relative {
  \compoundMeter #'((1 2 3 8) (3 4))
  \repeat unfold 12 c'8
}
```



Voir aussi

Glossaire musicologique : Section "polymétrie" dans *Glossaire*, Section "métrique composite" dans *Glossaire*, Section "métrique" dans *Glossaire*.

Manuel de notation : [Changement d'échelle des durées], page 56, [Barres de ligature automatiques], page 87, [Barres de ligature manuelles], page 99, [Métrique], page 69.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "TimeSignature" dans *Référence des propriétés internes*, Section "Timing_translator" dans *Référence des propriétés internes*, Section "Default_bar_line_engraver" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Bien que les notes de différentes portées tombant au même moment seront positionnées horizontalement au même endroit, les barres de mesure (dans chacune des portées) peuvent causer un espacement incohérent pour chacune des différentes métriques.

L'utilisation d'un bloc `midi` dans le cadre de musique polymétrique peut entraîner des avertissements intempestifs quant au contrôle des barres de mesure. Il est possible de s'en prémunir en déplaçant, au sein du bloc `midi`, le `Timing_translator` du contexte `Score` au contexte `Staff`.
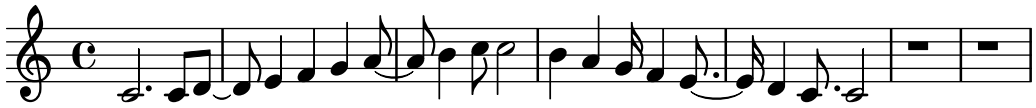
```
\midi {
  \context {
    \Score
```

```
    \remove "Timing_translator"
  }
  \context {
    \Staff
    \consists "Timing_translator"
  }
}
```

Découpage automatique des notes

On peut convertir automatiquement les notes longues en notes liées. Il suffit pour cela de remplacer le graveur `Note_heads_engraver` par le graveur `Completion_heads_engraver`. Il en va de même pour des silences ; le `Completion_rest_engraver` devra alors remplacer le `Rest_engraver`. Dans l'exemple suivant, les notes ou silences qui dépassent de la mesure sont divisés et au besoin liés.
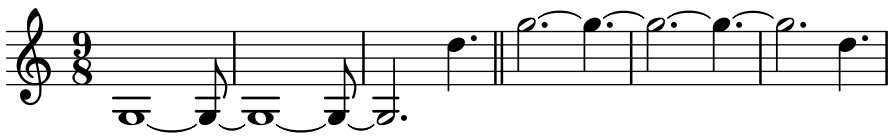
```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
  \remove "Rest_engraver"
  \consists "Completion_rest_engraver"
}
\relative {
  c'2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2
}
```



Ces graveurs divisent toutes les notes et silences qui débordent de la mesure, et insèrent des liaisons de prolongation. Dans la pratique, cette fonctionnalité permet de déboguer des partitions complexes : si les mesures ne sont pas entièrement remplies, alors les liaisons de prolongation montrent exactement la durée des décalages de mesure.

La propriété `completionUnit` détermine la durée de référence pour diviser les notes.
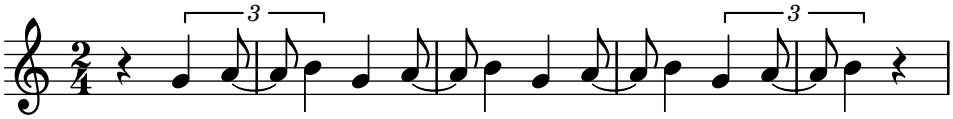
```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
} \relative {
  \time 9/8 g\breve. d''4. \bar "||"
  \set completionUnit = #(ly:make-moment 3 8)
  g\breve. d4.
}
```



Ces graveurs découperont les notes de durée altérée, telles celles d'un triolet, en notes ayant le même facteur d'échelle que les otes saisies.

```
\new Voice \with {
  \remove "Note_heads_engraver"
```

```
  \consists "Completion_heads_engraver"
} \relative {
  \time 2/4 r4
  \tuplet 3/2 {g'4 a b}
  \scaleDurations 2/3 {g a b}
  g4*2/3 a b
  \tuplet 3/2 {g4 a b}
  r4
}
```



Voir aussi

Glossaire musicologique : Section “liaison de tenue” dans *Glossaire*.

Manuel d'initiation : Section “Ajout et suppression de graveurs” dans *Manuel d'initiation*, Section “Tout savoir sur les graveurs” dans *Manuel d'initiation*.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “Completion_heads_engraver” dans *Référence des propriétés internes*, Section “Rest_engraver” dans *Référence des propriétés internes*, Section “Completion_rest_engraver” dans *Référence des propriétés internes*, Section “Forbid_line_break_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

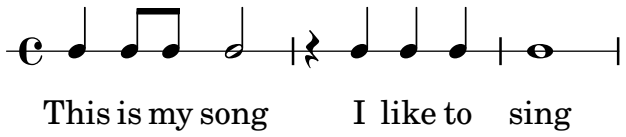
Pour rester cohérent avec le comportement précédent, les notes ou silences d'une durée supérieure à la mesure, comme un `c1*2`, seront divisés sans être mis à l'échelle – on aura donc `{ c1 c1 }`. La propriété `completionFactor`, qui contrôle ce comportement, peut être désactivée (valorisée à `#f`) pour autoriser les notes ou silences divisés à adopter le facteur d'échelle des durées saisies.

Gravure de lignes rythmiques

Au moyen d'une portée rythmique – *rhythmic staff* en anglais – on peut montrer seulement le rythme d'une mélodie : toutes les notes sont ramenées à la même hauteur, sur une portée d'une seule ligne.

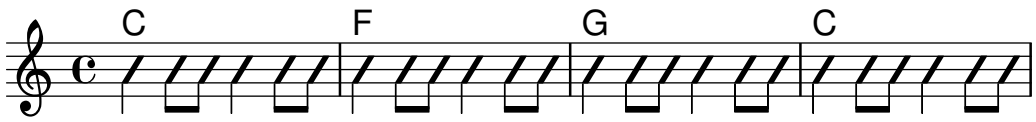
```
<<
  \new RhythmicStaff {
    \new Voice = "myRhythm" \relative {
      \time 4/4
      c'4 e8 f g2
      r4 g g f
      g1
    }
  }
  \new Lyrics {
    \lyricsto "myRhythm" {
      This is my song
      I like to sing
    }
  }
```

```
>>
```



L'utilisation combinée du `Pitch_squash_engraver` et de `\improvisationOn` permet d'afficher la structure rythmique d'une grille d'accords :

```
<<
  \new ChordNames {
    \chordmode {
      c1 f g c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
  }
>>
```



Une musique comportant des accords peut s'utiliser dans un `RhythmicStaff`, et avec le`Pitch_squash_engraver`, dès lors que les accords sont auparavant réduits en notes uniques à l'aide de la fonction musicale `\reduceChords` :

```
\new RhythmicStaff {
  \time 4/4
  \reduceChords {
    <c>2
    <e>2
    <c e g>2
    <c e g>4
    <c e g>4
  }
}
```



Commandes prédéfinies

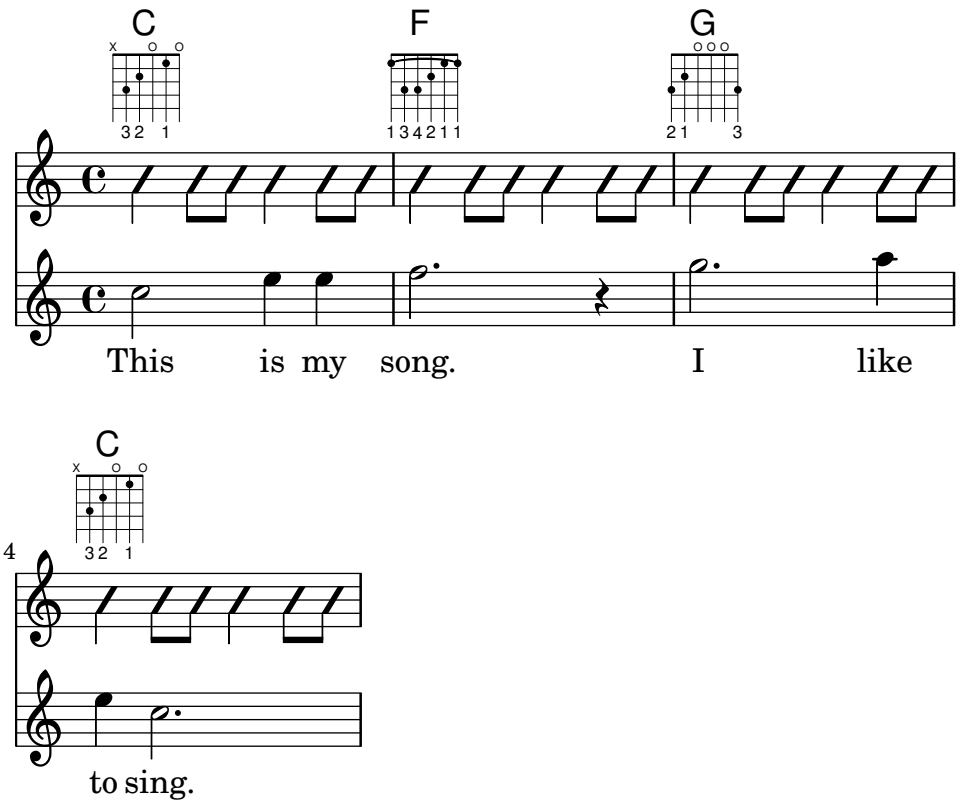
`\improvisationOn`, `\improvisationOff`.

Morceaux choisis

Rythmique et guitare

En matière de notation pour guitare, il arrive que soient indiqués les « coups de gratte » en plus de la mélodie, grilles d'accords et diagrammes de tablature.

```
\include "predefined-guitar-fretboards.ly"
<<
  \new ChordNames {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new FretBoards {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } {
    \relative c'' {
      \improvisationOn
      c4 c8 c c4 c8 c
      f4 f8 f f4 f8 f
      g4 g8 g g4 g8 g
      c4 c8 c c4 c8 c
    }
  }
  \new Voice = "melody" {
    \relative c'' {
      c2 e4 e4
      f2. r4
      g2. a4
      e4 c2.
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      This is my song.
      I like to sing.
    }
  }
>>
```



Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

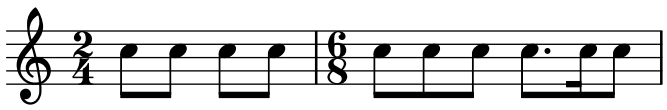
Référence des propriétés internes : Section “RhythmicStaff” dans *Référence des propriétés internes*. Section “Pitch_squash_engraver” dans *Référence des propriétés internes*.

1.2.4 Barres de ligature

Barres de ligature automatiques

LilyPond décide automatiquement de la manière de grouper les notes et d’imprimer les ligatures.

```
\relative c'' {
  \time 2/4 c8 c c c
  \time 6/8 c8 c c c8. c16 c8
}
```
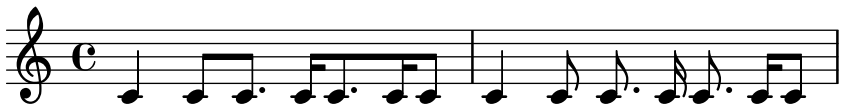


Lorsque ce comportement automatisé n’est pas satisfaisant, on peut définir des groupements manuellement – voir [Barres de ligature manuelles], page 99. Dans le cas où le groupe de notes en question contient un silence, il est **impératif** d’indiquer explicitement les début et fin de la ligature.

Lorsque les ligatures automatiques ne sont pas nécessaires, il suffit de désactiver la fonctionnalité par un `\autoBeamOff` – réactivation par `\autoBeamOn` :

```
\relative c' {
  c4 c8 c8. c16 c8. c16 c8
  \autoBeamOff
  c4 c8 c8. c16 c8.
  \autoBeamOn
  c16 c8
```

```
}
```



Note : Si des ligatures sont utilisées dans les paroles d'une chanson (pour indiquer des mélismes), les ligatures automatiques doivent être désactivées, avec `\autoBeamOff`, et indiquées manuellement.
L'utilisation conjointe de `\partCombine` et de `\autoBeamOff` peut produire des résultats quelque peu surprenants ; ceci fait l'objet d'un exemple particulier à la rubrique morceaux choisis.

Des règles de dérogation au comportement automatique par défaut sont possibles ; voir [Définition des règles de ligature automatique], page 90.

Commandes prédéfinies
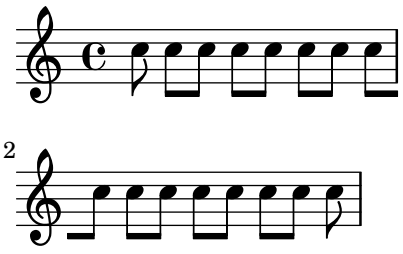
`\autoBeamOff`, `\autoBeamOn`.

Morceaux choisis

Ligature au moment d'un saut de ligne

Il est normalement impensable qu'un saut de ligne tombe au milieu d'une ligature. LilyPond permet néanmoins de l'obtenir.
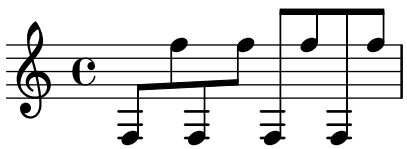
```
\relative c'' {
  \override Beam.breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}
```



Ligature et directions de hampe inversées

LilyPond insère automatiquement des ligatures coudées – certaines hampes vers le haut, d'autres vers le bas – lorsqu'il détecte un intervalle important entre des têtes de notes. Ce comportement peut être changé par l'intermédiaire de l'objet `auto-knee-gap` – défini par défaut à 5,5 espaces, plus la largeur et la pente de la ligature en question.

```
{
  f8 f''8 f8 f''8
  \override Beam.auto-knee-gap = #6
  f8 f''8 f8 f''8
}
```



PartCombine et autoBeamOff

La fonction `\autoBeamOff` dans le cadre d'un `\partCombine` agit de façon bien particulière ; c'est pourquoi il vaut mieux tout d'abord recourir à

```
\set Staff.autobeaming = ##f
```
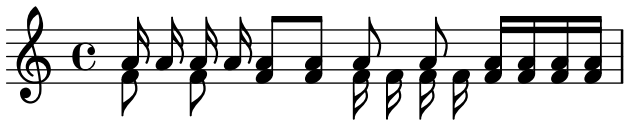
pour désactiver les ligatures automatiques pour l'ensemble de la portée concernée.

L'instruction `\partCombine` fonctionne apparemment sur la base de trois voix : solo hampes montantes, solo hampes descendantes et ensemble hampes montantes.

Lorsque `\autoBeamOff` apparaît dans le premier argument de la combinaison, il s'applique à la voix active à ce moment précis, qu'il s'agisse du solo hampes montantes ou du combiné hampes montantes. Lorsqu'elle est introduite dans le second argument, la commande `\autoBeamOff` s'appliquera au solo hampes descendantes.

Vous devrez donc, afin que `\autoBeamOff` soit pleinement opérationnel dans le cadre d'un `\partCombine`, l'introduire aux **trois** niveaux.

```
{
  %\set Staff.autoBeaming = ##f % turns off all autobeaming
  \partCombine
  {
    \autoBeamOff % applies to split up stems
    \repeat unfold 4 a'16
    %\autoBeamOff % applies to combined up stems
    \repeat unfold 4 a'8
    \repeat unfold 4 a'16
  }
  {
    \autoBeamOff % applies to down stems
    \repeat unfold 4 f'8
    \repeat unfold 8 f'16 |
  }
}
```



Voir aussi

Manuel de notation : [Barres de ligature manuelles], page 99, [Définition des règles de ligature automatique], page 90.

Fichiers d'initialisation : `scm/auto-beam.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "Auto_beam_engraver" dans *Référence des propriétés internes*, Section "Beam_engraver" dans *Référence des propriétés internes*, Section "Beam" dans *Référence des propriétés internes*. Section "BeamEvent" dans *Référence des propriétés internes*, Section "BeamForbidEvent" dans *Référence des propriétés internes*, Section "beam-interface" dans *Référence des propriétés internes*, Section "unbreakable-spanner-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les propriétés d'une ligature sont déterminées **dès le début** de sa construction ; toute adaptation qui interviendrait avant sa terminaison ne sera prise en compte qu'à l'occasion de la **prochaine** ligature.

Définition des règles de ligature automatique

Lorsque la fonction de ligature automatique est active, le positionnement des ligatures dépend des trois propriétés `baseMoment`, `beatStructure` et `beamExceptions`. Les valeurs par défaut de ces variables peuvent s'adapter, comme indiqué ci-après, ou bien carrément être modifiées – voir [Métrique], page 69.

Dès lors qu'une règle affectant `beamExceptions` est définie pour la métrique en vigueur, c'est cette règle qui servira à déterminer le placement des ligatures, ignorant les valeurs de `baseMoment` et `beatStructure`.

En l'absence de règle affectant `beamExceptions` pour la métrique en vigueur, les ligatures seront déterminées par les réglages conjoints de `baseMoment` et `beatStructure`.

Ligature basée sur `baseMoment` *et* `beatStructure`

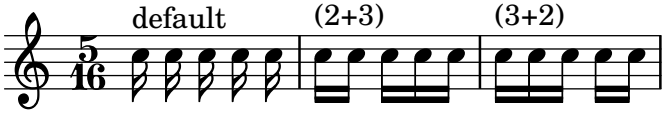
`beamExceptions` dispose par défaut de règles pour les métriques les plus courantes ; il est donc impératif de les invalider pour gérer les ligatures automatiques à l'aide de `baseMoment` et `beatStructure`. Les règles de `beamExceptions` se désactivent par un

```
\set Timing.beamExceptions = #'()
```

Lorsque `beamExceptions` est défini à `#'()`, que ce soit explicitement ou en raison de l'absence de règles par défaut de `beamExceptions` pour la métrique en vigueur, la terminaison des ligatures est directement liée à la pulsation telle que spécifiée par les propriétés `baseMoment` et `beatStructure`. La propriété `beatStructure` est constituée d'une liste d'éléments Scheme qui définit la longueur de chaque pulsation, prenant `baseMoment` comme unité. L'unité de base (le `baseMoment`) correspond par défaut à l'inverse du dénominateur de la métrique. D'autre part, chaque unité de `baseMoment` constitue par défaut une seule pulsation.
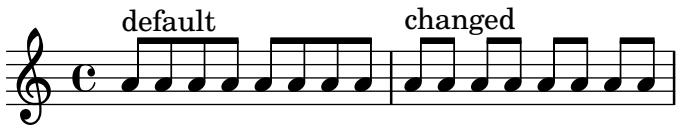
Notez bien la présence de valeurs distinctes de `beatStructure` et `baseMoment` pour chaque métrique. Toute modification de ces variables ne s'applique qu'à la métrique alors en vigueur, raison pour laquelle elles doivent se placer à la suite de la commande `\time` qui entame un fragment ayant une nouvelle métrique, non au préalable. Les nouvelles valeurs affectées à une métrique particulière resteront en vigueur et ré-instaurées si cette métrique réapparaît plus tard.

```
\relative c'' {
  \time 5/16
  c16^"default" c c c c |
  % beamExceptions are unlikely to be defined for 5/16 time,
  % but let's disable them anyway to be sure
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 2,3
  c16^"(2+3)" c c c c |
  \set Timing.beatStructure = 3,2
  c16^"(3+2)" c c c c |
}
```



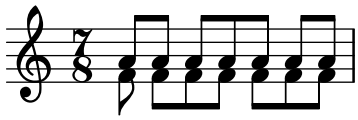

```
\relative {
  \time 4/4
  a'8^"default" a a a a a a a
  % Disable beamExceptions because they are definitely
  % defined for 4/4 time
  \set Timing.beamExceptions = #'()
```

```
  \set Timing.baseMoment = #(ly:make-moment 1/4)
  \set Timing.beatStructure = 1,1,1,1
  a8^"changed" a a a a a a a
}
```



Les effets de règles de ligature peuvent être restreints à un contexte particulier. En l'absence de règle particulière déterminée dans un contexte de niveau inférieur, les règles définies au niveau directement supérieur s'appliqueront.
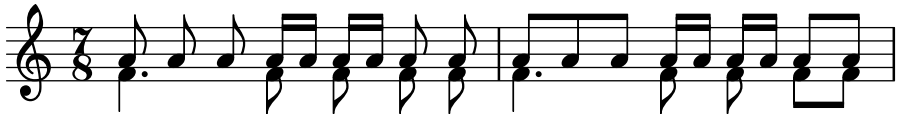
```
\new Staff {
  \time 7/8
  % No need to disable beamExceptions
  % as they are not defined for 7/8 time
  \set Staff.beatStructure = 2,3,2
  <<
    \new Voice = one {
      \relative {
        a'8 a a a a a a
      }
    }
    \new Voice = two {
      \relative {
        \voiceTwo
        \set Voice.beatStructure = 1,3,3
        f'8 f f f f f f
      }
    }
  >>
}
```



Lorsque plusieurs voix cohabitent sur une même portée et que les règles de ligature doivent s'appliquer sans distinction, il faut spécifier que ces règles affectent le contexte `Staff` :
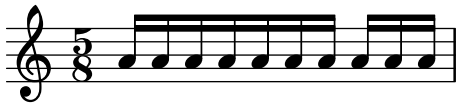
```
\time 7/8
% rhythm 3-1-1-2
% Change applied to Voice by default --  does not work correctly
% Because of autogenerated voices, all beating will
% be at baseMoment (1 . 8)
\set beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>

% Works correctly with context Staff specified
\set Staff.beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>
```



Vous pouvez ajuster la valeur de `baseMoment` afin d'obtenir des ligatures selon vos besoin. Notez cependant que la valeur de `beatStructure` devra être en corrélation avec cette nouvelle valeur de `baseMoment`.

```
\time 5/8
% No need to disable beamExceptions
% as they are not defined for 5/8 time
\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = 7,3
\repeat unfold 10 { a'16 }
```



`baseMoment` constitue un **moment**, autrement dit une unité de durée musicale. La fonction Scheme `ly:make-moment` est tout particulièrement chargée de créer cette quantité de type *moment* – voir [Gestion du temps], page 125, pour plus de précisions.
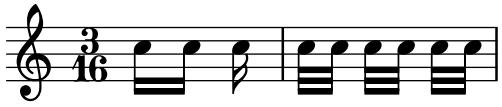
La pulsation – *baseMoment* en anglais – découle directement de la métrique telle que définie par la commande `\time`. Elle est par défaut égale à un sur le dénominateur de la métrique. Les exceptions à cette règle par défaut sont répertoriées dans le fichier `scm/time-signature-settings.scm`. Pour savoir comment jouer avec la valeur de `baseMoment` selon la métrique, reportez vous au chapitre [Métrique], page 69.

Les règles de ligature et de subdivision spécifiques sont enregistrées dans la propriété `beamExceptions`. Ses valeurs par défaut, rangées par métrique et type de règle, sont répertoriées dans le fichier `scm/time-signature-settings.scm`.

Ligature basée sur `beamExceptions`

Les règles spécifiques autres que celles concernant la terminaison des ligatures sont gérées par la propriété `beamExceptions`.

```
\relative c'' {
  \time 3/16
  \set Timing.beatStructure = 2,1
  \set Timing.beamExceptions =
    \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }
  c16 c c |
  \repeat unfold 6 { c32 } |
}
```



Note : La propriété `beamExceptions` doit répertorier absolument **toutes** les exceptions. Il n'est en effet pas possible d'en ajouter, modifier ou supprimer *a posteriori*. Cela peut paraître fastidieux, mais toutes les règles de ligature devraient être appréciées avant de les spécifier.

Lorsqu'intervient un changement de métrique, les valeurs par défaut de `Timing.baseMoment`, `Timing.beatStructure` et `Timing.beamExceptions` sont réinitialisées. Il suffit donc, pour revenir aux règles de ligature par défaut d'un contexte `Timing`, de spécifier à nouveau la métrique.
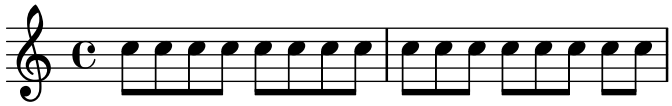
```
\relative a' {
  \time 6/8
  \repeat unfold 6 { a8 }
  % group (4 + 2)
  \set Timing.beatStructure = 4,2
  \repeat unfold 6 { a8 }
  % go back to default behavior
  \time 6/8
  \repeat unfold 6 { a8 }
}
```



Les règles de ligature automatique par défaut sont répertoriées, par métrique, dans le fichier `scm/time-signature-settings.scm`. Les manières de déroger à ce comportement sont abordées au chapitre [Métrique], page 69.

De nombreuses règles de ligature automatique comportent une clé `beamExceptions`. Par exemple, s'il n'y a que des croches dans une mesure à 4/4, celles-ci seront réparties en deux groupes. Le fait de ne pas réinitialiser `beamExceptions` lors d'un aménagement de la pulsation – l'élément `beatStructure` – empêchera l'application de cette dérogation.

```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = 3,3,2
% This won't beam (3 3 2) because of beamExceptions
\repeat unfold 8 {c''8} |
% This will beam (3 3 2) because we clear beamExceptions
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c''8}
```



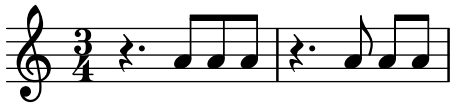
De la même manière, les croches d'une mesure à 3/4 sont ligaturées sur la mesure par défaut. Ligaturer sur le temps requiert un appel à `beamExceptions`.

```
\time 3/4
% by default we beam in (6) due to beamExceptions
\repeat unfold 6 {a'8} |
% This will beam (1 1 1) due to default baseMoment and beatStructure
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a'8}
```



Certaines partitions des périodes romantique ou classique font apparaître des ligatures sur la moitié d'une mesure à 3/4 (ou à 3/8), ce qui va à l'encontre de la règle établie – comme le fait remarquer Gould à la page 153 de son ouvrage – puisque donne l'impression d'une mesure à 6/8. Il en va de même pour une mesure à 3/8. La reproduction d'un tel comportement se contrôle à l'aide de la propriété de contexte `beamHalfMeasure`, qui d'ailleurs ne sera effective que lorsque le numérateur de la métrique est un 3.

```
\relative a' {
  \time 3/4
  r4. a8 a a |
  \set Timing.beamHalfMeasure = ##f
  r4. a8 a a |
}
```



Principes de la ligature automatique

Lorsqu'elle est activée, la gestion automatisée des ligatures est directement liée aux propriétés `baseMoment`, `beatStructure` et `beamExceptions`.

Les règles déterminant le positionnement des ligatures automatiques s'appliquent dans l'ordre suivant de priorité :

- une ligature explicite – indiquée par `[...]` – sera toujours respectée ; sinon
- si une règle explicite de terminaison a été définie grâce à la propriété `beamExceptions` pour un type de ligature spécifique dans la métrique en cours, c'est elle qui s'appliquera ; sinon
- si une règle explicite de terminaison a été définie grâce à la propriété `beamExceptions` pour un type de ligature plus large, c'est elle qui s'appliquera ; sinon
- utilisation des valeurs de `baseMoment` et `beatStructure` pour regrouper les notes par des ligatures.

Le *type de ligature* correspond à la durée la plus courte dans le groupe.

Les règles de ligature par défaut sont répertoriées dans le fichier `scm/time-signature-settings.scm`.

Morceaux choisis

Subdivision des ligatures

Les ligatures d'une succession de notes de durée inférieure à la croche ne sont pas subdivisées par défaut. Autrement dit, tous les traits de ligature (deux ou plus) seront continus. Ce comportement peut être modifié afin de diviser la ligature en sous-groupes grâce à la propriété `subdivideBeams`. Lorsqu'elle est activée, les ligatures seront subdivisées selon un intervalle défini par `baseMoment` ; il n'y aura alors plus que le nombre de traits de ligature déterminé entre chaque sous-groupe. Si le groupe qui suit la division est plus court que la valeur pour la métrique en cours – généralement lorsque la ligature est incomplète –, le nombre de traits de ligature correspond au regroupement de la bubdivision la plus longue. Cette restriction ne sera toutefois pas appliquée dans le cas où ne reste qu'une note après la division. Par défaut, `baseMoment` fixe la valeur de référence par rapport à la métrique en vigueur. Il faudra donc lui fournir, à l'aide de la fonction `ly:make-moment`, une fraction correspondant à la durée du sous-groupe désiré comme dans l'exemple ci-dessous. Gardez à l'esprit que, si vous venez à modifier `baseMoment`, vous devrez probablement adapter `beatStrusture` afin qu'il reste en adéquation avec les nouvelles valeurs de `baseMoment`.

```
\relative c'' {
```

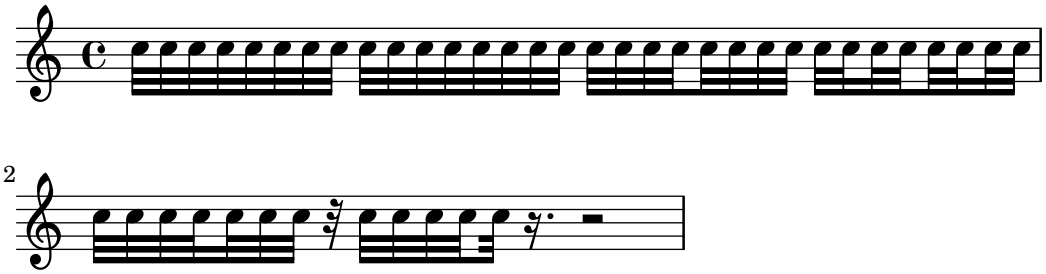
```
  c32[ c c c c c c c]
  \set subdivideBeams = ##t
  c32[ c c c c c c c]

  % Set beam sub-group length to an eighth note
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = 2,2,2,2
  c32[ c c c c c c c]

  % Set beam sub-group length to a sixteenth note
  \set baseMoment = #(ly:make-moment 1/16)
  \set beatStructure = 4,4,4,4
  c32[ c c c c c c c]

  % Shorten beam by 1/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = 2,2,2,2
  c32[ c c c c c c] r32
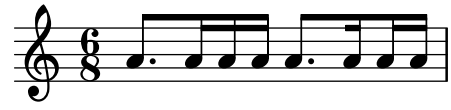
  % Shorten beam by 3/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = 2,2,2,2
  c32[ c c c c] r16.
  r2
}
```



Ligature à la pulsation

Une sous-ligature tronquée peut pointer en direction de la pulsation à laquelle elle se rattache. Dans l'exemple suivant, la première ligature évite toute troncature (comportement par défaut), alors que la deuxième respecte rigoureusement la pulsation.

```
\relative c'' {
  \time 6/8
  a8. a16 a a
  \set strictBeatBeaming = ##t
  a8. a16 a a
}
```
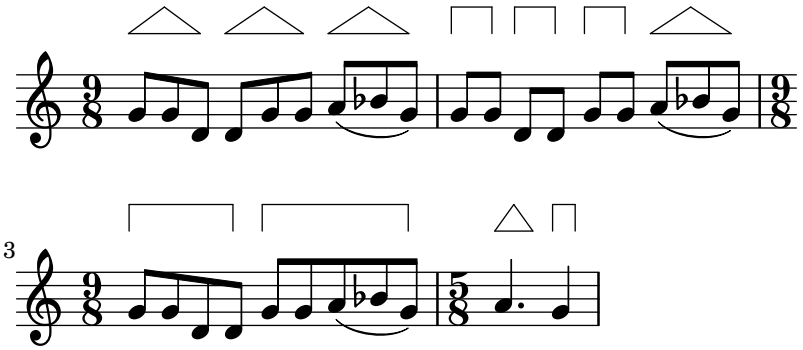


Signes de direction, signes de sous-groupe

Les règles de ligature par mesure sont gérées par la propriété de contexte `beatStructure`. Ses valeurs par défaut sont répertoriées, par métrique, dans le fichier `scm/time-signature-settings.scm`. Elles sont modifiables grâce à la commande `\set`.

La fonction Scheme `set-time-signature` permet quant à elle de définir à la fois la métrique et la pulsation. Celle-ci prend trois arguments : le nombre de pulsations, la durée de la pulsation et le regroupement des pulsations dans la mesure. `\time` et `set-time-signature` s'appliquent tous deux au contexte `Timing` ; ils ne redéfiniront donc pas les valeurs de `beatStructure` ou `baseMoment` lorsqu'elles sont modifiées dans un contexte de niveau inférieur comme `Voice` par exemple.

Si l'on fait appel au `Measure_grouping_engraver`, la fonction `set-time-signature` créera aussi des symboles `MeasureGrouping`. Ces symboles aident à la lecture des œuvres modernes à la rythmique complexe. Dans l'exemple qui suit, la mesure à 9/8 est divisée en 2, 2, 2 et 3, alors que la mesure à 5/8 répond aux règles par défaut contenues dans le fichier `scm/time-signature-settings.scm`.

```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) |
    \time 4,5 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```
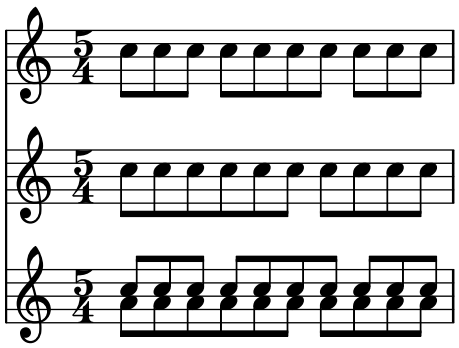


Définition de règles de ligature pour la partition

Les règles de ligature définies au niveau du contexte `Score` s'appliqueront à toutes les portées. Il est toutefois possible de moduler au niveau `Staff` ou `Voice` :

```
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.baseMoment = #(ly:make-moment 1/8)
```

```
\set Score.beatStructure = 3,4,3
<<
  \new Staff {
    c8 c c c c c c c c c
  }
  \new Staff {
    % Modify beaming for just this staff
    \set Staff.beatStructure = 6,4
    c8 c c c c c c c c c
  }
  \new Staff {
    % Inherit beaming from Score context
    <<
      {
        \voiceOne
        c8 c c c c c c c c c
      }
      % Modify beaming for this voice only
      \new Voice {
        \voiceTwo
        \set Voice.beatStructure = 6,4
        a8 a a a a a a a a a
      }
    >>
  }
>>
}
```



Voir aussi

Manuel de notation : [Métrique], page 69.

Fichiers d'initialisation : `scm/time-signature-settings.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Auto_beam_engraver" dans *Référence des propriétés internes*, Section "Beam" dans *Référence des propriétés internes*, Section "BeamForbidEvent" dans *Référence des propriétés internes*, Section "beam-interface" dans *Référence des propriétés internes*.

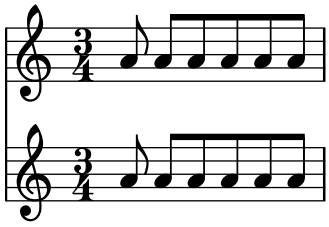
Problèmes connus et avertissements

Si une partition se termine alors qu'une ligature automatique est restée inachevée, cette dernière ligature ne sera pas imprimée du tout. C'est également valable dans le cas d'une musique po-

lyphonique saisie avec la syntaxe `<< ... \\ ... >>`, où une voix se terminerait sans que la dernière ligature ne soit achevée. Le plus simple, en pareil cas, est de spécifier manuellement les dernières ligatures.

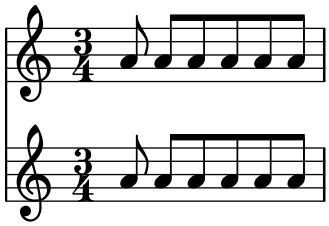
Le traducteur `Timing` est par défaut affecté au contexte `Score`. Définir la métrique dans une portée aura donc des effets sur les ligatures de toutes les autres. Par voie de conséquence, la définition de la métrique apparaissant dans une autre portée annulera les aménagements précédemment apportés aux règles de ligature. Il est donc préférable, pour éviter tout désagrément, de ne spécifier la métrique que dans une seule portée.

```
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1/8)
    \set Timing.beatStructure = 1,5
    \set Timing.beamExceptions = #'()
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \repeat unfold 6 { a'8 }
  }
>>
```



Vous pouvez adapter les règles de ligature par défaut pour une métrique particulière de telle sorte que ces règles que vous aurez définies soient toujours prises en compte. La modification des règles de ligature automatiques est abordée au chapitre [Métrique], page 69.
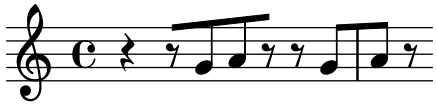
```
<<
  \new Staff {
    \overrideTimeSignatureSettings
      3/4               % timeSignatureFraction
      1/8               % baseMomentFraction
      1,5               % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
>>
```



Barres de ligature manuelles
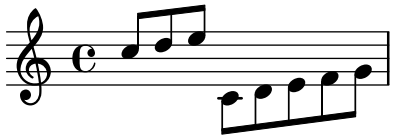
Dans certaines situations, il peut s'avérer nécessaire de supplanter l'algorithme de regroupement automatique des notes, par exemple pour prolonger une ligature par-dessus un silence ou une barre de mesure, ou bien pour suivre le rythme des paroles plutôt que celui des notes. Le début et la fin de la ligature sont alors indiqués respectivement par `[` et `]`.

```
r4 r8[ g' a r] r8 g[ | a] r
```
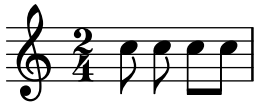


Le positionnement des ligatures manuelles se détermine comme pour toute indication attachée à une note :

```
\relative { c''8^[ d e] c,_[ d e f g] }
```
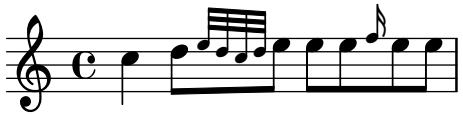


Le fait d'affubler une note particulière d'un `\noBeam` aura pour effet de l'empêcher d'être ligaturée :

```
\relative {
  \time 2/4
  c''8 c\noBeam c c
}
```
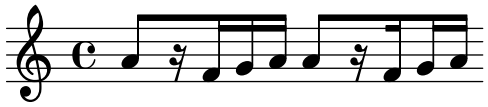


Notes d'ornement et normales font l'objet d'un traitement distinct. Il est dont possible de ligaturer ou non des notes d'ornement sans gêner ce qui est en place au niveau de la notation normale.

```
\relative {
  c''4 d8[
  \grace { e32 d c d }
  e8] e[ e
  \grace { f16 }
  e8 e]
}
```



LilyPond peut déterminer automatiquement les sous-groupes à l'intérieur d'un groupement de notes, bien que le résultat ne soit pas toujours optimal. Les propriétés `stemLeftBeamCount` et `stemRightBeamCount` permettent alors d'ajuster ce comportement. Lorsque l'une ou l'autre de ces propriétés est définie, elle ne s'applique qu'une seule fois, après quoi sa définition est effacée. Dans l'exemple qui suit, le dernier `fa` n'a de ligature supplémentaire que sur sa gauche ; autrement dit, c'est la ligature à la croche qui est importante.

```
\relative a' {
  a8[ r16 f g a]
  a8[ r16
  \set stemLeftBeamCount = #2
  \set stemRightBeamCount = #1
  f16
  \set stemLeftBeamCount = #1
  g16 a]
}
```



Commandes prédéfinies

`\noBeam`.

Morceaux choisis

Crochet rectiligne et débordement de ligature

En combinant `stemLeftBeamCount`, `stemRightBeamCount` et des paires de `[]`, vous pourrez obtenir des crochets rectilignes et des ligatures qui débordent à leurs extrémités.

Pour des crochets rectilignes à droite sur des notes isolées, il suffit d'ajouter une paire d'indicateurs de ligature `[]` et de déterminer `stemLeftBeamCount` à zéro, comme dans l'exemple 1.

Pour des crochets rectiligne à gauche, c'est `stemRightBeamCount` qu'il faudra déterminer (exemple 2).

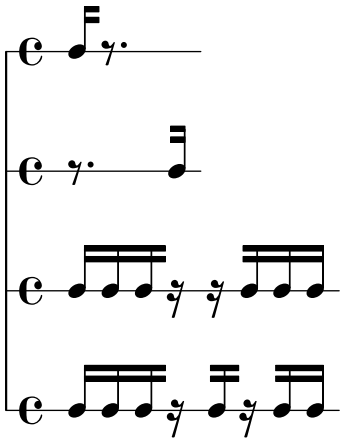
Pour que les barres de ligature débordent sur la droite, `stemRightBeamCount` doit avoir une valeur positive ; pour un débordement à gauche, c'est sur `stemLeftBeamCount` qu'il faut jouer. Tout ceci est illustré par l'exemple 3.

Il est parfois judicieux, lorsqu'une note est encadrée de silences, de l'affubler de crochets rectilignes de part et d'autre. L'exemple 4 montre qu'il suffit d'adjoindre à cette note un `[]`.

(Notez bien que `\set stemLeftBeamCount` sera toujours synonyme de `\once \set`. Autrement dit, la détermination des ligatures n'est pas « permanente » ; c'est la raison pour laquelle les crochets du `16[]` isolé du dernier exemple n'ont rien à voir avec le `\set` indiqué deux notes auparavant.)

```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
```

```
      \new RhythmicStaff {
        r8.
        \set stemRightBeamCount = #0
        16[]
      }
      % Example 3
      \new RhythmicStaff {
        16 16
        \set stemRightBeamCount = #2
        16 r r
        \set stemLeftBeamCount = #2
        16 16 16
      }
      % Example 4
      \new RhythmicStaff {
        16 16
        \set stemRightBeamCount = #2
        16 r16
        16[]
        r16
        \set stemLeftBeamCount = #2
        16 16
      }
    >>
}
```



Voir aussi

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667, [Notes d'ornement], page 119.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Beam” dans *Référence des propriétés internes*, Section “BeamEvent” dans *Référence des propriétés internes*, Section “Beam_engraver” dans *Référence des propriétés internes*, Section “beam-interface” dans *Référence des propriétés internes*, Section “Stem_engraver” dans *Référence des propriétés internes*.

Liens de croches en soufflet
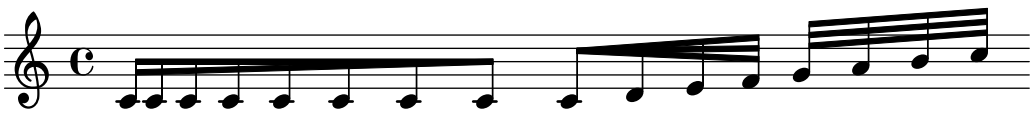
Les ligatures en soufflet permettent d'indiquer qu'un petit groupe de notes se joue en accélérant ou en ralentissant sans pour autant modifier le tempo du morceau. L'étendue du soufflet s'indique par `[` et `]` ; son orientation est déterminée par la propriété `grow-direction` de l'objet `Beam`.

Lorsque la sortie MIDI doit refléter les *ritardando* ou *accelerando* indiqués par une ligature en soufflet, les notes qui la composent doivent être regroupées dans une expression musicale délimitée par des accolades, précédée de la commande `\featherDurations`. Cette commande détermine le ratio entre les durées des premières et dernières notes du groupe en question.

Les crochets indiquent l'étendue de la ligature et les accolades les notes concernées par une modification de leur durée. Il s'agit en général du même groupe de notes, mais les deux commandes sont indépendantes l'une de l'autre.

Dans l'exemple ci-après, les huit doubles croches occupent exactement le même espace qu'une blanche, mais la première est moitié moins longue que la dernière et celles qui les séparent s'allongent peu à peu. Les quatre triples croches qui suivent vont s'accélérant, alors que les quatre dernières gardent un tempo régulier.

```
\relative c' {
  \override Beam.grow-direction = #LEFT
  \featherDurations #(ly:make-moment 2/1)
  { c16[ c c c c c c c] }
  \override Beam.grow-direction = #RIGHT
  \featherDurations #(ly:make-moment 2/3)
  { c32[ d e f] }
  % revert to non-feathered beams
  \override Beam.grow-direction = #'()
  { g32[ a b c] }
}
```



Si le résultat imprimable ne reflète les durées que de manière approximative, la sortie MIDI sera quant à elle parfaitement « ponctuelle ».

Commandes prédéfinies

`\featherDurations`.

Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Problèmes connus et avertissements

La commande `\featherDurations` ne permet de traiter que de très courts extraits, avec une faible amplitude.
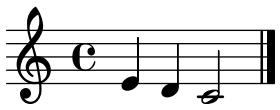
1.2.5 Mesures

Barres de mesure

Les barres de mesures délimitent les mesures, mais peuvent aussi indiquer une reprise. En principe, elles sont insérées automatiquement en respectant la métrique en vigueur.

Il est possible de forcer l'impression d'une barre de mesure spéciale, avec la commande `\bar` – c'est d'ailleurs l'habitude en fin de morceau, où l'on voit une double barre :

```
\relative { e'4 d c2 \bar "|." }
```



Rien ne s'oppose à ce que la dernière note d'une mesure ne s'arrête avant la barre de mesure ; on considère simplement qu'elle se prolonge sur la mesure suivante. Des débordements à répétition finissent par générer une musique comprimée ou qui sort de la page, pour la simple et bonne raison que les sauts de ligne automatiques ne peuvent intervenir qu'à la fin d'une mesure complète, autrement dit lorsque toutes les notes sont terminées avant la fin de la mesure.

> **Note :** Une durée erronée peut empêcher les sauts de ligne, ce qui conduit à une musique compressée, voire à un débordement de la page.

Il est possible d'autoriser un saut de ligne même s'il n'y a pas de barre de mesure visible, en utilisant :

```
\bar ""
```
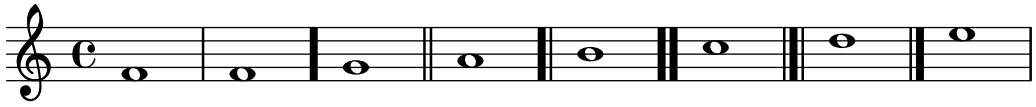
Ceci insérera une barre de mesure invisible, et permettra – sans pour autant le forcer – de sauter de ligne à cet endroit, sans incrémenter le numéro de mesure. Pour forcer le saut de ligne, référez vous à Section 4.3.1 [Sauts de ligne], page 588.

Cette barre invisible, ainsi que d'autres barres spéciales, peuvent être insérées manuellement n'importe où. Lorsqu'elles coïncident avec la fin d'une mesure, elles remplacent la simple barre que LilyPond aurait insérée automatiquement. Dans le cas contraire, la barre spécifiée s'insérera là où vous l'aurez positionnée.

Ces insertions n'affectent en rien le calcul du positionnement automatique des barres de mesure à suivre ni les propriétés y afférentes – numérotation, altérations accidentelles, sauts de ligne. . .
Lorsqu'une barre manuelle est insérée à l'endroit où viendrait se placer une barre normale, seul l'effet visuel en sera modifié.

Vous disposez de deux types de barres simples et de cinq différentes doubles barres :

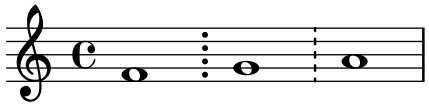
```
\relative {
  f'1 \bar "|"
  f1 \bar "."
  g1 \bar "||"
  a1 \bar ".|"
  b1 \bar ".."
  c1 \bar "|.|"
  d1 \bar "|."
  e1
}
```



ainsi que d'une barre en pointillé et d'une discontinue :

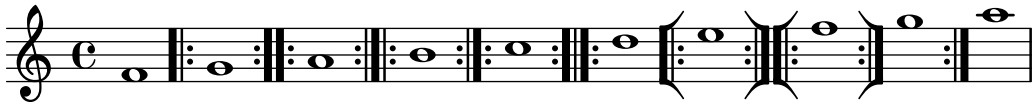
```
\relative {
```

```
  f'1 \bar ";"
  g1 \bar "!"
  a1
}
```



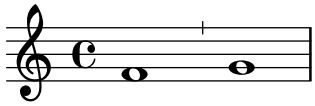
et de neuf types de barre de reprise :

```
\relative {
  f'1 \bar ".|:"
  g1 \bar ":..:"
  a1 \bar ":|.|:"
  b1 \bar ":|.:"
  c1 \bar ":.|.:"
  d1 \bar "[|:"
  e1 \bar ":|][|:"
  f1 \bar ":|]"
  g1 \bar ":|."
  a1
}
```



De plus, une barre de mesure peut s'imprimer sous la forme d'une coche :

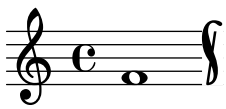
```
f'1 \bar "'" g'1
```



On rencontre habituellement ces signes dans le chant grégorien ; nous vous invitons dans ce cadre particulier à plutôt utiliser `\divisioMinima`, comme indiqué au paragraphe [Divisions], page 481, du chapitre consacré au grégorien.

LilyPond prend en charge la notation kiévienne, qui dispose d'une barre de mesure spécifique :

```
f'1 \bar "k"
```
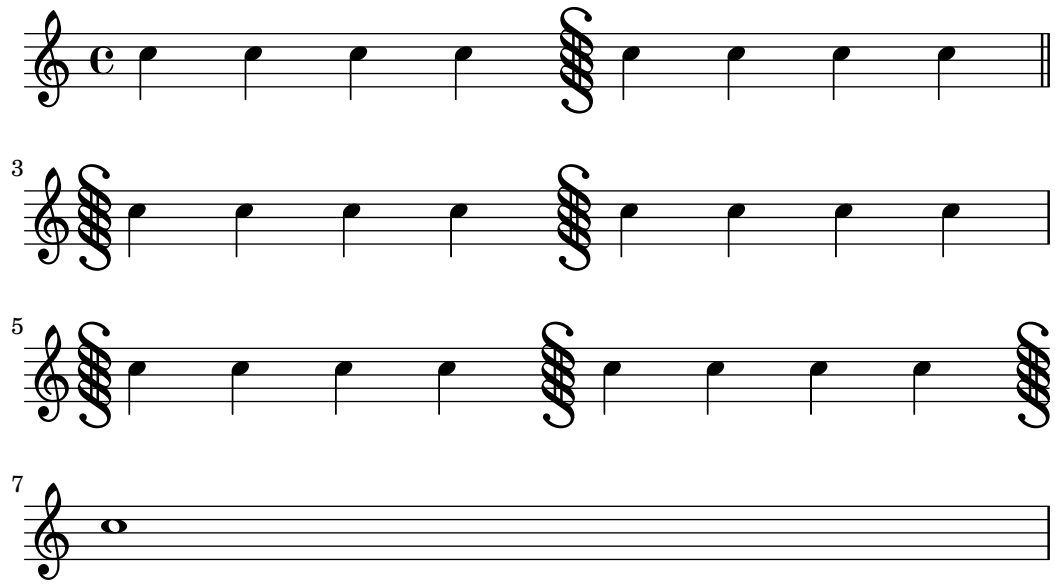


De plus amples détails quant à cette forme de notation sont fournis à la rubrique Section 2.9.5 [Typographie de notation kiévienne], page 490.

L'insertion d'un *segno* directement sur la portée s'obtient à l'aide de trois types de barre de mesure, comme indiqué ci-dessous :

```
\relative c'' {
  c4 c c c
  \bar "S-||"
```

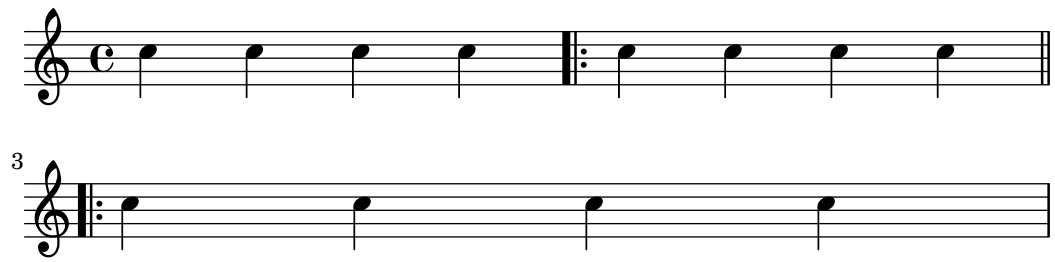
```
  c4 c c c \break
  \bar "S-||"
  c4 c c c
  \bar "S"
  c4 c c c \break
  \bar "S"
  c4 c c c
  \bar "S-S"
  c4 c c c \break
  \bar "S-S"
  c1
}
```



Bien que l'on puisse insérer des barres de reprise manuellement, LilyPond n'en déduira pas pour autant qu'il s'agit d'un passage à répéter. Il est préférable d'indiquer les passages répétés à l'aide des différentes commandes de reprise (voir Section 1.4 [Répétitions et reprises], page 156) qui se chargeront d'imprimer le type de barre approprié.

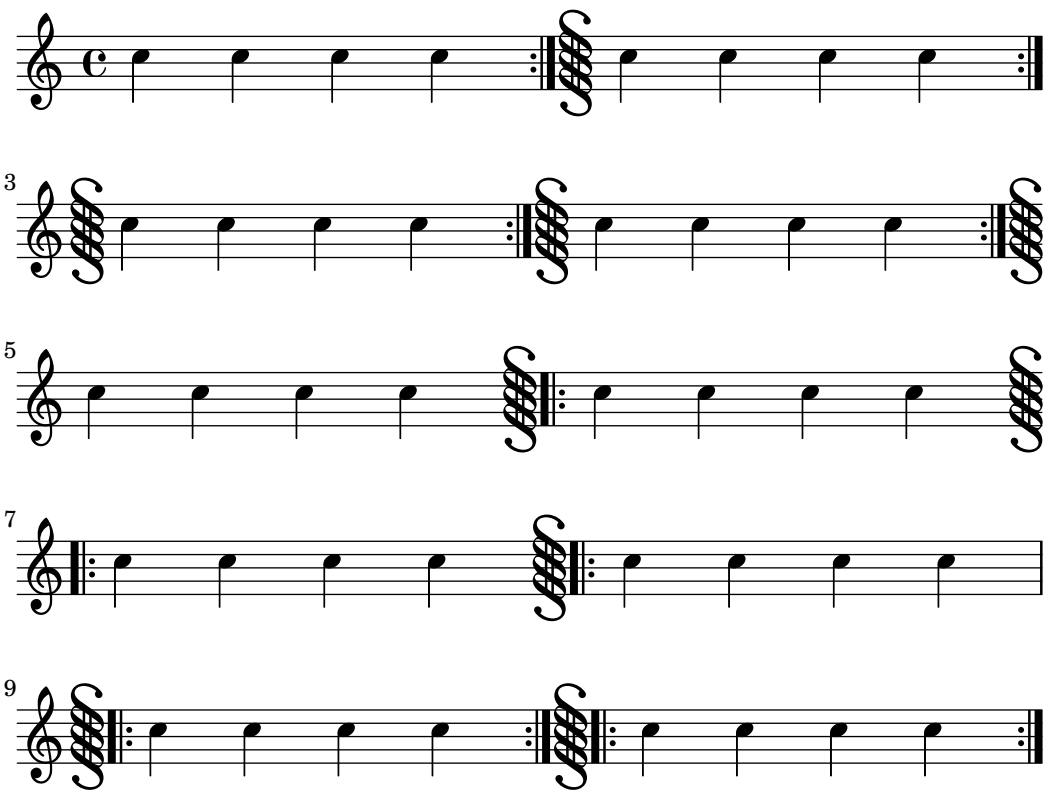
Dans les faits, un `".|:-||"` équivaut à un `".|:"` sauf s'il intervient à un saut de ligne : une double barre sera alors imprimée en fin de portée, et la barre de reprise au début de la nouvelle.

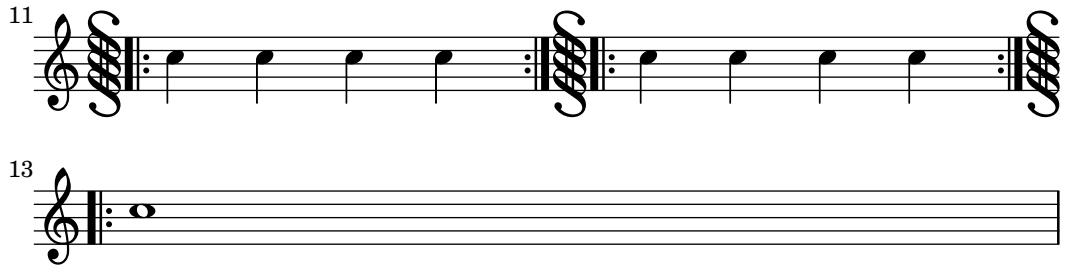
```
\relative c'' {
  c4 c c c
  \bar ".|:-||"
  c4 c c c \break
  \bar ".|:-||"
  c4 c c c
}
```



LilyPond dispose de six différents moyens de combiner une barre de reprise avec un *segno* :

```
\relative c'' {
  c4 c c c
  \bar ":|.S"
  c4 c c c \break
  \bar ":|.S"
  c4 c c c
  \bar ":|.S-S"
  c4 c c c \break
  \bar ":|.S-S"
  c4 c c c
  \bar "S.|:-S"
  c4 c c c \break
  \bar "S.|:-S"
  c4 c c c
  \bar "S.|:"
  c4 c c c \break
  \bar "S.|:"
  c4 c c c
  \bar ":|.S.|:"
  c4 c c c \break
  \bar ":|.S.|:"
  c4 c c c
  \bar ":|.S.|:-S"
  c4 c c c \break
  \bar ":|.S.|:-S"
  c1
}
```





Par ailleurs, la commande `\inStaffSegno` crée une barre de mesure surmontée d'un *segno*, et sait coopérer avec l'instruction `\repeat volta` – voir [Barre de *segno*], page 161.

L'instruction `\defineBarLine` permet de définir ses propres types de barre de mesure, en respectant la syntaxe :

`\defineBarLine` *type_de_barre* `#'(`*fin début extension*`)`

Au-delà de *type_de_barre* (la chaîne de caractères qui sera utilisée pour faire appel à cette nouvelle barre), cette instruction prend trois valeurs : les deux premières déterminent l'apparence de la barre lorsqu'elle intervient sur un saut de ligne, auquel cas les premier et second glyphes indiqués s'imprimeront respectivement à la *fin* du système et au *début* du suivant. Le troisième glyphe indiqué n'est utile que dans le cas d'un système à plusieurs portées (voir [Regroupement de portées], page 199) où il apparaîtra en tant qu'*extension* entre les portées.

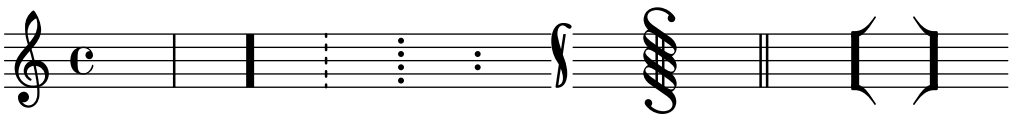
Les variables fournies à `\defineBarline` peuvent inclure la chaîne vide `""` qui correspond à une barre invisible, ou bien être valorisées à `#f` – ce qui aura pour effet de n'imprimer aucune barre.

Une fois la définition explicitée, la nouvelle barre s'utilise à l'aide de `\bar` *type_de_barre*.

Sont à ce jour disponibles dix éléments différents :

```
\defineBarLine ":" #'("" ":" "")
\defineBarLine "=" #'("=" "" "")
\defineBarLine "[" #'("" "[" "")
\defineBarLine "]" #'("]" "" "")

\new Staff {
  s1 \bar "|"
  s1 \bar "."
  s1 \bar "!"
  s1 \bar ";"
  s1 \bar ":"
  s1 \bar "k"
  s1 \bar "S"
  s1 \bar "="
  s1 \bar "["
  s1 \bar "]"
  s1 \bar ""
}
```

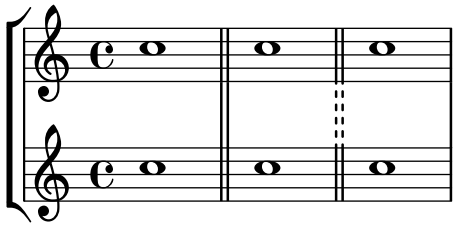


Le type `"="` fournit un double trait destiné à être utilisé en combinaison avec un *segno*. Nous vous recommandons de lui préférer `\bar "||"` pour imprimer une simple double barre fine.

Le signe `"-"` permet d'annoter un type de barre de mesure pour le distinguer lorsqu'il en existe plusieurs ayant la même apparence mais au comportement différent en fin de ligne ou en matière d'extension. Ce qui suit le `"-"` n'est d'aucune utilité dans la construction de la barre.

```
\defineBarLine "||-dashedSpan" #'("||" "" "!!")
```

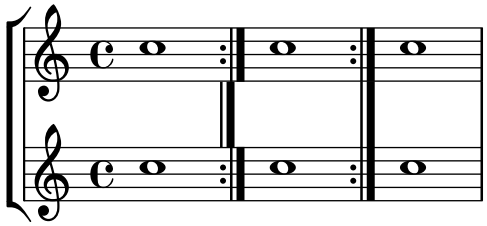
```
\new StaffGroup <<
  \new Staff \relative c'' {
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }
  \new Staff \relative c'' {
    c1
    c1
    c1
  }
>>
```



Par ailleurs, le caractère espace " " permer de préserver de l'espace et ainsi aligner correctement les différents tronçons d'une barre d'un seul tenant entre les portées d'un système :

```
\defineBarLine ":|.-wrong" #'(":|." "" "|.")
\defineBarLine ":|.-right" #'(":|." "" " |.")

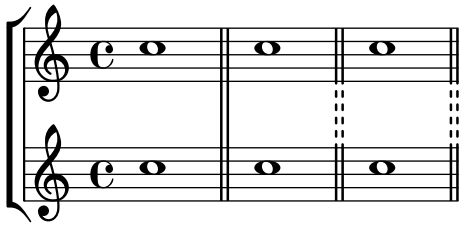
\new StaffGroup <<
  \new Staff \relative c'' {
    c1 \bar ":|.-wrong"
    c1 \bar ":|.-right"
    c1
  }
  \new Staff \relative c'' {
    c1
    c1
    c1
  }
>>
```



Un nouveau type de barre de mesure défini à l'aide de `\defineBarLine` peut même, à son tour, s'utiliser dans la définition d'un autre. Une telle définition « imbriquée » autorise le recours à des glyphes personnalisés là où ce ne serait en principe pas possible, comme en fin de système :

```
\defineBarLine "||-dashEverywhere" #'("!!" "!!" "!!")
\defineBarLine "||-advancedDashSpan" #'("||-dashEverywhere" "" "!!")
```
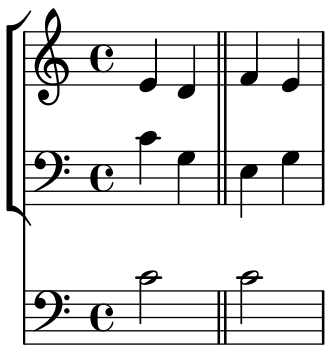
```
\new StaffGroup <<
  \new Staff \relative c'' {
    c1 \bar "||"
    c1 \bar "||-advancedDashSpan"
    c1 \bar "||-advancedDashSpan"
  }
  \new Staff \relative c'' {
    c1
    c1
    c1
  }
>>
```



Si d'autres éléments étaient nécessaires, LilyPond dispose de moyens aisés pour les définir. Pour de plus amples informations quant à la manière de modifier ou ajouter des barres de mesure, consultez le fichier `scm/bar-line.scm`.

Dans une partition comprenant plusieurs portées, la commande `\bar` placée sur une portée s'applique automatiquement à toutes les portées. Les barres de mesure que l'on obtient alors sont d'un seul tenant sur les portées d'un `StaffGroup`, d'un `PianoStaff` ou d'un `GrandStaff`.

```
<<
  \new StaffGroup <<
    \new Staff \relative {
      e'4 d
      \bar "||"
      f4 e
    }
    \new Staff \relative { \clef bass c'4 g e g }
  >>
  \new Staff \relative { \clef bass c'2 c2 }
>>
```



La commande '`\bar` *`type_de_barre`*' sert de raccourci pour '`\set Timing.whichBar =` *`type_de_barre`*'. Dès que l'on définit `whichBar`, une barre de mesure est créée selon le style défini.

Le type de barre de mesure par défaut utilisé pour l'insertion automatique est `"|"`. Vous pouvez en changer à tout moment grâce à '`\set Timing.defaultBarType = type_de_barre`'.

Voir aussi

Manuel de notation : [Regroupement de portées], page 199, Section 1.4 [Répétitions et reprises], page 156, Section 4.3.1 [Sauts de ligne], page 588.

Fichiers d'initialisation : `scm/bar-line.scm`.

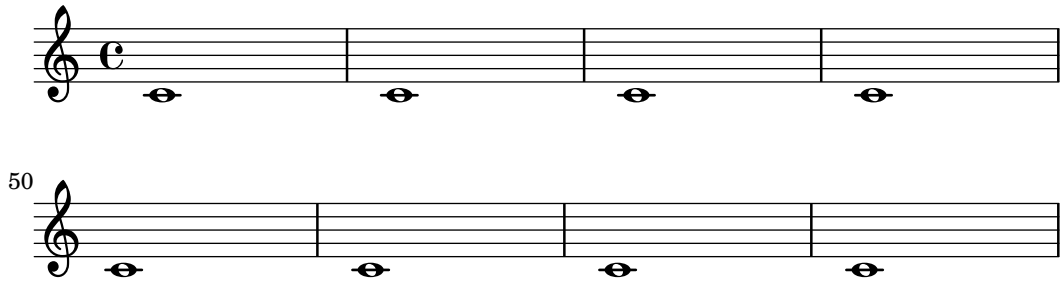
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "BarLine" dans *Référence des propriétés internes* (faisant partie du contexte `Staff`), Section "SpanBar" dans *Référence des propriétés internes* (sur plusieurs portées), Section "Timing_translator" dans *Référence des propriétés internes* (pour les propriétés liées au temps).

Numéros de mesure

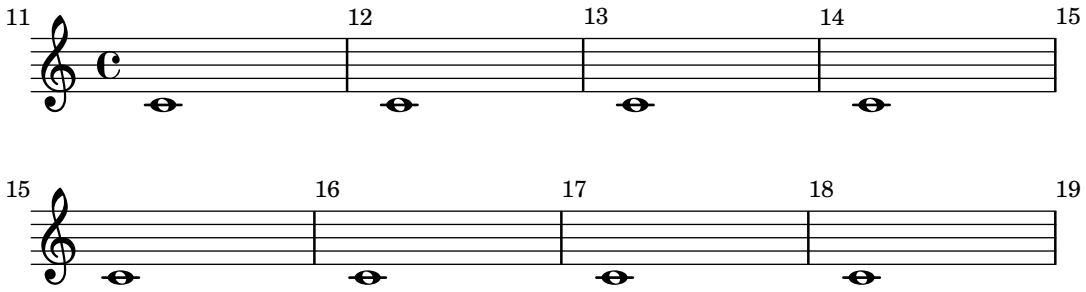
Les numéros de mesure sont imprimés par défaut à chaque début de ligne, sauf la première. Ce nombre est stocké par la propriété `currentBarNumber` qui sera mise à jour à chaque mesure. Vous pouvez aussi le définir de manière arbitraire :

```
\relative c' {
  c1 c c c
  \break
  \set Score.currentBarNumber = #50
  c1 c c c
}
```



Vous pouvez imprimer un numéro de mesure à intervalle régulier plutôt qu'en tête de chaque ligne. Pour y arriver, il faudra dans un premier temps annuler le comportement par défaut afin que les numéros puissent être imprimés ailleurs qu'en début de ligne. Tout ceci est contrôlé par la propriété `break-visibility` du `BarNumber`. Elle se compose de trois commutateurs – définis à « vrai » (`#t`) ou « faux » (`#f`) – pour spécifier si le numéro de mesure est visible ou non. Les valeurs sont rangées dans l'ordre suivant : `visible en fin de ligne`, `visible en cours de ligne` et `visible en début de ligne`. Voici comment imprimer partout les numéros de mesure :

```
\relative c' {
  \override Score.BarNumber.break-visibility = ##(#t #t #t)
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  c1 | c | c | c |
  \break
  c1 | c | c | c |
}
```



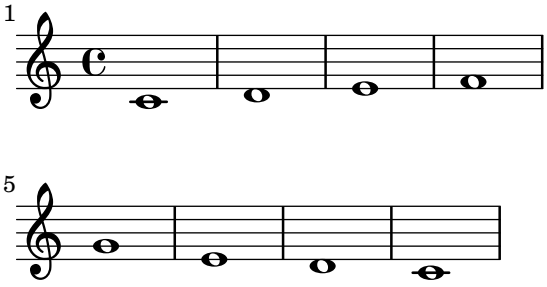
Morceaux choisis

Afficher le numéro de la première mesure

Par défaut, LilyPond n'affiche pas le premier numéro de mesure s'il est inférieur à 2. Le fait de définir `barNumberVisibility` à `all-bar-numbers-visible` vous permettra d'imprimer n'importe quel numéro pour la première mesure. Notez que l'impression d'un numéro de mesure ne peut intervenir que s'il y a une barre. Aussi, pour pouvoir le faire au début d'un morceau, devrez-vous ajouter une barre vide avant la première note.

```
\layout {
  indent = 0
  ragged-right = ##t
}

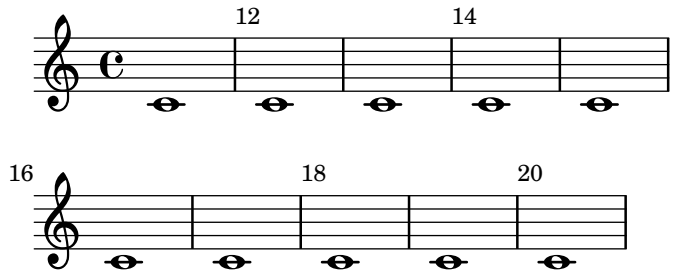
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Imprimer les numéros de mesure à intervalle régulier

Vous pouvez imprimer un numéro de mesure à intervalle régulier plutôt qu'en tête de chaque ligne seulement, en recourant à la propriété `barNumberVisibility`. Voici comment afficher le numéro toutes les deux mesures sauf en fin de ligne.

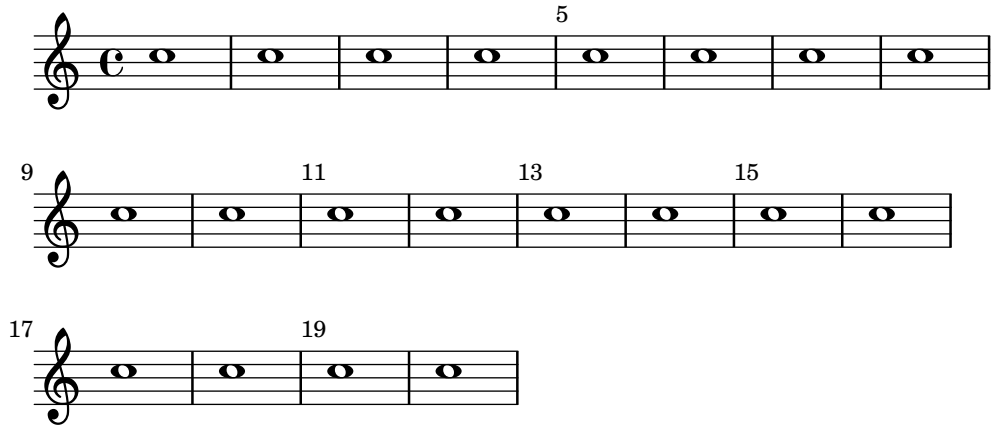
```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```



Changement de la fréquence d'impression du numéro de mesure

La fonction de contexte `{set-bar-number-visibility}` permet de modifier la fréquence à laquelle les numéros de mesures s'impriment.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \context Score \applyContext #(set-bar-number-visibility 4)
  \repeat unfold 10 c'1
  \context Score \applyContext #(set-bar-number-visibility 2)
  \repeat unfold 10 c
}
```

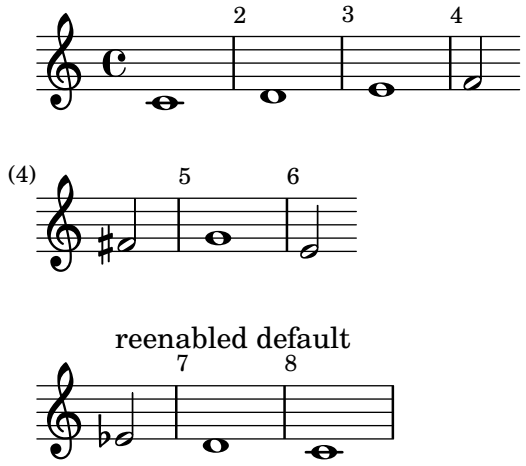


Impression du numéro des mesures tronquées

L'objet `BarNumber` (numéro de mesure) n'est pas répété en début de ligne pour une mesure tronquée. Il apparaîtra, entre parenthèses, dès lors que la propriété `barNumberVisibility` sera affublée de `first-bar-number-invisible-save-broken-bars`.

```
\layout {
  \context {
    \Score
    barNumberVisibility = #first-bar-number-invisible-save-broken-bars
    \override BarNumber.break-visibility = ##(#f #t #t)
  }
}

\relative c' {
  c1 | d | e | f2 \bar "" \break
  fis | g1 | e2 \bar "" \break
  <>^"reenabled default"
  %% back to default, \unset Score.barNumberVisibility would do so as well.
  \set Score.barNumberVisibility =
    #first-bar-number-invisible-and-no-parenthesized-bar-numbers
  es | d1 | c
}
```



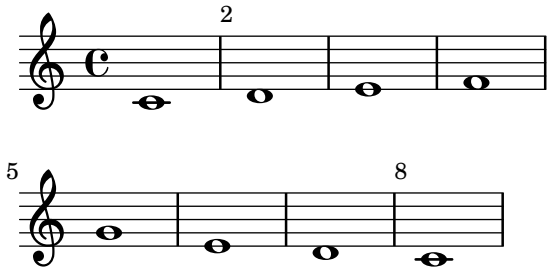
Impression du numéro de mesure selon `modulo-bar-number-visible`

Lorsque le reste de la division du numéro de la mesure courante par le premier argument de `modulo-bar-number-visible` égale le deuxième argument, LilyPond imprime le `BarNumber`. Ceci permet d'imprimer le numéro de mesure à un intervalle donné, par exemple :

- `(modulo-bar-number-visible 3 2)` -> affichage à 2, 5, 8...
- `(modulo-bar-number-visible 4 2)` -> affichage à 2, 6, 10...
- `(modulo-bar-number-visible 3 1)` -> affichage à 3, 5, 7...
- `(modulo-bar-number-visible 5 2)` -> affichage à 2, 7, 12...

```
\layout {
  \context {
    \Score
    \override BarNumber.break-visibility = ##(#f #t #t)
    barNumberVisibility = #(modulo-bar-number-visible 3 2)
  }
}

\relative c' {
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Inscrire le numéro de mesure dans un cadre ou un cercle

Les numéros de mesure peuvent être encadrés ou entourés d'un cercle.

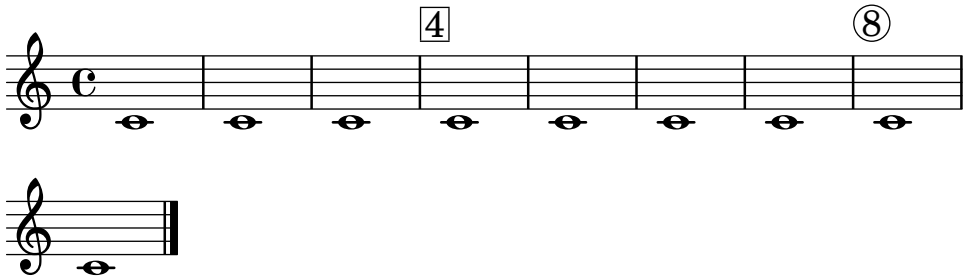
```
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2
```

```
  \override Score.BarNumber.font-size = #2

  % Draw a box round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \repeat unfold 5 { c1 }

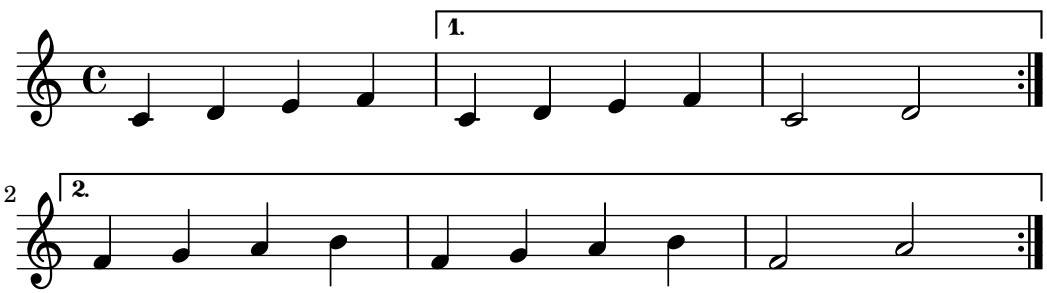
  % Draw a circle round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|."
}
```
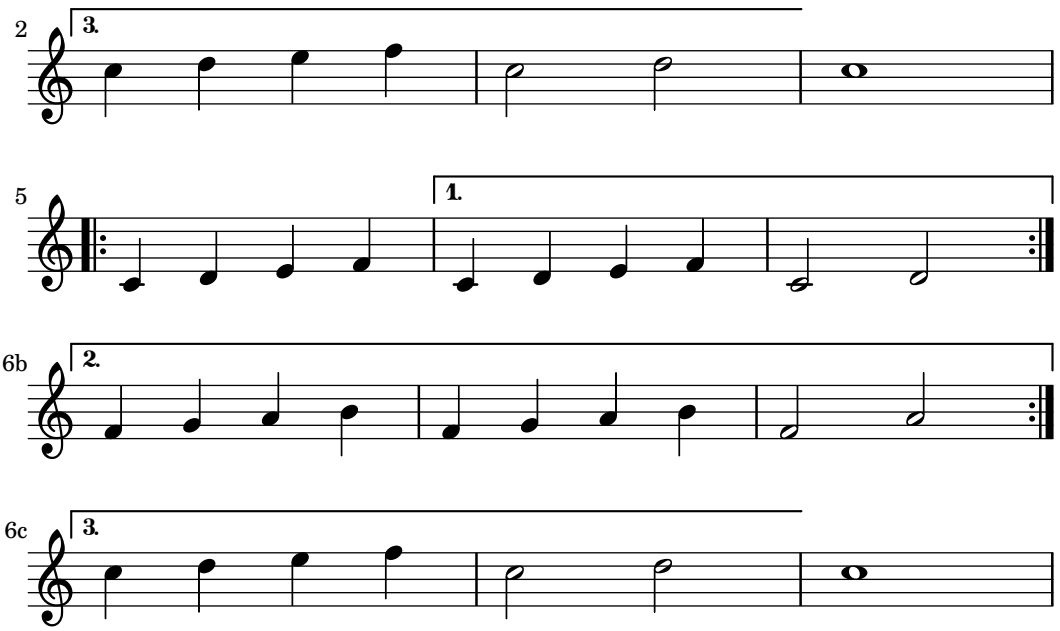


Numérotation des mesures et alternatives

Deux méthodes alternatives vous permettent de gérer la numérotation des mesures en cas de reprises.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```

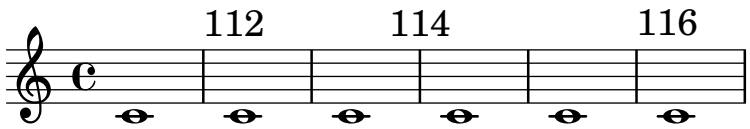




Alignement des numéros de mesure

Les numéros de mesure s'alignent en principe sur la droite de l'objet dont ils dépendent. C'est normalement le coin gauche de la portée ou, en cours de ligne, à gauche de la barre. Vous pouvez toutefois les centrer par rapport à la barre ou les afficher à droite de la barre.

```
\relative c' {
  \set Score.currentBarNumber = #111
  \override Score.BarNumber.break-visibility = #all-visible
  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = #2
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c1
  % Center-align bar numbers
  \override Score.BarNumber.self-alignment-X = #CENTER
  c1 | c1
  % Left-align bar numbers
  \override Score.BarNumber.self-alignment-X = #LEFT
  c1 | c1
}
```
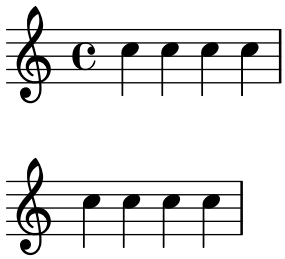


Suppression des numéros de mesure d'une partition

Désactiver le graveur concerné – `Bar_number_engraver` – donnera une partition – contexte `Score` – sans numéros de mesure.

```
\layout {
  \context {
    \Score
    \omit BarNumber
    % or:
    %\remove "Bar_number_engraver"
  }
}
```

```
\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “BarNumber” dans *Référence des propriétés internes*, Section “Bar_number_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les numéros de mesure peuvent entrer en collision avec les crochets d’un Section “StaffGroup” dans *Référence des propriétés internes*. La propriété `padding` – décalage – de l’objet Section “BarNumber” dans *Référence des propriétés internes* permet alors d’ajuster leur positionnement.

Vérification des limites et numéros de mesure

Les tests de limite de mesure (ou tests de mesure) aident à détecter les erreurs dans les durées. Un test de mesure s’écrit avec une barre verticale, `|`. Lors du traitement, elle doit correspondre à une barre de mesure. Sinon, un avertissement est émis qui indique le numéro de ligne où est détectée l’erreur. Dans l’exemple suivant, le deuxième test de mesure signale une erreur.

```
\time 3/4 c2 e4 | g2 |
```

Des durées incorrectes font échouer les tests de mesure et peuvent souvent mettre la partition sens dessus dessous, particulièrement s’il s’agit de musique polyphonique. Vérifier les tests de mesure qui ont échoué et les durées incorrectes est un bon moyen de commencer à corriger sa partition.

Lorsque plusieurs tests successifs présentent un même décalage, seul le message d’avertissement concernant la première occurrence est affiché. L’origine du problème est de fait plus évidente.

Le test de mesure peut être aussi utilisé dans les paroles, par exemple :
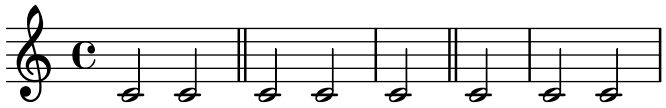
```
\lyricmode {
 \time 2/4
 Twin -- kle | Twin -- kle
}
```

Notez bien qu’en matière de paroles, le test est effectué à l’instant musical où la syllabe **suivant** la marque est traitée. Par voie de conséquence, lorsqu’une mesure débute par un silence, il n’y a pas moyen de positionner une syllabe à cet endroit, et LilyPond émettra un avertissement.

Il est aussi possible d’attribuer une autre valeur au symbole `|`, en assignant une expression musicale à `"|"`. Dans l’exemple suivant, le `|` servira à insérer une double barre là où il apparaît, au lieu de simplement vérifier que la fin de la mesure est atteinte.

```
"|" = \bar "||"
```

```
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Lorsque l'on recopie de longues pièces, il peut être utile de vérifier que les numéros de mesure de LilyPond correspondent à l'original que l'on recopie. Cela se fait avec `\barNumberCheck`. Par exemple,

```
\barNumberCheck #123
```

affiche un avertissement lors du traitement si le numéro de mesure à ce point (variable `currentBarNumber`) n'est pas égal à 123.

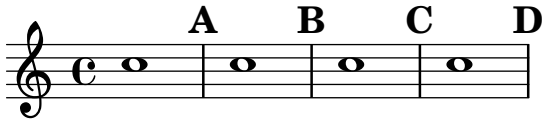
Voir aussi

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Indications de repère

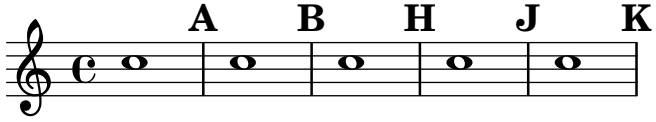
Indiquer un repère s'obtient grâce à la commande `\mark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}
```



Lorsque vous utilisez `\mark \default`, le repère s'incrémente automatiquement ; toutefois donner un nombre en argument permet de spécifier manuellement le repère en question. La valeur à utiliser est enregistrée dans la propriété `rehearsalMark`.

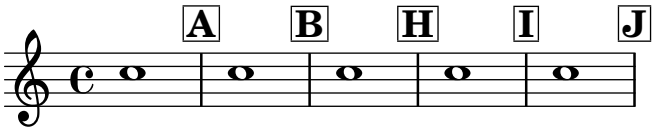
```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```



La lettre **I** n'est pas utilisée, conformément aux usages de la gravure. Cependant, vous pourrez intégrer la lettre **I** en utilisant l'une des commandes suivantes selon que ce repère doive être simple, inclus dans un rectangle ou dans un cercle :

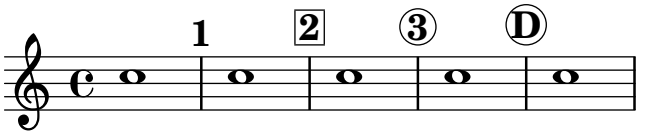
```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet

\relative c'' {
  \set Score.markFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```



Le style de repère est déterminé par la propriété `markFormatter`. Il s'agit d'une fonction qui prend en arguments le repère en cours (un entier) ainsi que le contexte en cours, et retournera un objet de type *markup*. Dans l'exemple qui suit, `markFormatter` est réglé pour une procédure type. Quelques mesures plus loin, son comportement est modifié pour imprimer un repère encadré.

```
\relative c'' {
  \set Score.markFormatter = #format-mark-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = #format-mark-box-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-letters
  c1
}
```



Le fichier `scm/translation-functions.scm` comporte les définitions de `format-mark-letters` (comportement par défaut), `format-mark-box-letters`, `format-mark-numbers` et `format-mark-box-numbers`. Vous pouvez vous en inspirer pour d'autres fonctions de formatage.

`format-mark-barnumbers`, `format-mark-box-barnumbers` et `format-mark-circle-barnumbers` permettent d'imprimer le numéro de mesure au lieu des compteurs alphabétique ou numérique.

On peut aussi spécifier manuellement une marque de repère :

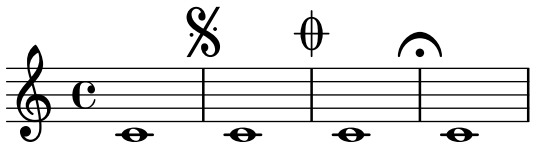
```
\mark "A1"
```

`Score.markFormatter` sera sans effet sur des repères ainsi définis. Un `\markup` peut néanmoins s'utiliser en argument.

```
\mark \markup { \box A1 }
```

Un `\mark` peut contenir un glyphe musical tel que le signe *segno*.

```
\relative c' {
  c1 \mark \markup { \musicglyph "scripts.segno" }
  c1 \mark \markup { \musicglyph "scripts.coda" }
  c1 \mark \markup { \musicglyph "scripts.ufermata" }
  c1
}
```



Pour connaître les différents symboles accessibles par `\musicglyph`, consultez Section A.8 [La fonte Emmentaler], page 718.

Pour affiner le positionnement des repères, veuillez vous référer à Section 1.8.2 [Mise en forme du texte], page 263, et tout particulièrement ce qui concerne la `break-alignable-interface` au chapitre Section 5.5.1 [Alignement des objets], page 684.

Les définitions `format-mark-numbers` et `format-mark-letters` sont inscrites dan le fichier `scm/translation-functions.scm`. Elles seront une source d'inspiration en matière de fonctions de formatage.

Voir aussi

Manuel de notation : Section 5.5.1 [Alignement des objets], page 684, Section A.8 [La fonte Emmentaler], page 718, Section 1.8.2 [Mise en forme du texte], page 263.

Fichiers d'initialisation : `scm/translation-functions.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.
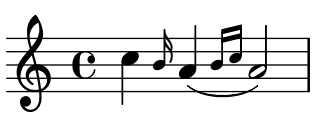
Référence des propriétés internes : Section "MarkEvent" dans *Référence des propriétés internes*, Section "Mark_engraver" dans *Référence des propriétés internes*, Section "RehearsalMark" dans *Référence des propriétés internes*.

1.2.6 Fonctionnalités rythmiques particulières

Notes d'ornement

Les petites notes sont des ornements entièrement écrits. Leur taille est un peu plus petite que celle des notes normales et elles n'occupent pas de temps dans la mesure.
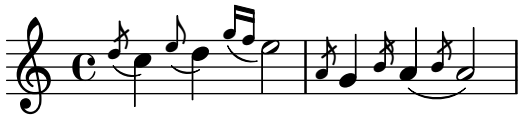
```
\relative {
  c''4 \grace b16 a4(
  \grace { b16 c16 } a2)
}
```



Les plus courantes sont les acciaccatures, qui doivent se jouer très vite, et qui s'écrivent sous forme d'une petite note barrée (sur la hampe) et liée. L'appoggiature est une petite note
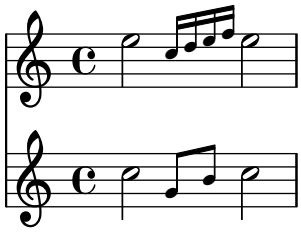
non barrée, qui vole une fraction à la durée de la note réelle qui la suit. LilyPond dispose aussi, grâce à la fonction `\slashedGrace`, d'une petite note barrée et dépourvue de liaison, qui viendra s'insérer entre deux notes déjà liées.

```
\relative {
  \acciaccatura d''8 c4
  \appoggiatura e8 d4
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
  \slashedGrace b8 a2)
}
```
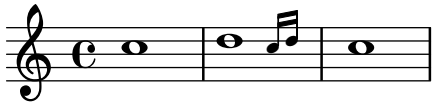


Les petites notes se placent de façon synchrone entre les différentes portées. Dans l'exemple suivant, il y a deux petites double-croches pour chaque petite croche.

```
<<
  \new Staff \relative { e''2 \grace { c16 d e f } e2 }
  \new Staff \relative { c''2 \grace { g8 b } c2 }
>>
```



La commande `\afterGrace` sert à placer une petite note après une note réelle – et non *avant* comme d'ordinaire. Cette commande requiert deux arguments : la note réelle, et la ou les petites notes qui s'y rattachent.

```
\relative { c''1 \afterGrace d1 { c16[ d] } c1 }
```



Les petites notes se placent alors **après** la note réelle. Leur positionnement est déterminé par une fraction de la durée de la note principale. Cette fraction, fixée par défaut à
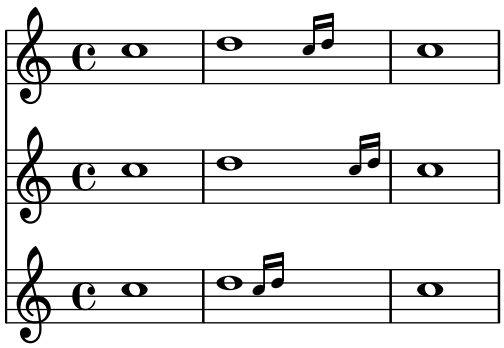
```
afterGraceFraction = 3/4
```

peut être changée en début de fichier. Elle peut aussi se définir à la suite de la commande `afterGraceFraction`.

Dans l'exemple suivant, vous pouvez observer la différence entre le comportement par défaut, à `15/16` et enfin à la moitié de la durée de base.
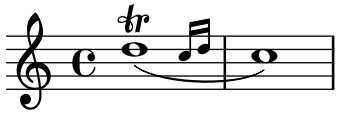
```
<<
  \new Staff \relative {
    c''1 \afterGrace d1 { c16[ d] } c1
  }
```

```
  \new Staff \relative {
    c''1 \afterGrace 15/16 d1 { c16[ d] } c1
  }
  \new Staff \relative {
    c''1 \afterGrace 1/2 d1 { c16[ d] } c1
  }
>>
```
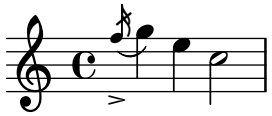


Les effets d'une commande `\afterGrace` peuvent aussi s'obtenir à l'aide de silences invisibles. Nous pourrions positionner ces petites notes à sept huitièmes de la durée de la note de base :

```
\new Voice \relative {
  <<
    { d''1^\trill_( }
    { s2 s4. \grace { c16 d } }
  >>
  c1)
}
```



Les expressions `\grace` obéissent à des règles typographiques particulières, notamment en matière d'orientation et de taille des objets. De ce fait, toute subtilité de mise en forme devra être indiquée **à l'intérieur** de l'expression introduite par `\grace` ; ces réglages additionnels doivent également être désactivés dans cette même expression.

```
\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
}
```



Morceaux choisis

Utilisation de hampe barrée pour une note normale

Le trait que l'on trouve sur les hampes des acciaccatures peut être appliqué dans d'autres situations.

```
\relative c'' {
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}
```
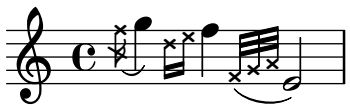


Mise en forme des notes d'ornement

Il est possible de changer globalement la mise en forme des notes d'ornement dans un morceau, au moyen des fonctions `add-grace-property` et `remove-grace-property`.

Ici, par exemple, on ôte la définition de l'orientation des objets `Stem` pour toutes les petites notes, afin que les hampes ne soient pas toujours orientées vers le haut, et on leur préfère des têtes en forme de croix.

```
\relative c'' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
       \acciaccatura { f16 } g4
       \grace { d16 e } f4
       \appoggiatura { f,32 g a } e2
    }
  }
}
```



Redéfinition des réglages de mise en forme par défaut des notes d'ornement

Les réglages par défaut des notes d'ornement sont stockés dans les variables suivantes :
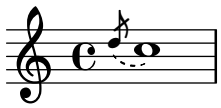
```
startGraceMusic
stopGraceMusic
startAcciaccaturaMusic
stopAcciaccaturaMusic
startAppoggiaturaMusic
stopAppoggiaturaMusic
```

Ces variables sont définies dans le fichier `ly/grace-init.ly`. Amender leur définition permet d'en varier les effets.

```
startAcciaccaturaMusic = {
  <>(
  \override Flag.stroke-style = #"grace"
  \slurDashed
}
```

```
stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>)
}
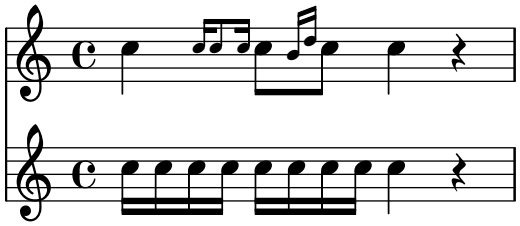
\relative c'' {
  \acciaccatura d8 c1
}
```



Positionnement des notes d'ornement avec espace flottant

Lorsqu'est activée la propriété `strict-grace-spacing`, l'espacement des notes d'ornement se fera de manière « élastique ». Autrement dit, elles seront décollées de leur note de rattachement : LilyPond commence par espacer les notes normales, puis les ornements sont placés à la gauche de leur note de rattachement.

```
\relative c'' {
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section "ornements" dans *Glossaire*, Section "acciaccature" dans *Glossaire*, Section "appoggiature" dans *Glossaire*.

Manuel de notation : [Barres de ligature manuelles], page 99, [Changement d'échelle des durées], page 56.

Fichiers d'initialisation : `ly/grace-init.ly`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "GraceMusic" dans *Référence des propriétés internes*, Section "Grace_auto_beam_engraver" dans *Référence des propriétés internes*, Section
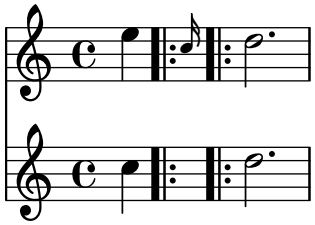
"Grace_beam_engraver" dans *Référence des propriétés internes*, Section "Grace_engraver" dans *Référence des propriétés internes*, Section "Grace_spacing_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un groupe de notes ligaturées constituant une *acciaccatura* apparaîtra comme une *appoggiatura*, c'est-à-dire sans trait.
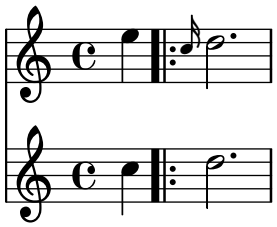
La synchronisation des petites notes se fait de façon parfois surprenante, car les autres objets de la portée – barre de mesure, armure, etc. – sont eux aussi synchrones. Pensez-y lorsque vous mêlez des portées comprenant des petites notes et d'autres sans :

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" d2. }
>>
```



Il est possible de remédier à cela en insérant, sur les autres portées, des silences invisibles dans une expression précédée de `\grace` et correspondant à la durée des petites notes :

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" \grace s16 d2. }
>>
```



Bien que la partie visible contient un `\acciaccatura` ou un `\appoggiatura`, veillez bien à utiliser l'instruction `\grace` dans la partie invisible, au risque de voir apparaître un tronçon de liaison connectant la petite note invisible à la note qui la suit.

Seules des expressions musicales séquentielles peuvent être utilisées pour des petites notes ; il n'est pas possible d'imbriquer ni de juxtaposer des sections de petites notes, faute de quoi le traitement du code peut échouer ou produire des erreurs.

En ce qui concerne la sortie MIDI, les petites notes ont une durée du quart de la valeur que vous leur attribuez. Par voie de conséquence, si la durée globale d'une succession de petites notes venait à dépasser la durée de la note qui précède, vous déclencheriez une erreur du type « `Going back in MIDI time` ». Il vous faudra donc raccourcir les petites notes. Par exemple,

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

deviendrait

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

ou bien modifier explicitement l'échelle des durées :

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

Voir [Changement d'échelle des durées], page 56.

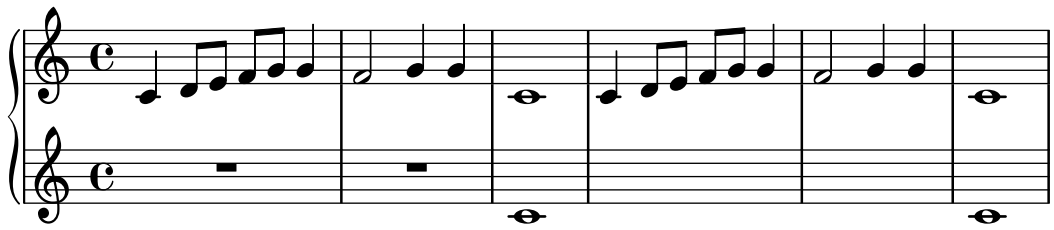
Alignement et cadences

Dans un contexte orchestral, une cadence constitue un problème spécifique. Lors du montage d'une partition contenant une cadence, tous les autres instruments doivent sauter autant de notes que ce qu'en comporte la cadence, faute de quoi ils démarreraient trop tôt ou trop tard.

Les fonctions `mmrest-of-length` ou `skip-of-length` permettent de pallier ce problème. Ces fonctions Scheme prennent en argument un fragment de musique, et génèrent un `\skip` ou un silence multimesure d'une durée correspondant à ce fragment.

```
MyCadenza = \relative {
  c'4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
    c'1
  }
>>
```



Voir aussi

Glossaire musicologique : Section "cadenza" dans *Glossaire*.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Gestion du temps

Le temps est administré par le `Timing_translator`, qui réside en principe dans le contexte `Score`. Un alias, `Timing`, sera ajouté au contexte auquel le `Timing_translator` est rattaché. Déclarer explicitement un contexte `Voice` ou `Staff` assure l'existence de cet alias.

`Timing` dispose des propriétés suivantes afin de garder trace du minutage de la partition.

`currentBarNumber`
: Le numéro de la mesure en cours. Un exemple d'utilisation se trouve au chapitre [Numéros de mesure], page 110.
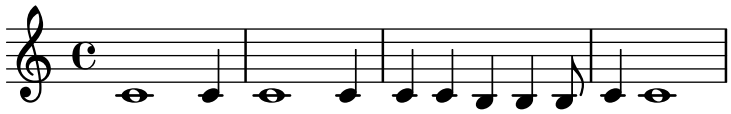
`measureLength`
: La longueur de la mesure, dans la métrique en cours. Pour une mesure à 4/4, elle est de 1, et de 3/4 pour une mesure à 6/8. Sa valeur détermine où peut s'insérer une barre et comment seront générées les ligatures automatiques.

`measurePosition`
: Le moment où l'on en est dans la mesure en cours. Cette quantité est remise à 0 dès lors qu'on dépasse `measureLength` ; la variable `currentBarNumber` est alors incrémentée.

`timing`
: Lorsqu'on lui assigne la valeur *vrai*, les valeurs ci-dessus mentionnées sont mises à jour à chaque pas. Fixée à *faux*, le graveur restera indéfiniment dans la mesure en cours.

Le calage peut être modifié en réglant explicitement l'une de ces variables. Dans l'exemple qui suit, nous réglons la métrique à 4/4, tout en fixant `measureLength` à 5/4. Arrivé à 4/8 dans la troisième mesure, nous avançons de 1/8, en assignant 5/8 à `measurePosition`, raccourcissant donc cette mesure d'une croche. La barre de mesure suivante tombera donc à 9/8 et non à 5/8.

```
\new Voice \relative {
  \set Timing.measureLength = #(ly:make-moment 5/4)
  c'1 c4 |
  c1 c4 |
  c4 c
  \set Timing.measurePosition = #(ly:make-moment 5/8)
  b4 b b8 |
  c4 c1 |
}
```



Comme le montre cet exemple, `ly:make-moment n/d` construit une durée de n/d fois une ronde. Par conséquent, `ly:make-moment 1/8` correspond à une croche, et `ly:make-moment 7/16` à la durée de sept doubles croches.
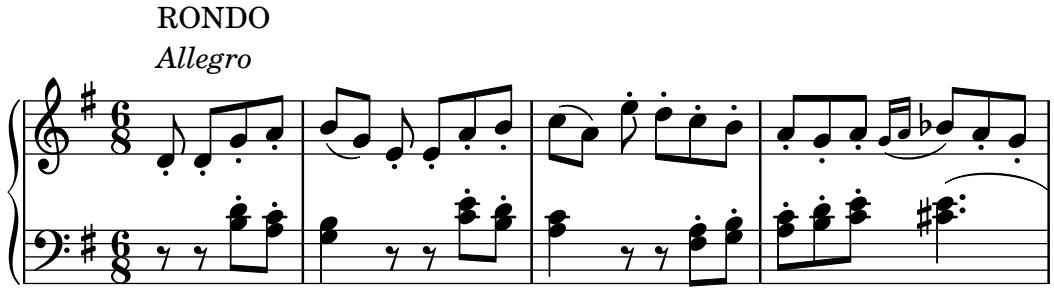
Voir aussi

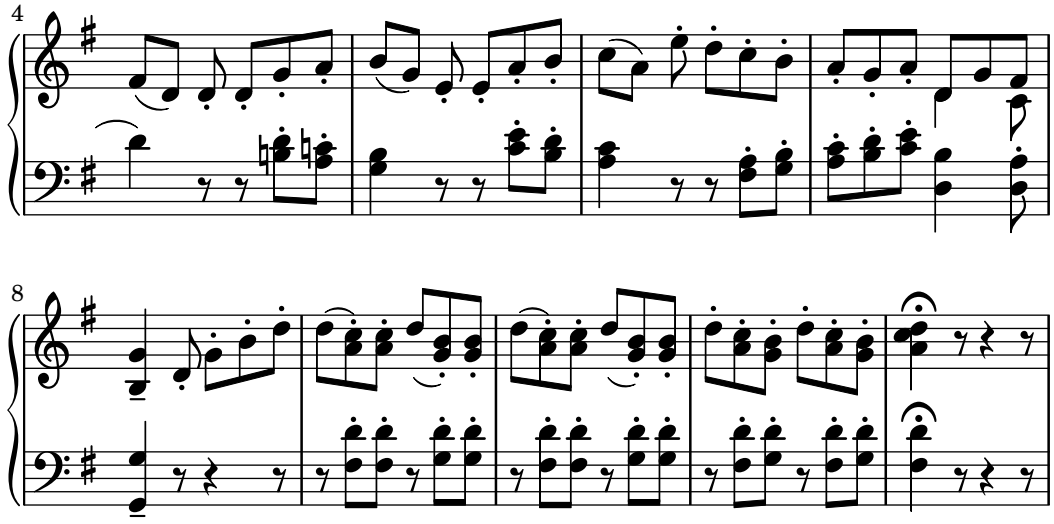
Manuel de notation : [Musique sans métrique], page 78, [Numéros de mesure], page 110.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “Timing_translator” dans *Référence des propriétés internes*, Section “Score” dans *Référence des propriétés internes*

1.3 Signes d'interprétation





Ce chapitre traite des différentes indications d'interprétation que l'on peut trouver sur les partitions.

1.3.1 Signes d'interprétation attachés à des notes

Nous allons voir au cours de ces lignes comment ajouter aux notes des indications d'interprétation – articulation, ornementation, nuance – et aborderons la manière de générer vos propres signes.
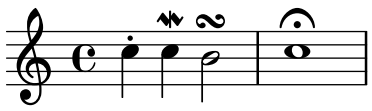
Articulations et ornements

Les différents symboles qui indiquent des ponctuations ou des modes de jeu différents s'ajoutent aux notes de la manière suivante :

*note**nom*
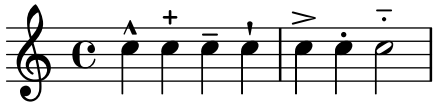
Les valeurs de *nom* sont répertoriées dans l'annexe Section A.14 [Liste des signes d'articulation], page 805. En voici un exemple :

```
\relative {
  c''4\staccato c\mordent b2\turn
  c1\fermata
}
```



Certains signes d'articulation disposent d'un raccourci. On les ajoute à chaque note au moyen d'un tiret suivi du caractère correspondant à l'articulation désirée. C'est entre autres le cas pour *marcato*, *stopped*, *tenuto*, *staccatissimo*, *accent*, *staccato*, et *portato*, comme l'illustre l'exemple ci-dessous.

```
\relative {
  c''4-^  c-+  c--  c-!
  c4->  c-.  c2-_
}
```



Même si LilyPond place automatiquement ces symboles, selon les règles contenues dans le fichier `scm/script.scm`, il est possible de l'obliger à les positionner au-dessus ou en dessous de la note, comme indiqué au chapitre Section 5.4.2 [Direction et positionnement], page 667.

Les articulations sont des objets de type `script` ; les propriétés de ces objets sont abordées plus en détail au chapitre Section “Script” dans *Référence des propriétés internes*.

En dehors des articulations habituelles, vous pouvez adjoindre du texte – avec ou sans mise en forme – à n'importe quelle note. Voir à ce propos [Commentaires textuels], page 256.

Pour plus d'information sur la manière d'ordonner `Scripts` et `TextScripts`, consultez le chapitre Section “Positionnement des objets” dans *Manuel d'initiation*.

Morceaux choisis

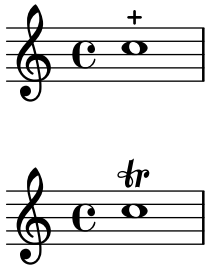
Modification de la signification des raccourcis pour les signes d'articulation

Les raccourcis sont répertoriés dans le fichier ‘`ly/script-init.ly`’, dans lequel on retrouve les variables `dashHat`, `dashPlus`, `dashDash`, `dashBang`, `dashLarger`, `dashDot` et `dashUnderscore` ainsi que leur valeur par défaut. Ces valeurs peuvent être modifiées selon vos besoins. Il suffit par exemple, pour affecter au raccourci `-+` (`dashPlus`) le symbole du trille en lieu et place du `+` (caractère plus), d'assigner la valeur `trill` à la variable `dashPlus` :

```
\relative c'' { c1-+ }

dashPlus = "trill"

\relative c'' { c1-+ }
```
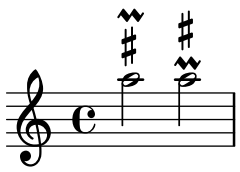


Contrôle de l'ordre vertical des articulations et ornements

Les symboles s'ordonnent verticalement suivant la propriété `script-priority`. Plus sa valeur numérique est faible, plus le symbole sera proche de la note. Dans l'exemple suivant, l'objet `TextScript` – le dièse – a d'abord la propriété la plus basse et se voit donc placé au plus près de la note ; ensuite, c'est l'objet `Script` – le mordant – qui a la propriété la plus basse, et se place alors sous le dièse. Lorsque deux objets ont la même priorité, c'est l'ordre dans lequel ils sont indiqués qui détermine lequel sera placé en premier.

```
\relative c''' {
  \once \override TextScript.script-priority = #-100
  a2^\prall^\markup { \sharp }
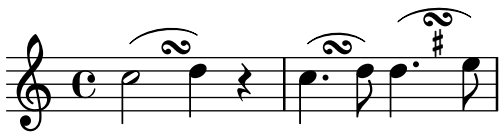
  \once \override Script.script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



Création d'un groupetto retardé

Obtenir un groupetto retardé et dans lequel la note la plus basse est altérée requiert quelques surcharges. La propriété `outside-staff-priority` doit être désactivée (`#f`) pour éviter qu'elle prenne le pas sur la propriété `avoid-slur`. L'ajustement du positionnement horizontal s'effectue en jouant sur les fractions `2/3` et `1/3`.

```
\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
    { c4.( d8) }
    { s4 s\turn }
  >>
  \transpose c d \relative c'' <<
    { c4.( d8) }
    {
      s4
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion.outside-staff-priority = ##f
      \once \override AccidentalSuggestion.avoid-slur = #'inside
      \once \override AccidentalSuggestion.font-size = -3
      \once \override AccidentalSuggestion.script-priority = -1
      \single \hideNotes
      b8-\turn \noBeam
      s8
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section "accent" dans *Glossaire*, Section "portato" dans *Glossaire*, Section "staccato" dans *Glossaire*, Section "tenuto" dans *Glossaire*.

Manuel d'initiation : Section "Positionnement des objets" dans *Manuel d'initiation*.

Manuel de notation : [Commentaires textuels], page 256, Section 5.4.2 [Direction et positionnement], page 667, Section A.14 [Liste des signes d'articulation], page 805, [Trilles], page 154.

Fichiers d'initialisation : `scm/script.scm`.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

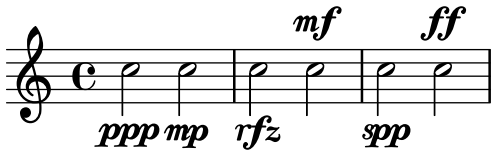
Référence des propriétés internes : Section "Script" dans *Référence des propriétés internes*, Section "TextScript" dans *Référence des propriétés internes*.

Nuances

À chaque nuance absolue correspond une commande qui peut être indiquée après une note : `c4\ff` par exemple. Les commandes de nuance disponibles sont `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, `\rfz`et `\n`. Les nuances se placent aussi bien en dessous qu'au-dessus de la portée ; pour plus d'information, consultez Section 5.4.2 [Direction et positionnement], page 667.

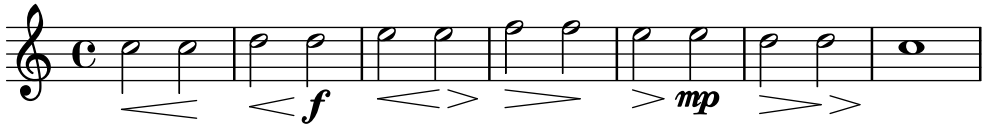
```
\relative c'' {
  c2\ppp c\mp
```

```
  c2\rfz c^\mf
  c2_\spp c^\ff
}
```
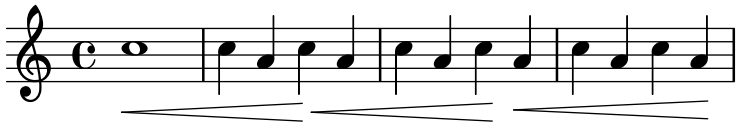


Un crescendo est délimité par `\<` et `\!`, ou peut se terminer par une commande de nuance explicite, ou bien un decrescendo ou un nouveau crescendo. Il en va de même pour un diminuendo. Au lieu de `\<` et `\>`, vous pouvez utiliser `\cr` et `\decr`, tout comme `\endcr` et `\enddecr` au lieu de `\!`, auquel cas LilyPond n'imprimera pas de soufflet (*hairpin* en anglais).

```
\relative c'' {
  c2\< c\!
  d2\< d\f
  e2\< e\>
  f2\> f\!
  e2\> e\mp
  d2\> d\>
  c1\!
}
```
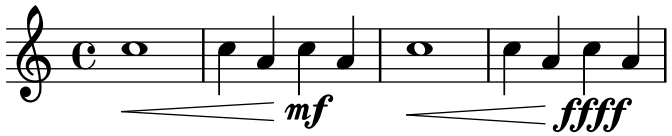


Un soufflet terminé par un simple `\!` prendra fin sur la droite de la tête de note à laquelle il est attaché. Dans le cas où il se termine par l'intervention d'un autre soufflet (contraire ou non), il prendra fin au milieu de la tête de note affublée d'un `\<` ou d'un `\>`, et le nouveau soufflet débutera à l'extrémité droite de cette même tête de note. Un soufflet se terminant sur le premier temps d'une mesure s'arrêtera à la barre de mesure.

```
\relative {
  c''1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
}
```



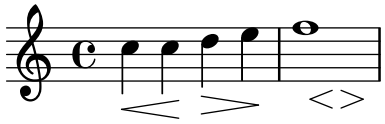
Il en va de même lorsqu'un soufflet est interrompu par une nuance explicite. Notez bien que la largeur occupée par cette nuance explicite influe sur la terminaison du soufflet :

```
\relative {
  c''1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
}
```
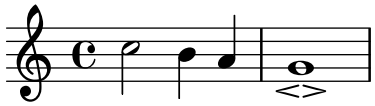


Les indications de nuance sont attachées aux notes ; aussi, lorsque l'on veut faire se succéder plusieurs nuances pendant une note tenue, il faudra avoir recours à des silences invisibles :

```
\relative {
  c''4\< c\! d\> e\!
  << f1 { s4 s4\< s4\> s4\! } >>
}
```
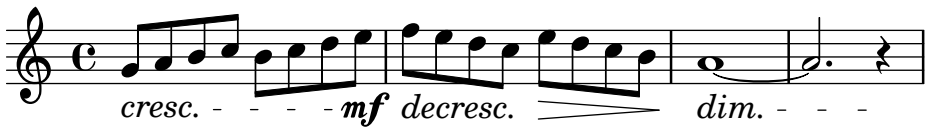


On peut avoir recours à l'indication `\espressivo` pour indiquer un crescendo suivi d'un decrescendo sur une même note. Gardez à l'esprit qu'il s'agit d'une articulation, et en aucun cas d'une nuance.

```
\relative {
  c''2 b4 a
  g1\espressivo
}
```
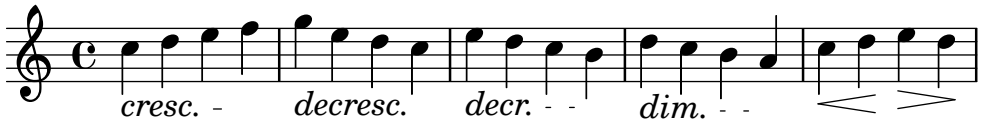


La commande `\cresc` permet d'indiquer textuellement le début d'un crescendo. `\decresc` ou `\dim` marquent le début d'un decrescendo. Les lignes d'extension sont gérées automatiquement.

```
\relative {
  g'8\cresc a b c b c d e\mf |
  f8\decresc e d c e\> d c b |
  a1\dim ~ |
  a2. r4\! |
}
```



Une indication textuelle peut indiquer, au lieu d'un soufflet, un changement de nuance :

```
\relative c'' {
  \crescTextCresc
  c4\< d e f\! |
  \dimTextDecresc
  g4\> e d c\! |
  \dimTextDecr
  e4\> d c b\! |
  \dimTextDim
  d4\> c b a\! |
  \crescHairpin
  \dimHairpin
  c4\< d\! e\> d\! |
}
```
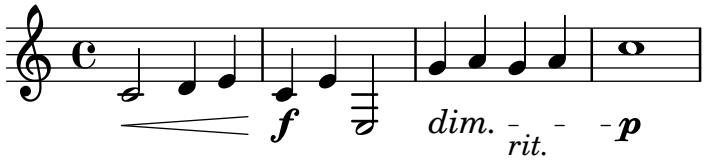


Pour créer des indications de nuance qui restent alignées avec les nuances habituelles, reportez-vous au chapitre [Personnalisation des indications de nuance], page 136.

Le positionnement vertical des nuances est géré par le Section “DynamicLineSpanner” dans *Référence des propriétés internes*.

L'utilisation d'un contexte `Dynamics` permet de graver les nuances sur leur propre ligne – Il suffit de placer des silences invisibles pour gérer le temps. Bien que le contexte `Dynamics` accepte des notes pour indiquer les durées, celles-ci ne seront pas imprimées. Le contexte `Dynamics` peut aussi contenir des indications textuelles avec ou sans extenseur, ainsi que les indications de pédale.

```
<<
  \new Staff \relative {
    c'2 d4 e |
    c4 e e,2 |
    g'4 a g a |
    c1 |
  }
  \new Dynamics {
    s1\< |
    s1\f |
    s2\dim s2-"rit." |
    s1\p |
  }
>>
```



Commandes prédéfinies

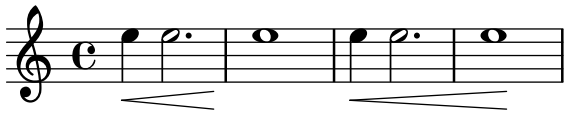
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

Morceaux choisis

Soufflets et barres de mesure

En principe, un soufflet – (de)crescendo imprimé sous forme graphique – commence au bord gauche de la note de départ, et se termine au bord droit de la note d'arrivée. Cependant, si la note d'arrivée est sur un premier temps, le soufflet s'arrêtera au niveau de la barre de mesure qui la précède. Ce comportement peut être annulé en assignant *faux* (`#f`) à la propriété `to-barline`.

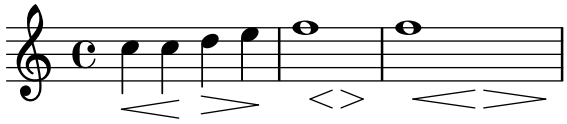
```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}
```



Ajustement de la longueur d'un soufflet

Si un soufflet est trop court, il suffit d'ajuster la propriété `minimum-length` de l'objet `Hairpin` pour l'allonger.
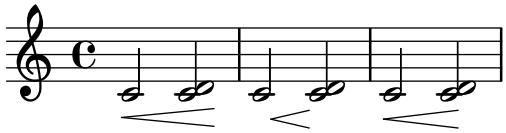
```
\relative c'' {
  c4\< c\! d\> e\!
  << f1 { s4 s\< s\> s\! } >>
  \override Hairpin.minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```



Alignement des bornes de soufflet relativement aux `NoteColumn`*s*

Les terminaisons des soufflets peuvent s'aligner sur la gauche, au centre ou sur la droite des *grobs* `NoteColumn` à l'aide d'une dérogation à la propriété `endpoint-alignments` – `LEFT`, `CENTER` ou `RIGHT` – sous forme de paire représentant les extrémités gauche et droite du soufflet. Les `endpoint-alignments` devraient être des directions (soit −1, 0 ou 1), les autres valeurs entraînant l'émission d'un avertissement. L'extrémité droite d'un soufflet se terminant sur un silence ne sera pas affectée et s'alignera toujours sur le bord gauche de ce silence.

```
{
  c'2\< <c' d'>\! |
  \override Hairpin.endpoint-alignments = #'(1 . -1)
  c'2\< <c' d'>\! |
  \override Hairpin.endpoint-alignments = #`(,LEFT . ,CENTER)
  c'2\< <c' d'>\! |
}
```

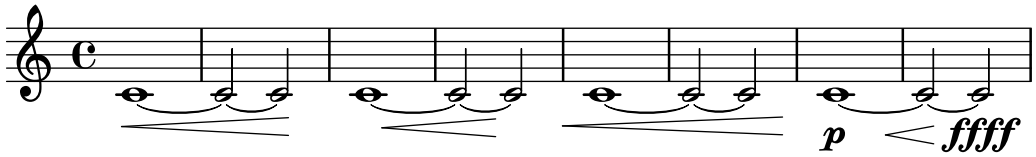


Déplacer les extrémités de soufflets

Les terminaisons des soufflets peuvent être décalées en jouant sur la propriété `shorten-pair` de l'objet`Hairpin`. Les valeurs positives déplacent vers l'intérieur, les négatives vers l'extérieur. Contrairement à la propriété `minimum-length`, cette propriété n'affecte que l'apparence du soufflet sans ajuster l'espacement horizontal (y compris avec les nuances textuelles). Cette méthode est donc appropriée aux situations dans lesquelles un soufflet requiert un ajustement fin dans l'espace qui lui est alloué.

```
{
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(2 . 2)
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(-2 . -2)
  c'1~\<
```
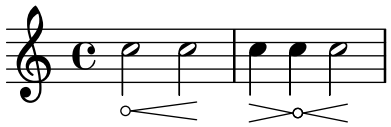
```
  c'2~ c'\!
  c'1~\p-\tweak shorten-pair #'(2 . 0)\<
  c'2~ c'\ffff
}
```



Impression de soufflets « al niente »

Des crescendos ou decrescendos *al niente* peuvent être indiqués de manière graphique, en assignant *vrai* (`#t`) à la propriété `circled-tip`, ce qui affiche un cercle à leur extrémité.
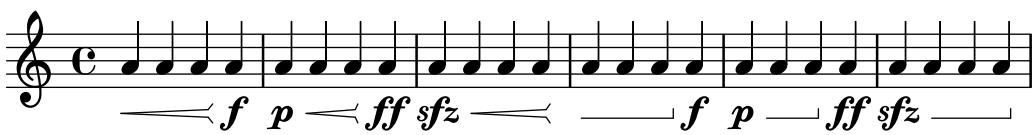
```
\relative c'' {
  \override Hairpin.circled-tip = ##t
  c2\< c\!
  c4\> c\< c2\!
}
```

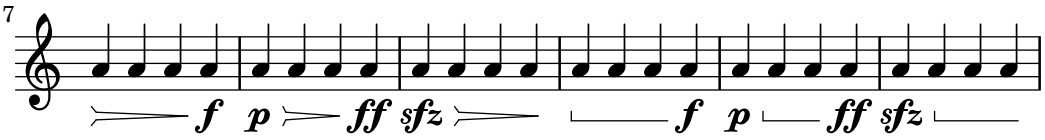


Différents styles de soufflet

Les soufflets de nuance peuvent adopter des styles différents.

```
\relative c'' {
  \override Hairpin.stencil = #flared-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```





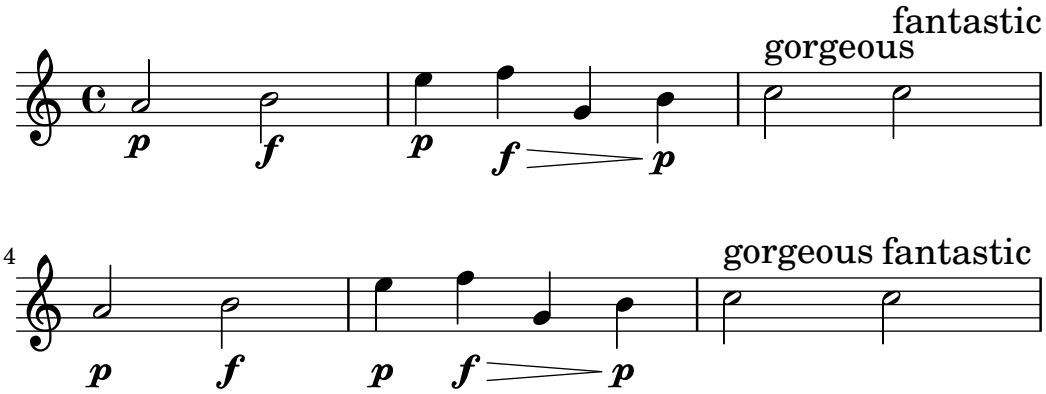
Alignement vertical des nuances indications textuelles

Tous les objets `DynamicLineSpanner` (soufflets ou nuances textuelles) viennent s'aligner sur une ligne de référence placée, par rapport à la portée, à au moins la valeur de `'staff-padding` sauf lorsque d'autres éléments de notation les en éloignent plus. Les nuances seront centrés sur une même ligne dès lors que `'staff-padding` aura été défini à une valeur suffisante.

C'est le même principe – en combinaison avec `\textLengthOn` – qui sert à aligner les indications textuelles sur une ligne de référence.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

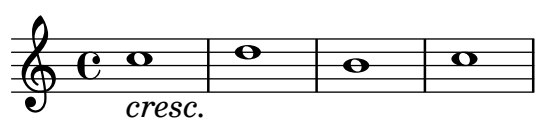
{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = #3
  \textLengthOn
  \override TextScript.staff-padding = #1
  \music
}
```



Masquage de l'extension des nuances textuelles

Les crescendos et decrescendos indiqués textuellement – tels que *cresc.* ou *dim.* – sont suivis de pointillés qui montrent leur étendue. On peut empêcher l'impression de ces pointillés.
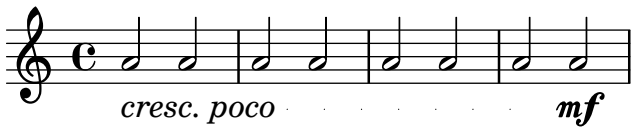
```
\relative c'' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



Modification du texte et de l'extension de nuances textuelles

Le texte par défaut des crescendos et decrescendos se change en modifiant les propriétés de contexte `crescendoText` et `decrescendoText`. L'aspect de la ligne d'extension est fonction de la propriété `style` du `DynamicTextSpanner`. Sa valeur par défaut est `'hairpin`, mais d'autres valeurs sont disponibles, comme `'line`, `'dashed-line` et `'dotted-line`.

```
\relative c'' {
  \set crescendoText = \markup { \italic { cresc. poco } }
  \set crescendoSpanner = #'text
  \override DynamicTextSpanner.style = #'dotted-line
  a2\< a
  a2 a
  a2 a
  a2 a\mf
}
```



Voir aussi

Glossaire musicologique : Section "al niente" dans *Glossaire*, Section "crescendo" dans *Glossaire*, Section "decrescendo" dans *Glossaire*, Section "soufflet" dans *Glossaire*.

Manuel d'initiation : Section "Articulations et nuances" dans *Manuel d'initiation*.

Manuel de notation : Section 3.5.9 [Amélioration du rendu MIDI], page 567, Section 5.4.2 [Direction et positionnement], page 667, Section 3.5.4 [Gestion des nuances en MIDI], page 558, [Personnalisation des indications de nuance], page 136.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.
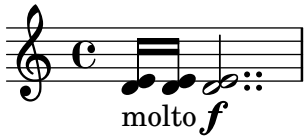
Référence des propriétés internes : Section "DynamicText" dans *Référence des propriétés internes*, Section "Hairpin" dans *Référence des propriétés internes*, Section "DynamicLineSpanner" dans *Référence des propriétés internes*, Section "Dynamics" dans *Référence des propriétés internes*.

Personnalisation des indications de nuance

La manière la plus simple de personnaliser une indication de nuance consiste à utiliser un objet `\markup`.

```
moltoF = \markup { molto \dynamic f }

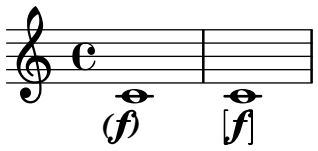
\relative {
  <d' e>16_\moltoF <d e>
  <d e>2..
}
```



Vous pouvez créer des indications de nuance éditoriales (entre parenthèses ou crochets) grâce aux étiquettes (*mode markup*) ; la syntaxe en est abordée au chapitre Section 1.8.2 [Mise en forme du texte], page 263.
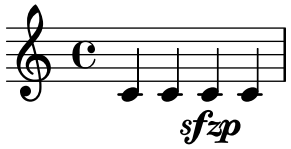
```
roundF = \markup {
```

```
    \center-align \concat { \bold { \italic ( }
          \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative {
  c'1_\roundF
  c1_\boxF
}
```



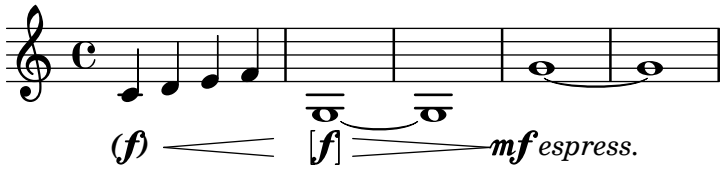
Grâce à la fonction `make-dynamic-script`, vous pouvez créer de nouvelles marques textuelles que vous combinerez avec les signes de nuance.

```
sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfzp c
}
```



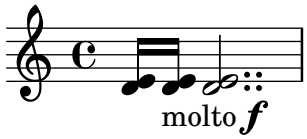
`make-dynamic-script` accepte en argument tout objet de type *markup*. Notez bien que la police des nuances ne contient que les caractères `f`,`m`,`p`,`r`,`s` et `z`, et que les marques de nuance possèdent des propriétés particulières et prédéfinies quant à leur police. Aussi, lorsque vous créez du texte en pareille situation, nous vous recommandons d'utiliser `\normal-text` pour annuler ces propriétés. L'intérêt majeur de recourir à la fonction `make-dynamic-script` plutôt qu'à un simple *markup* réside dans l'assurance que ces objets personnalisés et les soufflets seront alignés lorsqu'attachés à une même note.

```
roundF = \markup { \center-align \concat {
           \normal-text { \bold { \italic ( } }
           \dynamic f
           \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
              \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative {
  c'4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g
  g'~\mfEspressDynamic
  g
}
```
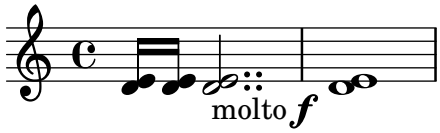


La construction d'une indication de nuance personnalisée peut aussi se faire en langage Scheme ; voir Section "Construction d'un markup en Scheme" dans *Extension de LilyPond* pour en connaître les modalités.

```
moltoF = #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF
}
```



L'utilisation d'un `\tweak` permettra d'aligner par la gauche cette nuance textuelle sur la tête de note, plutôt qu'un centrage :

```
moltoF = \tweak DynamicText.self-alignment-X #LEFT
         #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF <d e>1
}
```



L'utilisation des fontes en mode *markup* est abordée au chapitre [Sélection de la fonte et de la taille], page 265.

Voir aussi

Manuel de notation : Section 3.5.9 [Amélioration du rendu MIDI], page 567, Section 3.5.4 [Gestion des nuances en MIDI], page 558, Section 1.8.2 [Mise en forme du texte], page 263, [Sélection de la fonte et de la taille], page 265.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Manuel d'extension : Section "Construction d'un markup en Scheme" dans *Extension de LilyPond*.

1.3.2 Signes d'interprétation sous forme de courbe
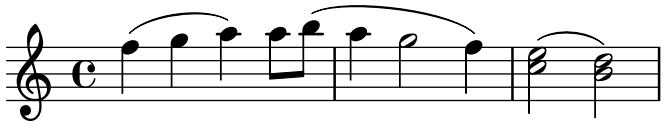
Ce chapitre traite des signes d'interprétation imprimés sous forme de courbe : liaisons d'articulation ou de phrasé, respirations, chutes et sauts.

Liaisons d'articulation

Une liaison d'articulation indique que les notes doivent être jouées liées, ou *legato*. Ces liaisons s'indiquent au moyen de parenthèses.

> **Note :** Lorsque la musique est polyphonique, la liaison doit se terminer dans la voix où elle a été entamée.

```
\relative {
  f''4( g a) a8 b(
  a4 g2 f4)
  <c e>2( <b d>2)
}
```



Vous pouvez décider de l'orientation des liaisons par rapport à la portée, comme indiqué au chapitre Section 5.4.2 [Direction et positionnement], page 667.
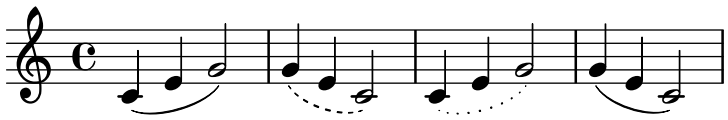
Plusieurs liaisons peuvent intervenir simultanément ou concurremment, ce qui requiert une attention spéciale. Dans la majorité des cas, la liaison externe indique un phrasé, et un phrasé peut recouvrir plusieurs liaisons d'articulation – voir [Liaisons de phrasé], page 141. Dans le cas où plusieurs liaisons d'articulation interviennent au sein d'un même contexte `Voice`, leurs début et fin doivent être labellisés par un `\=` suivi d'un identifiant (symbole ou entier positif).

```
\fixed c' {
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >
}
```



Une liaison est par défaut dessinée d'un trait plein. Il est aussi possible de l'imprimer sous la forme de tirets ou en pointillé :
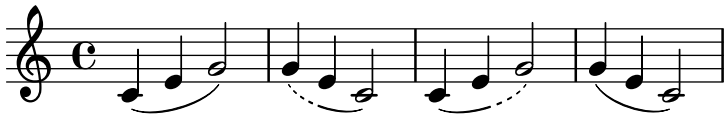
```
\relative {
  c'4( e g2)
  \slurDashed
  g4( e c2)
  \slurDotted
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



En utilisant `\slurHalfDashed`, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L'inverse s'obtient avec `\slurHalfSolid`.
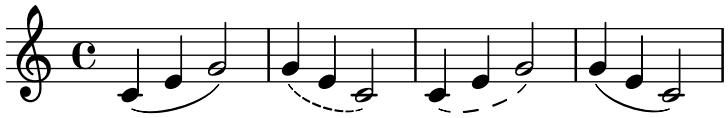
```
\relative {
```

```
  c'4( e g2)
  \slurHalfDashed
  g4( e c2)
  \slurHalfSolid
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
\relative {
  c'4( e g2)
  \slurDashPattern #0.7 #0.75
  g4( e c2)
  \slurDashPattern #0.5 #2.0
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Commandes prédéfinies

`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

Morceaux choisis

Accords et double liaison d'articulation

Certains compositeurs utilisent deux liaisons lorsqu'ils veulent lier des accords. Dans LilyPond, il faut pour cela activer la propriété `doubleSlurs`.

```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```
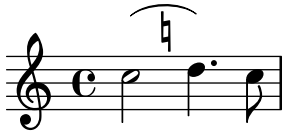


Positionnement d'une annotation à l'intérieur d'une liaison

Lorsqu'une annotation doit s'incrire à l'intérieur d'une liaison, la propriété `outside-staff-priority` doit être désactivée.

```
\relative c'' {
  \override TextScript.avoid-slur = #'inside
```

```
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```
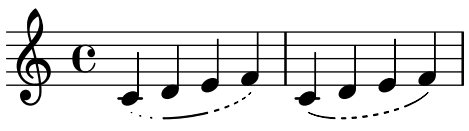


Dessin d'une liaison d'articulation au trait discontinu

Grâce à la propriété `dash-definition`, une liaison d'articulation peut être formée de traits discontinus variables. `dash-definition` se compose d'une liste de `segments-discontinus` (*dash-elements*). Chaque `segment-discontinu` contient une liste de paramètres qui déterminent le comportement du trait pour une section de la liaison.

Cette liaison se définit selon le paramètre de Bézier `t` qui est compris entre 0 (l'extrémité gauche de la liaison) et 1 (l'extrémité droite de la liaison). Chaque `segment-discontinu` se composera selon la liste `(t-début t-fin segment-style segment-taille)`. La portion de liaison allant de `t-début` à `t-fin` aura un trait `segment-style` de longueur `segment-taille`. `segment-taille` est exprimé en espace de portée ; un `segment-style` à 1 donnera un trait plein.

```
\relative c' {
  \once \override
    Slur.dash-definition = #'((0 0.3 0.1 0.75)
                              (0.3 0.6 1 1)
                              (0.65 1.0 0.4 0.75))
  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0 0.25 1 1)
                              (0.3 0.7 0.4 0.75)
                              (0.75 1.0 1 1))
  c4( d e f)
}
```



Voir aussi

Glossaire musicologique : Section "liaison" dans *Glossaire*.

Manuel d'initiation : Section "Non-imbrication des crochets et liaisons" dans *Manuel d'initiation*.

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667, [Liaisons de phrasé], page 141.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Slur" dans *Référence des propriétés internes*.

Liaisons de phrasé

Une liaison de phrasé relie plusieurs notes en délimitant une phrase musicale. On indique les points de départ et d'arrivée avec `\(` et `\)` respectivement.

```
\relative {
```

```
  c''4\( d( e) f(
  e2) d\)
}
```
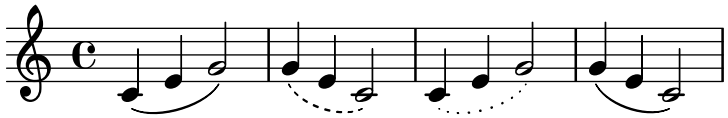


D'un point de vue typographique, rien ne distingue une liaison de phrasé d'une liaison d'articulation. Cependant, LilyPond les considère comme des objets différents. Une commande `\slurUp` n'affectera donc pas une liaison de phrasé. Vous pouvez décider de l'orientation des liaisons de phrasé par rapport à la portée, comme indiqué au chapitre Section 5.4.2 [Direction et positionnement], page 667.

Plusieurs liaisons de phrasé peuvent intervenir en même temps, dès lors qu'elles sont labellisées, comme des liaisons normales – voir [Liaisons d'articulation], page 139.
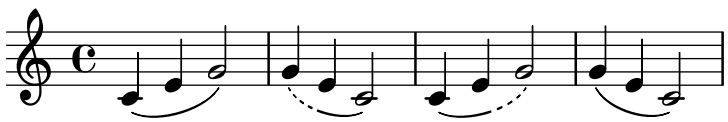
Une liaison est par défaut dessinée d'un trait plein. Il est aussi possible de l'imprimer sous la forme de tirets ou en pointillé :

```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashed
  g4\( e c2\)
  \phrasingSlurDotted
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



En utilisant `\phrasingslurHalfDashed`, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L'inverse s'obtient avec `\phrasingslurHalfSolid`.
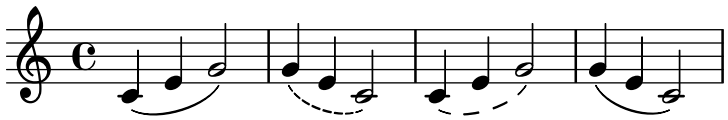
```
\relative {
  c'4\( e g2\)
  \phrasingSlurHalfDashed
  g4\( e c2\)
  \phrasingSlurHalfSolid
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
\relative {
  c'4\( e g2\)
```

```
  \phrasingSlurDashPattern #0.7 #0.75
  g4\( e c2\)
  \phrasingSlurDashPattern #0.5 #2.0
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



La personnalisation des lignes discontinues est identique pour les liaisons de phrasé et les liaisons d'articulation. Pour plus de détails, référez-vous aux morceaux choisis de la section [Liaisons d'articulation], page 139.

Commandes prédéfinies

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`, `\phrasingSlurDashed`, `\phrasingSlurDotted`, `\phrasingSlurHalfDashed`, `\phrasingSlurHalfSolid`, `\phrasingSlurDashPattern`, `\phrasingSlurSolid`.

Voir aussi

Manuel d'initiation : Section "Non-imbrication des crochets et liaisons" dans *Manuel d'initiation*.

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667, [Liaisons d'articulation], page 139.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PhrasingSlur" dans *Référence des propriétés internes*.

Signes de respiration

Les indications de respiration sont indiquées par la commande `\breathe`.
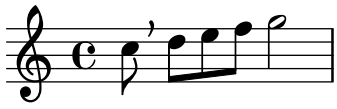
```
{ c''2. \breathe d''4 }
```



Contrairement aux autres signes d'interprétation, une respiration n'est pas associée à la note qui la précède ; il s'agit d'un événement musical à part entière. Par voie de conséquence, toute autre marque attachée à la note précédente, telle un crochet indiquant une ligature manuelle ou une parenthèse indiquant une liaison, doit impérativement se placer avant l'instruction `\breathe`.

Un signe de respiration interrompt obligatoirement les ligatures, même automatiques. Pour passer outre ce fonctionnement, voir [Barres de ligature manuelles], page 99.

```
\relative { c''8 \breathe d e f g2 }
```
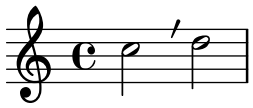


LilyPond gère les *divisiones*, signes utilisés en notation ancienne pour indiquer les respirations. Pour de plus amples détails, voir [Divisions], page 481.

Morceaux choisis

Modification de l'indicateur de respiration

On peut choisir le glyphe imprimé par cette commande, en modifiant la propriété `text` de l'objet `BreathingSign`, pour lui affecter n'importe quelle indication textuelle.
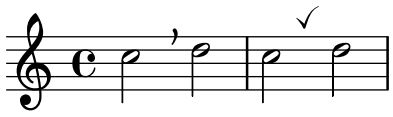
```
\relative c'' {
  c2
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.rvarcomma" }
  \breathe
  d2
}
```



Remplacement du signe de respiration par une coche

Les musiques vocales ou pour vents utilisent souvent une coche en tant que signe de respiration. Ceci indique une respiration qui enlève une fraction à la note précédente plutôt qu'une véritable pause comme le fait un signe sous forme de virgule. La coche peut être remontée un peu afin de l'isoler de la portée.

```
\relative c'' {
  c2
  \breathe
  d2
  \override BreathingSign.Y-offset = #2.6
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.tickmark" }
  c2
  \breathe
  d2
}
```



Insertion d'une césure
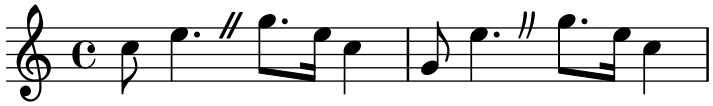
Une surcharge de la propriété `text` de l'objet `BreathingSign` permet de créer une marque de césure.

LilyPond dispose également d'une variante courbée.

```
\relative c'' {
  \override BreathingSign.text = \markup {
    \musicglyph "scripts.caesura.straight"
  }
  c8 e4. \breathe g8. e16 c4

  \override BreathingSign.text = \markup {
    \musicglyph "scripts.caesura.curved"
  }
```

```
  g8 e'4. \breathe g8. e16 c4
}
```



Voir aussi

Glossaire musicologique : Section “césure” dans *Glossaire*.

Manuel de notation : [Divisions], page 481.

Morceaux choisis : Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “BreathingEvent” dans *Référence des propriétés internes*, Section “BreathingSign” dans *Référence des propriétés internes*, Section “Breathing_sign_engraver” dans *Référence des propriétés internes*.

Chutes et sauts

Des indications de désinence peuvent être obtenues au moyen de la commande `\bendAfter`. Leur direction s’indique au moyen des signes plus (vers le haut) ou moins (vers le bas). Le chiffre indique l’intervalle avec la note de départ.

```
\relative c'' {
  c2\bendAfter #+4
  c2\bendAfter #-4
  c2\bendAfter #+6.5
  c2\bendAfter #-6.5
  c2\bendAfter #+8
  c2\bendAfter #-8
}
```



Morceaux choisis

Ajustement du galbe des chutes ou sauts

La propriété `shortest-duration-space` peut devoir être retouchée pour ajuster l’apparence des chutes ou sauts.

```
\relative c'' {
  \override Score.SpacingSpanner.shortest-duration-space = #4.0
  c2-\bendAfter #5
  c2-\bendAfter #-4.75
  c2-\bendAfter #8.5
  c2-\bendAfter #-6
}
```



Voir aussi

Glossaire musical : Section “chute” dans *Glossaire*, Section “saut” dans *Glossaire*.

Morceaux choisis : Section “Signes d’interprétation” dans *Morceaux choisis*.

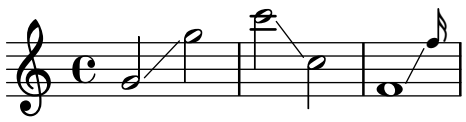
1.3.3 Signes d’interprétation sous forme de ligne

Cette partie traite de la manière de générer des signes d’interprétation d’aspect linéaire, tels les glissandos, arpèges et trilles.

Glissando

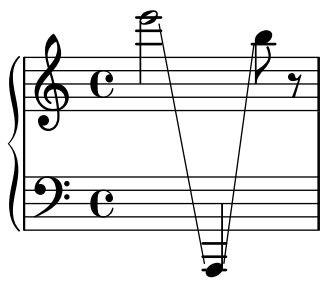
Un glissando relie une hauteur à une autre en passant par chaque hauteur intermédiaire. On l’obtient en accolant la commande `\glissando` à la première note.

```
\relative {
  g'2\glissando g'
  c2\glissando c,
  \afterGrace f,1\glissando f'16
}
```



Un glissando peut intervenir au moment d’un changement de portée :

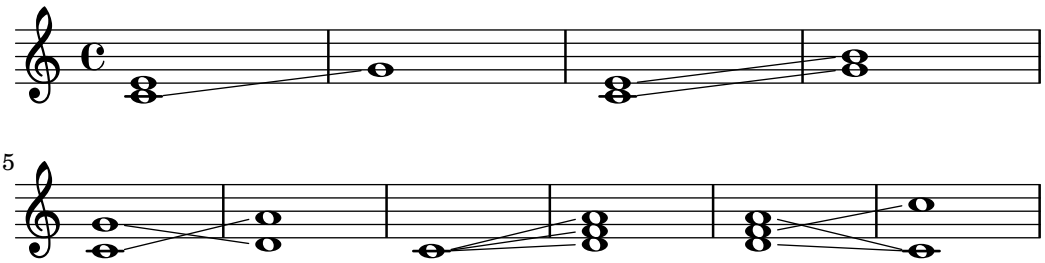
```
\new PianoStaff <<
  \new Staff = "right" {
    e'''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>
```



Un glissando peut affecter des notes d’un accord. En dehors du cas où les notes des deux accords sont reliées directement l’une à l’autre, les relations s’établissent à l’aide de la commande `\glissandoMap` ; les notes d’un accord sont numérotées à partir de zéro et dans leur ordre d’apparition dans le fichier `.ly`.

```
\relative {
  <c' e>1\glissando g' |
```

```
  <c, e>1\glissando |
  <g' b> |
  \break
  \set glissandoMap = #'((0 . 1) (1 . 0))
  <c, g'>1\glissando |
  <d a'> |
  \set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
  c1\glissando |
  <d f a> |
  \set glissandoMap = #'((2 . 0) (1 . 0) (0 . 1))
  <f d a'>1\glissando |
  <c c'> |
}
```
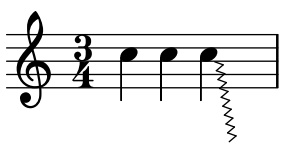


Un glissando est indiqué graphiquement, par une ligne ou des vaguelettes – voir Section 5.4.8 [Styles de ligne], page 682.

Morceaux choisis

Glissando contemporain

De nos jours, il peut arriver que la note d'arrivée d'un glissando soit absente de la partition. Pour ce faire, il vous faudra utiliser une cadence et « masquer » la note d'arrivée.

```
\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
  \bar "|"
}
```



Ajout de marques temporelles à un long glissando

Lorsqu'un glissando s'étend dans la durée, on trouve parfois des indications temporelles, matérialisées par des hampes sans tête de note. De telles hampes permettent aussi d'indiquer des éléments intermédiaires.

L'alignement des hampes avec la ligne de glissando peut requérir quelques aménagements.
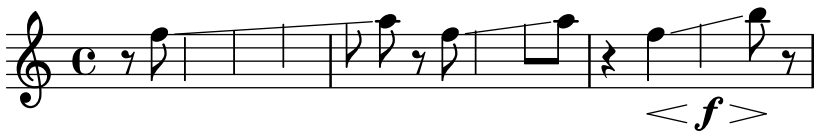
```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

glissandoSkipOff = {
  \revert NoteColumn.glissando-skip
  \undo \hide NoteHead
  \revert NoteHead.no-ledgers
}

\relative c'' {
  r8 f8\glissando
  \glissandoSkipOn
  f4 g a a8\noBeam
  \glissandoSkipOff
  a8

  r8 f8\glissando
  \glissandoSkipOn
  g4 a8
  \glissandoSkipOff
  a8 |

  r4 f\glissando \<
  \glissandoSkipOn
  a4\f \>
  \glissandoSkipOff
  b8\! r |
}
```



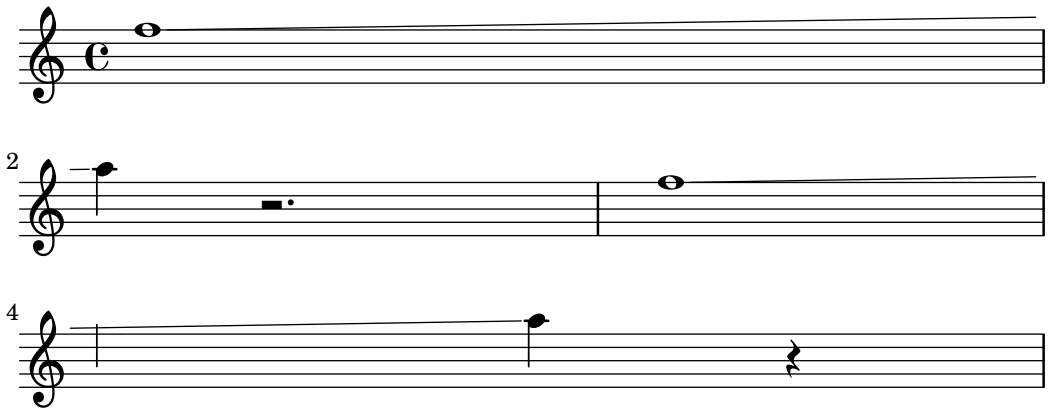
Saut de ligne et glissando

L'affectation de la valeur `#t` à la propriété `breakable`, combinée à `after-line-breaking`, permet la rupture d'une indication de glissando lors d'un saut de ligne.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

\relative c'' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
```

```
  a4 r2. |
  f1\glissando
  \once \glissandoSkipOn
  \break
  a2 a4 r4 |
}
```



Rappel du glissando à l'occasion d'une alternative

Un glissando qui se prolonge sur plusieurs sections `\alternative` peut se rappeler à l'aide d'une note d'ornement supplémentaire et masquée, à laquelle sera attaché le départ du glissando, ce dans chaque bloc `\alternative`. Cette note d'ornement devrait avoir la même hauteur que la note où commençait le glissando originel. Ceci est géré par une fonction musicale qui prendra en argument la hauteur de la note d'ornement.

Dans le cadre d'un musique polyphonique, il ne faudra pas oublier d'ajouter une note d'ornement dans toutes les autres voix afin de préserver la synchronisation.

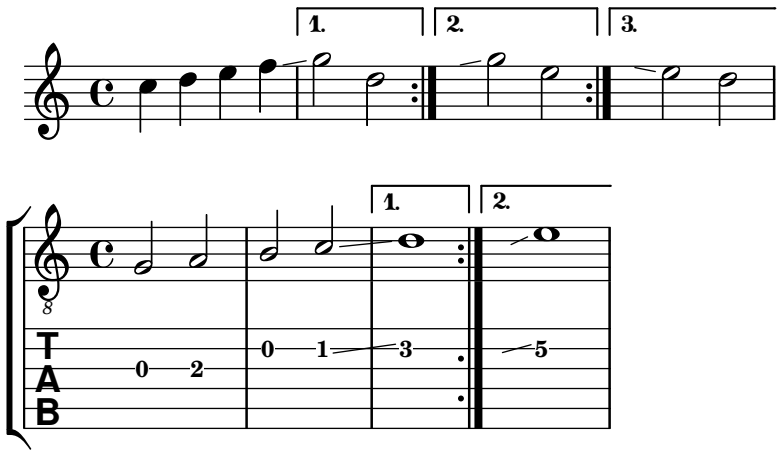
```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music =  \relative c' {
```

```
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \new Voice { \clef "G_8" \music }
    >>
    \new TabStaff  <<
      \new TabVoice { \clef "moderntab" \music }
    >>
  >>
}
```



Voir aussi

Glossaire musicologique : Section “glissando” dans *Glossaire*.

Manuel de notation : Section 5.4.8 [Styles de ligne], page 682.

Morceaux choisis : Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Glissando” dans *Référence des propriétés internes*.
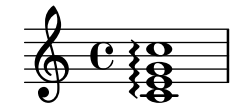
Problèmes connus et avertissements

Il n’est pas possible d’imprimer un texte (tel que *gliss.*) le long de la ligne de glissando.

Arpèges

On peut indiquer qu’un accord doit être arpégé en lui accolant la commande `\arpeggio` :

```
\relative { <c' e g c>1\arpeggio }
```



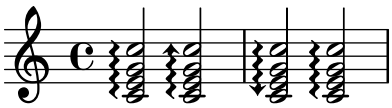
LilyPond dispose de différents graphismes pour indiquer un arpège ; `\arpeggioNormal` reviendra au style par défaut.

```
\relative {
  <c' e g c>2\arpeggio

  \arpeggioArrowUp
  <c e g c>2\arpeggio

  \arpeggioArrowDown
  <c e g c>2\arpeggio

  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Ces commandes prédéfinies modifient en interne la propriété `'arpeggio-direction` ; leur définition complète se trouve dans le fichier `ly/property-init.ly`.

Des crochets indiquent que l'accord devra être plaqué et non arpégé :

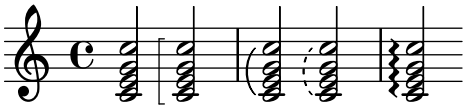
```
\relative {
  <c' e g c>2

  \arpeggioBracket
  <c e g c>2\arpeggio

  \arpeggioParenthesis
  <c e g c>2\arpeggio

  \arpeggioParenthesisDashed
  <c e g c>2\arpeggio

  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Ces commandes prédéfinies apportent une dérogation à la propriété `'stencil` de l'objet `Arpeggio` et peuvent aussi adapter son `'X-extent` (en la matière son étendue horizontale, de telle sorte qu'il n'entre pas en collision avec d'autres objets).

Les indications d'arpeggio peuvent se présenter sous la forme de ligne discontinue à l'aide de la propriété `'dash-definition`. Pour plus de détails à ce propos, consultez [Liaisons d'articulation], page 139.

Un arpège peut parfois s'écrire de manière explicite, à l'aide de liaisons de tenue. Pour plus d'information, voir [Liaisons de prolongation], page 57.

Commandes prédéfinies

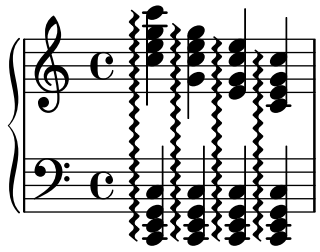
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`.

Morceaux choisis

Arpège distribué sur une partition pour piano

Dans une double portée pour piano (`PianoStaff`), un arpège peut s'étendre sur les deux portées grâce à la propriété `PianoStaff.connectArpeggios`.

```
\new PianoStaff \relative c'' <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {
    <c e g c>4\arpeggio
    <g c e g>4\arpeggio
    <e g c e>4\arpeggio
    <c e g c>4\arpeggio
  }
  \new Staff {
    \clef bass
    \repeat unfold 4 {
      <c,, e g c>4\arpeggio
    }
  }
>>
```

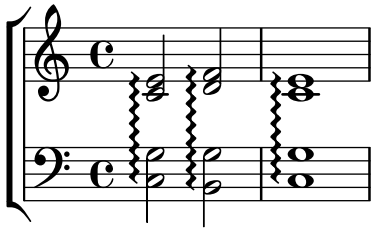


Arpège distribué pour un autre contexte que le piano

Il est possible de distribuer un arpège sur plusieurs portées d'un système autre que le `PianoStaff` dès lors que vous incluez le `Span_arpeggio_engraver` au contexte `Score`.

```
\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
```
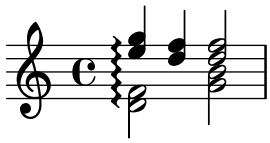
```
  }
  \layout {
    \context {
      \Score
      \consists "Span_arpeggio_engraver"
    }
  }
}
```



Arpège distribué sur plusieurs voix

Affecter le graveur `Span_arpeggio_engraver` au contexte de la portée (`Staff`) permet de distribuer un arpège sur plusieurs voix.

```
\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
    { <e' g>4\arpeggio <d f> <d f>2 }
    \\
    { <d, f>2\arpeggio <g b>2 }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “arpeggio” dans *Glossaire*.

Manuel de notation : [Liaisons d’articulation], page 139, [Liaisons de prolongation], page 57.

Fichiers d’initialisation : `ly/property-init.ly`.

Morceaux choisis : Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Arpeggio” dans *Référence des propriétés internes*, Section “PianoStaff” dans *Référence des propriétés internes*, Section “Slur” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les commandes prédéfinies telle que `\arpeggioArrowUp` s’appliquent uniquement au contexte en cours ; elles n’affecteront donc pas un arpège s’étendant sur d’autres voix ou portées. En pareil cas, ces commandes demandent à être utilisées au sein d’une section `\context` dans le bloc `\layout` ou bien avec une clause `\with` comme indiqué dans Section 5.1.5 [Modification

des réglages par défaut d'un contexte], page 638. Il peut être judicieux, au lieu d'utiliser ces raccourcis prédéfinis, d'introduire une dérogation aux propriétés concernées de l'objet `Arpeggio` directement dans le contexte approprié, comme par exemple

```
\override Staff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
```

pour que le crochet d'arpège couvre plusieurs voix au niveau `Staff`, ou

```
\override PianoStaff.Arpeggio.arpeggio-direction = #UP
```

pour obtenir un arpège à terminaison en flêche (pointant vers le haut) couvrant les portées d'un contexte `PianoStaff`.

Il est impossible de mêler au même instant, dans un contexte `PianoStaff`, des lignes d'arpèges connectées et d'autres non connectées.

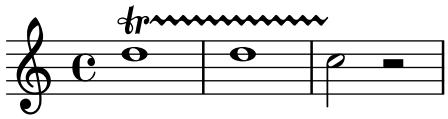
La manière simple de créer des lignes d'arpège sous forme de parenthèse n'est pas opérationnelle pour des arpèges inter-portées ; voir [Lignes de changement de portée], page 360.

Trilles

Les trilles brefs s'indiquent comme n'importe quelle ponctuation, avec un simple `\trill` ; voir [Articulations et ornements], page 127.
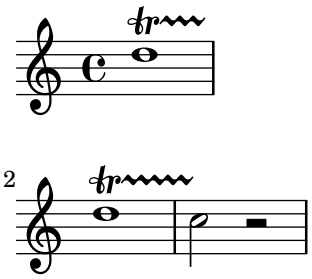
Les trilles plus longs sont délimités par `\startTrillSpan` et `\stopTrillSpan` :

```
\relative {
  d''1\startTrillSpan
  d1
  c2\stopTrillSpan r2
}
```



Lorsqu'un saut de ligne intervient alors qu'une prolongation de trille est présente, l'indication de trille et sa prolongation sont rappelées sur la première note de la nouvelle ligne :

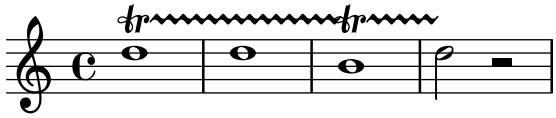
```
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan r2
}
```



Lorsque des trilles interviennent sur une succession de hauteurs différentes, point n'est besoin d'expliciter la commande `\stopTrillSpan` puisque l'apparition d'un nouveau trille interrompt de fait celui qui le précédait :
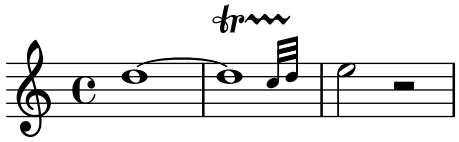
```
\relative {
  d''1\startTrillSpan
```

```
  d1
  b1\startTrillSpan
  d2\stopTrillSpan r2
}
```



Dans l'exemple suivant, un trille se combine avec des notes d'ornement. La syntaxe d'une telle construction ainsi que le moyen de positionner les notes d'ornement avec précision est expliquée au chapitre [Notes d'ornement], page 119.

```
\relative {
  d''1~\afterGrace
  d1\startTrillSpan { c32[ d]\stopTrillSpan }
  e2 r2
}
```



Les trilles qui font intervenir une hauteur précise peuvent être indiqués par la commande `pitchedTrill`. Le premier argument est la note réelle ; le second est une hauteur qui sera imprimée comme une tête de note noire entre parenthèses.

```
\relative {
  \pitchedTrill
  e''2\startTrillSpan fis
  d2 c2\stopTrillSpan
}
```
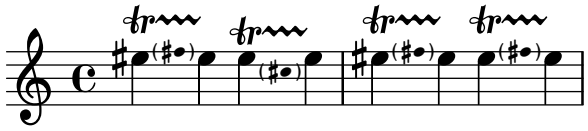


L'altération de cette hauteur explicite sera indiquée automatique pour le premier trille d'une mesure, même s'il s'agit d'un bécarre.

```
{
  \key d \major
  \pitchedTrill
  d'2\startTrillSpan cis d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan c d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan e d\stopTrillSpan
}
```



L'impression de l'altération (sur la même note dans la même mesure) devra être forcée en ajoutant un `!` à la note considérée.

```
\relative {
  \pitchedTrill
  eis''4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan cis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis!
  eis4\stopTrillSpan
}
```



Commandes prédéfinies

`\startTrillSpan`, `\stopTrillSpan`.

Voir aussi

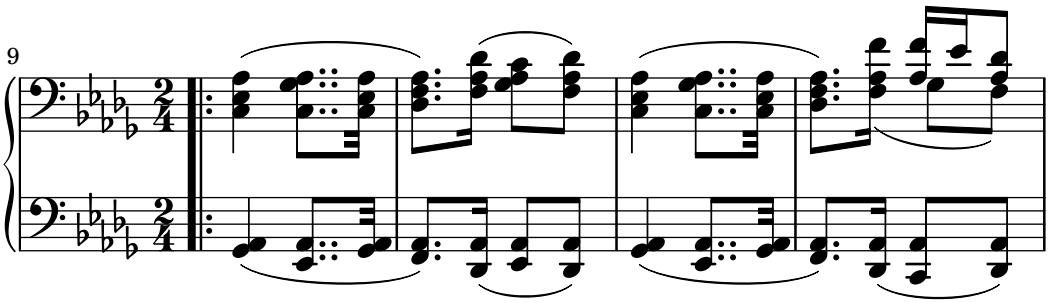
Glossaire musicologique : Section "trille" dans *Glossaire*.
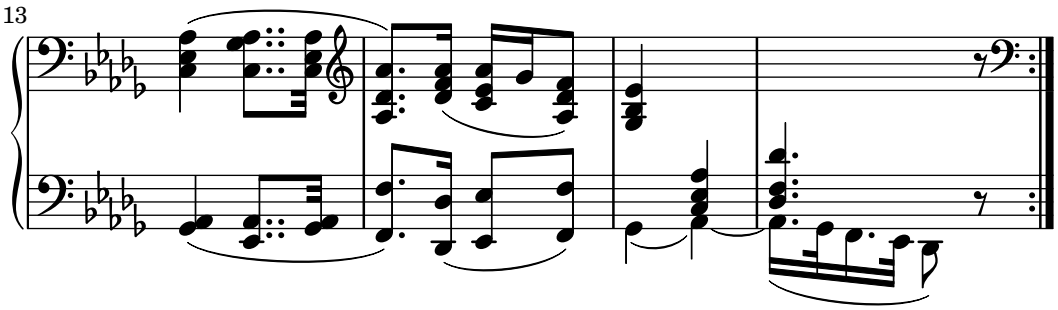
Manuel de notation : [Articulations et ornements], page 127, [Notes d'ornement], page 119.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TrillSpanner" dans *Référence des propriétés internes*.

1.4 Répétitions et reprises





La répétition est une notion essentielle en musique, et il existe de nombreuses façons de mettre en œuvre et noter ce concept. LilyPond prend en charge les types de répétition suivants :

`volta` Il s'agit ici de la notation courante des reprises avec ou sans fins alternatives. Le passage répété est encadré par des barres de reprise. Lorsque la répétition commence au début de la pièce, aucune barre de reprise n'est gravée en début de partition. Les fins alternatives s'impriment en séquence, avec un crochet et un numéro de *volta*.

`unfold` La musique répétée est développée dans la partition autant de fois qu'indiqué.

`percent` Des barres obliques ou signes de pourcentage indiquent la répétition de temps ou de mesures.

`tremolo` Ce type permet de réaliser des trémolos sous forme de liens de croches.

Les constructions d'accord peuvent se répéter à l'aide du symbole spécifique `q` – voir [Répétition d'accords], page 175.

1.4.1 Répétition d'un long passage

Cette section présente la syntaxe des répétitions longues.

Répétitions explicites

Adjoindre à la commande `\repeat` l'opion `unfold` permet de s'affranchir de ressaisir de la musique répétitive. La syntaxe est identique à celle de la commande `\repeat volta` abordée plus avant.

Aux fins d'éviter les redites, l'expansion n'est ici pas abordée en détail. Certains exemples des paragraphes qui suivent illustrent cependant des fonctionnalités à la fois des formes `volta` et `unfold` ainsi que de la commande `\unfoldRepeats` qui permet de convertir la forme `volta` en une forme `unfold`. Un autre sujet d'intérêt de la commande `unfoldRepeats` est abordé dans Section 3.5.6 [Gestion des répétitions en MIDI], page 563.

Dans certains cas, et tout particulièrement dans un contexte `\relative`, la fonction `\repeat unfold` ne revient pas à écrire littéralement la même expression musicale plusieurs fois. Ainsi :

```
\repeat unfold 2 { a'4 b c }
```

introduit un saut d'octave, contrairement à

```
a'4 b c |
a'4 b c |
```

Vous pouvez facilement imbriquer plusieurs fonctions `\repeat unfold`, ce qui n'est pas aussi simple avec `\repeat volta`.

> **Note :** L'insertion d'un `\relative` dans une section `\repeat` sans déclaration explicite du contexte `Voice` génèrera une portée supplémentaire – voir Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Voir aussi

Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

Référence des propriétés internes : Section “RepeatedMusic” dans *Référence des propriétés internes*, Section “UnfoldedRepeatedMusic” dans *Référence des propriétés internes*.
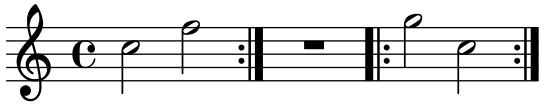
Répétitions simples

Les reprises courantes, sans alternative, s’indiquent comme ceci :

```
\repeat volta nombre_de_fois expression_musicale
```
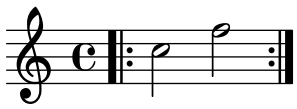
où *expression_musicale* représente ce qui doît être répété.

```
\fixed c'' {
  \repeat volta 2 { c2 f }
  R1
  \repeat volta 2 { g2 c }
}
```
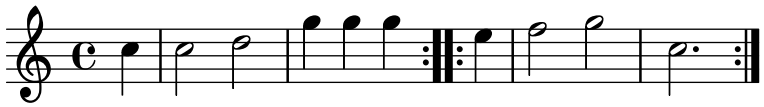


Aucun « début » de reprise n’est indiqué par défaut pour la première mesure d’un morceau. Vous pouvez cependant ajouter une barre de reprise avec un `\bar ".|:"` avant la première note.

```
\fixed c'' {
  \repeat volta 2 { \bar ".|:" c2 f }
}
```



Lorsqu’une reprise sans fin alternative débute au milieu d’une mesure, elle devrait se terminer aussi au milieu d’une mesure, de telle sorte que les mesures soient complètes. En pareil cas, les indications de reprise ne constituent pas des barres de mesure à proprement parler ; il n’est donc pas nécessaire de faire appel à la commande `\partial` ou à des contrôles d’intégrité de mesure. Le recours à l’instruction `\partial` est toutefois nécessaire lorsque la reprise est au début d’une pièce commençant par une levée afin de respecter cette entame.

```
\fixed c'' {
  \partial 4
  \repeat volta 2 {
    c4
    c2 d
    g4 g g
  }
  \repeat volta 2 {
    e4
    f2 g
    c2.
  }
}
```
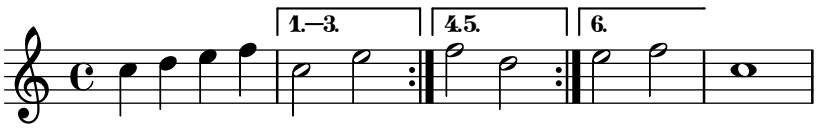


Fins alternatives

Une répétition avec fins alternatives s'indique ainsi :

```
\repeat volta nombre_de_fois expression_musicale
\alternative {
  \volta liste_de_nombres expression_musicale
  \volta liste_de_nombres expression_musicale
  ...
}
```

où *liste_de_nombres* est une liste de nombres, séparés par des virgules, indiquant les occurrences à répéter, de 1 à *nombre_de_fois*.

```
\fixed c'' {
  \repeat volta 6 { c4 d e f }
  \alternative {
    \volta 1,2,3 { c2 e }
    \volta 4,5 { f2 d }
    \volta 6 { e2 f }
  }
  c1
}
```



Si l'on donne trop peu d'alternatives en regard du nombre de fois où le passage doit être rejoué, la première alternative sera jouée plusieurs fois.

Note : Tout élément inclus dans un bloc `\alternative` sera traîté en tant qu'alternative. Quelque chose d'aussi simple qu'un contrôle de mesure placé du mauvais côté d'une accolade peut générer des résultats inattendus.

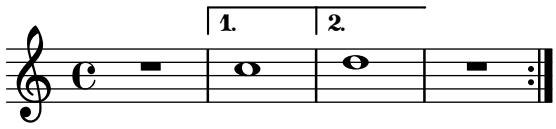
Note : Une clause `\relative` ne doit jamais se trouver à l'intérieur d'une section `\repeat` : vous aurez immanquablement des portées parasites. Voir Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Autres aspects des fragments répétés

Un bloc `\alternative` peut prendre place à l'intérieur d'un bloc `\repeat` afin de produire une notation comparable à des alternatives. Il ne saurait toutefois s'utiliser ainsi pour signifier les alternatives réelles – voir [Fins alternatives], page 159.

```
\fixed c'' {
  \repeat volta 2 {
    R1
    \alternative {
      \volta 1 { c1 }
      \volta 2 { d1 }
    }
```

```
    R1
  }
}
```

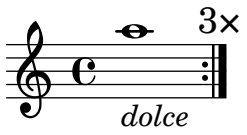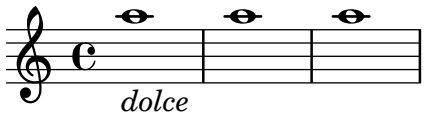


La présence d'une commande `\volta` n'est pas réservée aux blocs `\alternative`. Elle peut s'utiliser n'importe où dans un bloc `\repeat` pour désigner un fragment qui concerne des reprises particulières. Si le fragment spécifique à une reprise a une durée, celle-ci sera par défaut imprimée sous un crochet comme s'il s'agissait d'un bloc `\alternative` ; ce crochet peut être effacé – voir Section 5.4.7 [Visibilité des objets], page 675, – ou personnalisé pour répondre au besoin.

Lorsqu'un bloc `\repeat` est expansé, le fragment spécifique à une occurrence sera omis pour toutes celles auxquelles il ne s'applique pas. Fournir une liste Scheme vide au lieu du nombre de fois supprime carrément la musique.

```
music = \repeat volta 3 {
  \volta 1 { s1*0_\markup { \italic dolce } }
  a''1
  \volta #'() { \mark "3×" }
}

\new Score { \music }
\new Score { \unfoldRepeats \music }
```

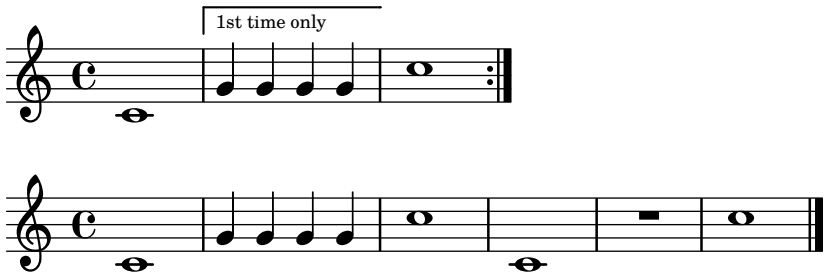




Lorsqu'un bloc `\repeat` est expansé, il peut être souhaitable de non seulement filtrer les fragments spécifiques, mais aussi d'ajouter de la musique qui serait absente de la forme `\volta`. La commande `\unfolded` permet de désigner la musique qui sera ignorée jusqu'à l'expansion du bloc `\repeat` qui la contient.

```
music = \fixed c' {
  \repeat volta 2 {
    c1
    \once \override Score.VoltaBracket.text = "1st time only"
    \once \override Score.VoltaBracket.font-name = "TeX Gyre Schola"
    <<
      \volta 1 { g4 g g g }
      \volta 2 { \unfolded { R1 } }
    >>
    c'1
    \volta 2 { \unfolded { \bar "|." } }
  }
}
```

```
\new Score { \music }
\new Score { \unfoldRepeats \music }
```
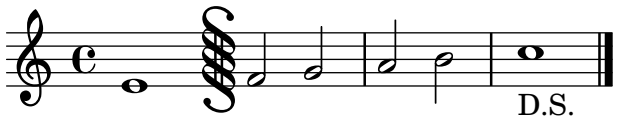


Note : Les commandes `\volta` et `\unfolded` fontionnent en respectant la répétition la plus imbriquée qui les entoure.

Barre de *segno*

La commande `\inStaffSegno` permet de générer une barre de mesure composite par l'adjonction d'un symbole de *segno* à une barre de reprise créée par une commande `\repeat volta`. Qu'il s'agisse d'un début, d'une fin ou d'une double reprise, le type de barre est automatiquement sélectionné. L'indication « D.S. » devra cependant être ajouté manuellement.
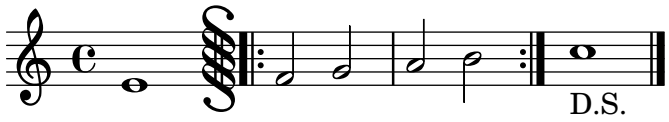
En dehors de toute reprise :

```
\relative {
  e'1
  \inStaffSegno
  f2 g a b
  c1_"D.S." \bar "|."
}
```



Au début d'une reprise :

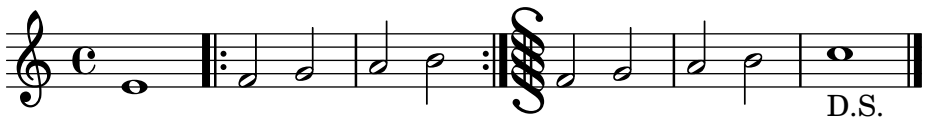
```
\relative {
  e'1
  \repeat volta 2 {
    \inStaffSegno  % start repeat
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```



En fin de reprise :

```
\relative {
  e'1
```

```
  \repeat volta 2 {
    f2 g a b
    \inStaffSegno  % end repeat
  }
  f2 g a b
  c1_"D.S." \bar "|."
}
```



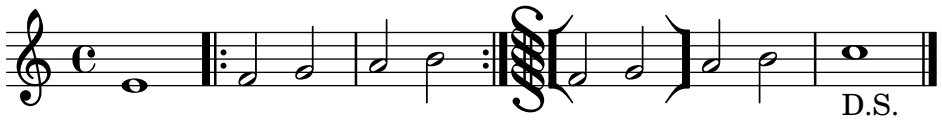
Entre deux reprises :

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }
  \inStaffSegno  % double repeat
  \repeat volta 2 {
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```



Des symboles alternatifs de barre de mesure sont aussi accessibles, dans un contexte `Score`, à l'aide des propriétés `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` ou `doubleRepeatSegnoType` selon les besoins. Ces types de barre alternative doivent être choisis parmi les types prédéfinis ou préalablement créés à l'aide d'une commande `\defineBarLine` – voir [Barres de mesure], page 102.

```
\defineBarLine ":|.S[" #'(":|." "S[" "")
\defineBarLine "]" #'("]" "" "")
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \once \set Score.endRepeatSegnoType = ":|.S["
    \inStaffSegno
  }
  f2 g \bar "]" a b
  c1_"D.S." \bar "|."
}
```
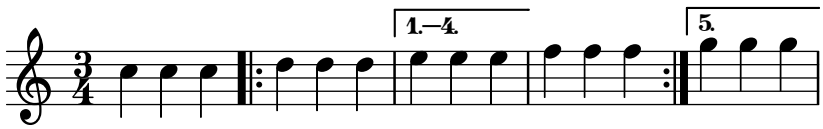


Morceaux choisis

Diminution de la taille du crochet d'alternative

Les crochets indiquant les fins alternatives s'étalent tout au long de celles-ci. On peut les raccourcir en jouant sur la propriété `voltaSpannerDuration`. Dans l'exemple suivant, le crochet ne se prolonge que sur une mesure à 3/4.

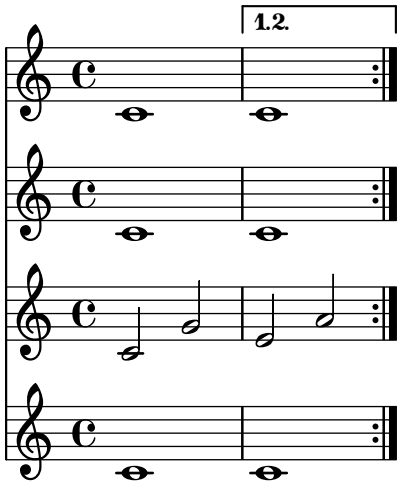
```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3/4)
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```



Ajout du crochet de reprise à d'autres portées

D'ordinaire, le graveur `Volta_engraver` réside dans le contexte `Score` ; les crochets précédant une reprise s'impriment donc seulement au-dessus de la portée du haut. On peut ajuster cela en déplaçant ce graveur vers les contextes de portée (`Staff`) qui doivent comporter ces crochets.

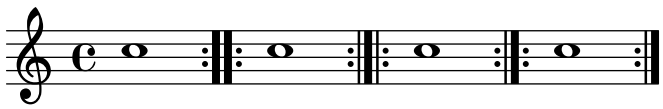
```
<<
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



Succession de reprises et style de barre par défaut

LilyPond dispose de trois différents styles de barre pour indiquer une succession de reprises. Vous devez opter pour un style par défaut, à l'aide de la propriété `doubleRepeatType`.
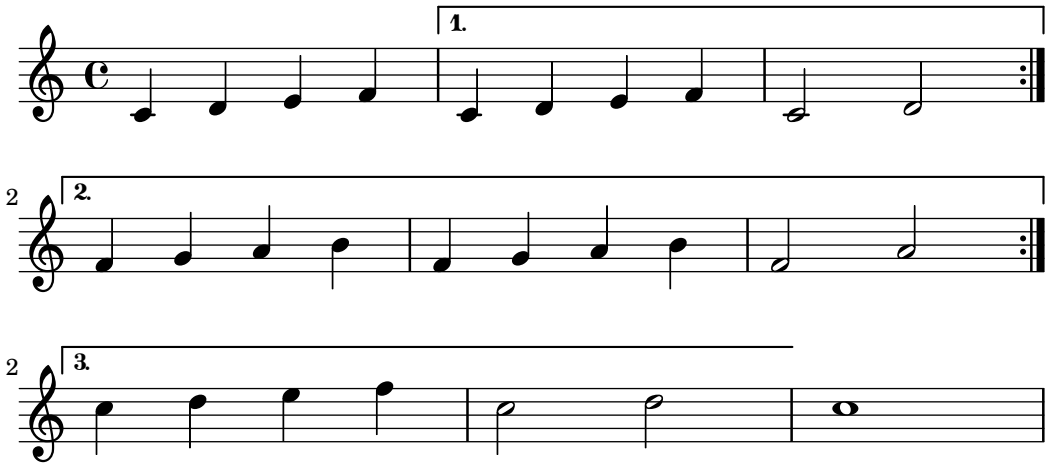
```
\relative c'' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":..:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.:"
  \repeat volta 1 { c1 }
}
```
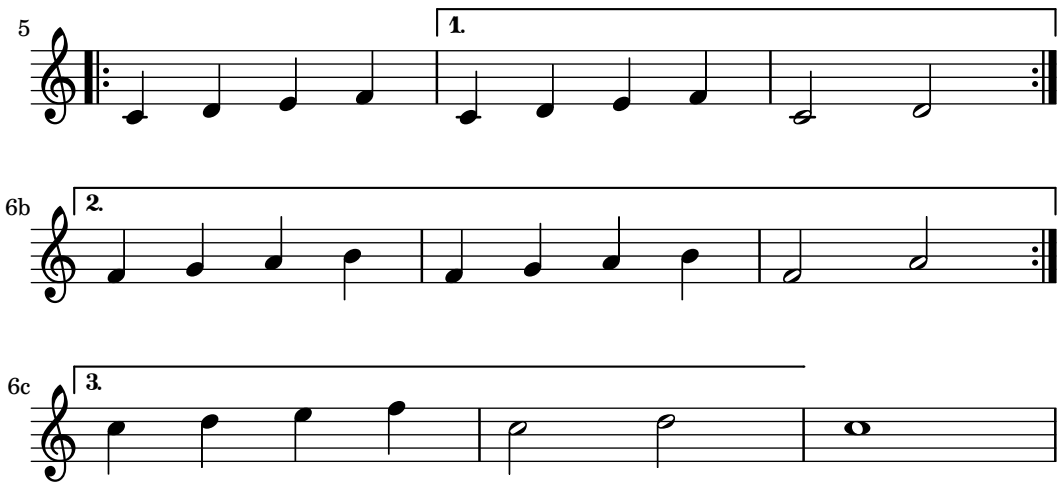


Numérotation des mesures et alternatives

Deux méthodes alternatives vous permettent de gérer la numérotation des mesures en cas de reprises.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```





Voir aussi

Glossaire musicologique : Section “répétition” dans *Glossaire*, Section “volta” dans *Glossaire*.

Manuel de notation : [Barres de mesure], page 102, [Gestion du temps], page 125, Section 5.1.4 [Modification des greffons de contexte], page 636, [Modification des liaisons], page 690.

Fichiers d’initialisation : `ly/engraver-init.ly`.

Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

Référence des propriétés internes : Section “VoltaBracket” dans *Référence des propriétés internes*, Section “RepeatedMusic” dans *Référence des propriétés internes*, Section “VoltaRepeatedMusic” dans *Référence des propriétés internes*, Section “UnfoldedRepeatedMusic” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L’extension d’une liaison, dans la forme `\volta`, n’est possible que pour la première alternative. Selon le même principe, une liaison ne saurait partir de la fin d’une alternative pour se terminer au début de la reprise.

L’aspect visuel d’une liaison se continuant dans les autres alternatives, peut être simulé à l’aide de la commande `\repeatTie` lorsqu’elle s’arrête sur sa première note – méthode qui cependant ne fonctionne pas pour un `TabStaff` ; voir [Liaisons de prolongation], page 58. D’autres moyens existent pour indiquer la prolongation d’une liaison sur des alternatives, y compris dans un `TabStaff`, en suivant les préceptes donnés à la rubrique [Modification des liaisons], page 690.

L’aspect visuel d’un glissando se continuant dans les autres alternatives peut être simulé à l’aide d’un glissando partant d’une note d’ornement supplémentaire et masquée. Un exemple se trouve à la rubrique Voir [Glissando], page 149.

Le développement, à l’aide de la commande `\unfoldRepeats`, d’une répétition qui commence sur une mesure incomplète et contient un bloc `alternative` avec modification de la propriété `measureLength` entraînera des messages d’erreur concernant le placement des barres de mesure.

Des reprises imbriquées telles que

```
\repeat ...
\repeat ...
\alternative
```

présentent une ambiguïté, dans la mesure où l’on ne sait à quelle section `\repeat` attribuer la section `\alternative`. Pour résoudre cette ambiguïté, il convient de toujours insérer la commande `\alternative` à l’intérieur de la section `\repeat`. Il est préférable, dans une telle situation, d’utiliser des accolades pour plus de clarté.

Un bloc `\alternative` clôturé à l’intérieur même du bloc `\repeat` – plutôt que venant après – ne produira pas les barres de reprise attendues à la fin des alternatives.

Indications de reprise manuelles
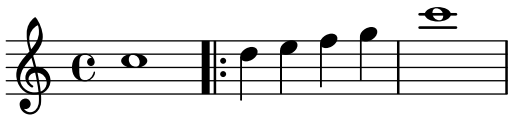
> **Note :** Les méthodes présentées dans les lignes qui suivent ne devraient servir à indiquer que des constructions de répétition inhabituelles. En règle générale, il vaut mieux recourir à la fonction `\repeat` pour créer une reprise ou bien insérer la barre de mesure adéquate. Pour plus d'information, voir le chapitre [Barres de mesure], page 102.

La propriété `repeatCommands` sert à contrôler la mise en forme des reprises. On la définit par une suite de commandes de reprise Scheme.

`start-repeat`

Pour imprimer une barre de reprise `.|:`

```
\relative {
  c''1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
  c1
}
```
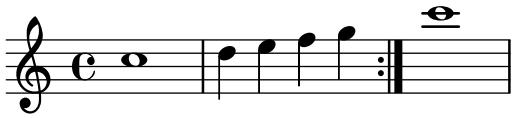


Traditionnellement, on n'imprime pas de signe de reprise en début de morceau.

`end-repeat`

Pour imprimer une barre de reprise `:|.`
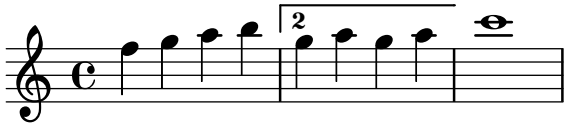
```
\relative {
  c''1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
  c1
}
```



`(volta` *nombre*`) ... (volta #f)`

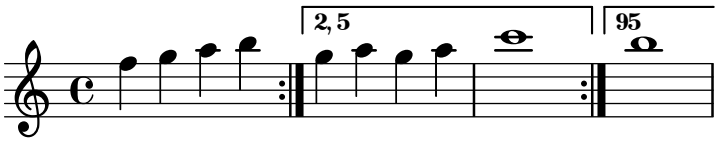
Pour obtenir un crochet indiquant le numéro de l'alternative. Pour que le crochet s'imprime effectivement, il faut spécifier explicitement l'endroit où il doit se terminer.

```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2"))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}
```
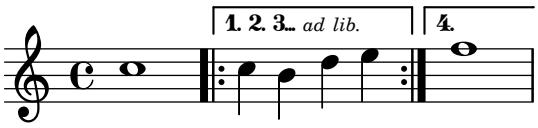


Plusieurs commandes de reprise peuvent intervenir au même moment :

```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2, 5") end-repeat)
  g4 a g a
  c1
  \set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
  b1
  \set Score.repeatCommands = #'((volta #f))
}
```



Le crochet indiquant une alternative peut contenir aussi du texte. Il peut s'agir d'un ou plusieurs nombres ou bien d'une indication textuelle (*markup*) – voir Section 1.8.2 [Mise en forme du texte], page 263. Le plus simple, dans le cas d'une indication textuelle, est de tout d'abord définir ce *markup*, puis de l'inclure dans une liste Scheme.

```
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative {
  c''1
  \set Score.repeatCommands =
    #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
```



Voir aussi

Manuel de notation : [Barres de mesure], page 102, Section 1.8.2 [Mise en forme du texte], page 263.

Morceaux choisis : Section "Répétitions" dans *Morceaux choisis*.

Référence des propriétés internes : Section "VoltaBracket" dans *Référence des propriétés internes*, Section "RepeatedMusic" dans *Référence des propriétés internes*, Section "VoltaRepeatedMusic" dans *Référence des propriétés internes*.

1.4.2 Autres types de répétition

Nous abordons ici les reprises de courte durée. Il en existe deux formes, à savoir la répétition d'une même note sur quelques mesures – représentée par une barre oblique ou le signe pourcent – et les trémolos.

Répétitions de mesure

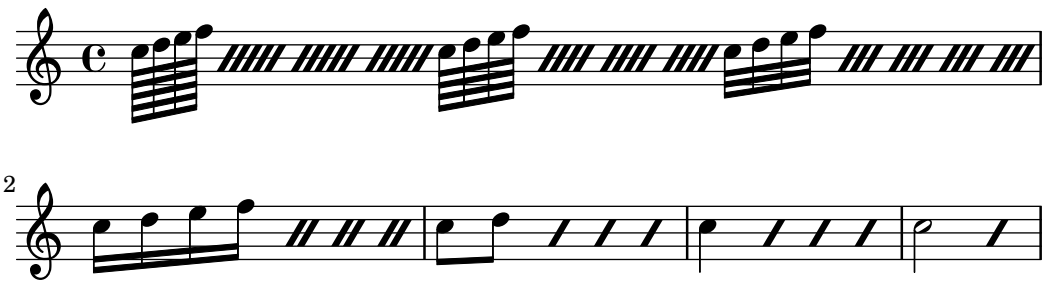
Le style de « reprise en pourcent » sert à répéter une séquence de notes. Elle sera imprimée une fois, puis remplacée par un symbole spécial.

En voici la syntaxe :

```
\repeat percent nombre expression_musicale
```
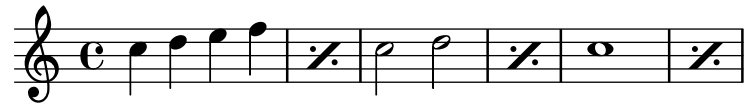
Les séquences inférieures à une mesure sont remplacées par une barre oblique.

```
\relative c'' {
  \repeat percent 4 { c128 d e f }
  \repeat percent 4 { c64 d e f }
  \repeat percent 5 { c32 d e f }
  \repeat percent 4 { c16 d e f }
  \repeat percent 4 { c8 d }
  \repeat percent 4 { c4 }
  \repeat percent 2 { c2 }
}
```
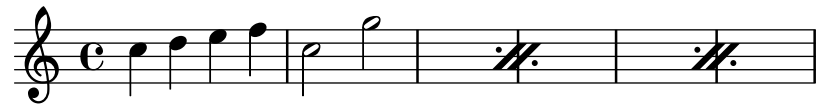


Les séquences d'une ou deux mesures sont remplacées par un symbole qui ressemble au symbole de pourcentage.

```
\relative c'' {
  \repeat percent 2 { c4 d e f }
  \repeat percent 2 { c2 d }
  \repeat percent 2 { c1 }
}
```

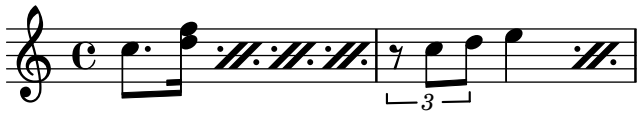


```
\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}
```



Les séquences inférieures à la mesure et qui contiennent des durées différentes sont remplacées par un double symbole de pourcentage.

```
\relative {
  \repeat percent 4 { c''8. <d f>16 }
  \repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
}
```

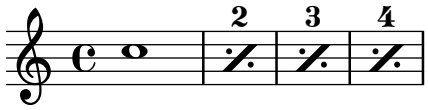


Morceaux choisis

Compteur de répétition en pourcent

Les répétitions de plus de deux mesures sont surmontées d'un compteur, si l'on active la propriété `countPercentRepeats` comme le montre l'exemple suivant :

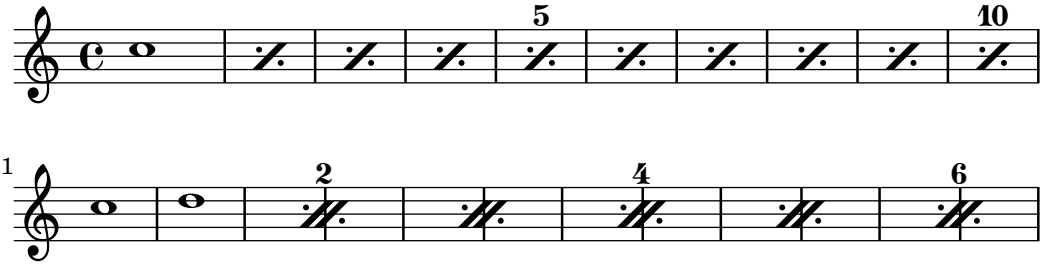
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



Affichage du numéro de répétition en pourcent

Le numéro de mesure répétée sera imprimé à intervalle régulier si vous déterminez la propriété de contexte `repeatCountVisibility`.

```
\relative c'' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
  \repeat percent 10 { c1 } \break
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
  \repeat percent 6 { c1 d1 }
}
```
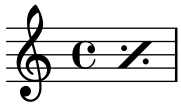


Répétition en pourcent isolée

Des symboles de pourcentage isolés peuvent aussi être obtenus au moyen d'un silence multi-mesure dont on modifie l'aspect :

```
makePercent =
#(define-music-function (note) (ly:music?)
   "Make a percent repeat the same length as NOTE."
   (make-music 'PercentEvent
               'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



Voir aussi

Glossaire musicologique : Section “percent repeat” dans *Glossaire*, Section “simile” dans *Glossaire*.
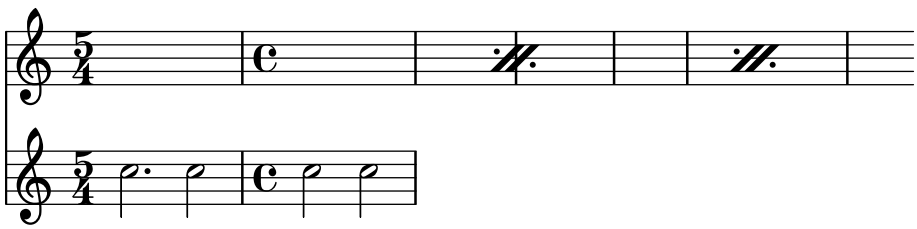
Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

Référence des propriétés internes : Section “RepeatSlash” dans *Référence des propriétés internes*, Section “RepeatSlashEvent” dans *Référence des propriétés internes*, Section “DoubleRepeatSlash” dans *Référence des propriétés internes*, Section “PercentRepeat” dans *Référence des propriétés internes*, Section “PercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatedMusic” dans *Référence des propriétés internes*, Section “Percent_repeat_engraver” dans *Référence des propriétés internes*, Section “DoublePercentEvent” dans *Référence des propriétés internes*, Section “DoublePercentRepeat” dans *Référence des propriétés internes*, Section “DoublePercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatedMusic” dans *Référence des propriétés internes*. Section “Double_percent_repeat_engraver” dans *Référence des propriétés internes*, Section “Slash_repeat_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

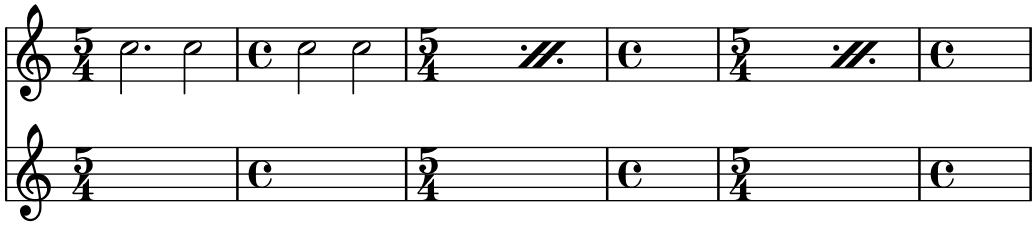
Les répétitions en pourcent ne peuvent contenir rien d’autre que le signe pourcent lui-même ; en particulier, les changements de métrique ne seront pas répété.

```
\repeat percent 3 { \time 5/4 c2. 2 \time 4/4 2 2 }
```



Tout changement de métrique ou insertion d’une commande `\partial` devra impérativement se traîter sous forme de construction parallèle, **en dehors** de la répétition en pourcentage :

```
<<
  \repeat percent 3 { c2. 2 2 2 }
  \repeat unfold 3 { \time 5/4 s4*5 \time 4/4 s1 }
>>
```

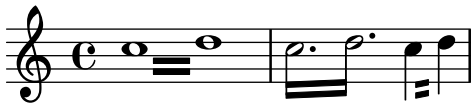


Répétitions en trémolo

Il y a deux formes de trémolo : la répétition alternative de deux notes ou accords, et la répétition rapide d’une seule note ou d’un accord. Lorsqu’il est constitué d’une alternance répétitive, le trémolo s’indique en ajoutant des barres de ligature entre les deux notes ou accords concernés. Lorsqu’il s’agit d’une répétition rapide, des barres de ligature penchées sont ajoutées à la note en question.

On peut placer une notation de trémolo entre deux notes, avec la commande `\repeat` suivie du style trémolo :

```
\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}
```
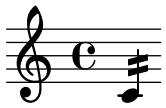


La syntaxe de `\repeat tremolo` requiert expressément deux notes encadrées par des accolades, et le nombre de répétitions exprimé en durée d'une note (pointée ou non). Ainsi, dans l'exemple ci-dessus, `\repeat tremolo 7` est valide car correspond à une note doublement pointée, à l'inverse de `\repeat tremolo 9`.

La durée d'un trémolo est égale à la durée de l'expression entre accolades multipliée par le nombre de fois à répéter : `\repeat tremolo 8 { c16 d16 }` correspond donc à la valeur d'une ronde, et sera représenté par deux rondes séparées par des barres de trémolo.
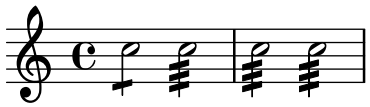
On peut indiquer de la même manière un trémolo sur une seule note, qu'il faudra alors laisser sans accolades :

```
\repeat tremolo 4 c'16
```



Le même résultat s'obtient en faisant suivre la note considérée de deux points et d'un nombre (`note:`*`nombre`*). Le nombre en question correspond à la valeur de la subdivision ; il doit être au moins de 8, auquel cas la hampe sera barrée par un seul trait de ligature. Si ce nombre est omis, la dernière valeur sera utilisée.

```
\relative {
  c''2:8 c:32
  c: c:
}
```
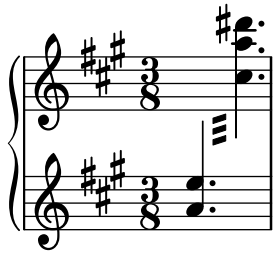


Morceaux choisis

Trémolo et changement de portée

Dans la mesure où `\repeat tremolo` requiert deux arguments musicaux pour un trémolo d'accords, la note ou l'accord de la portée opposée doit être encadré par des accolades et se voir adjoindre la commande `\change Staff`.

```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
```
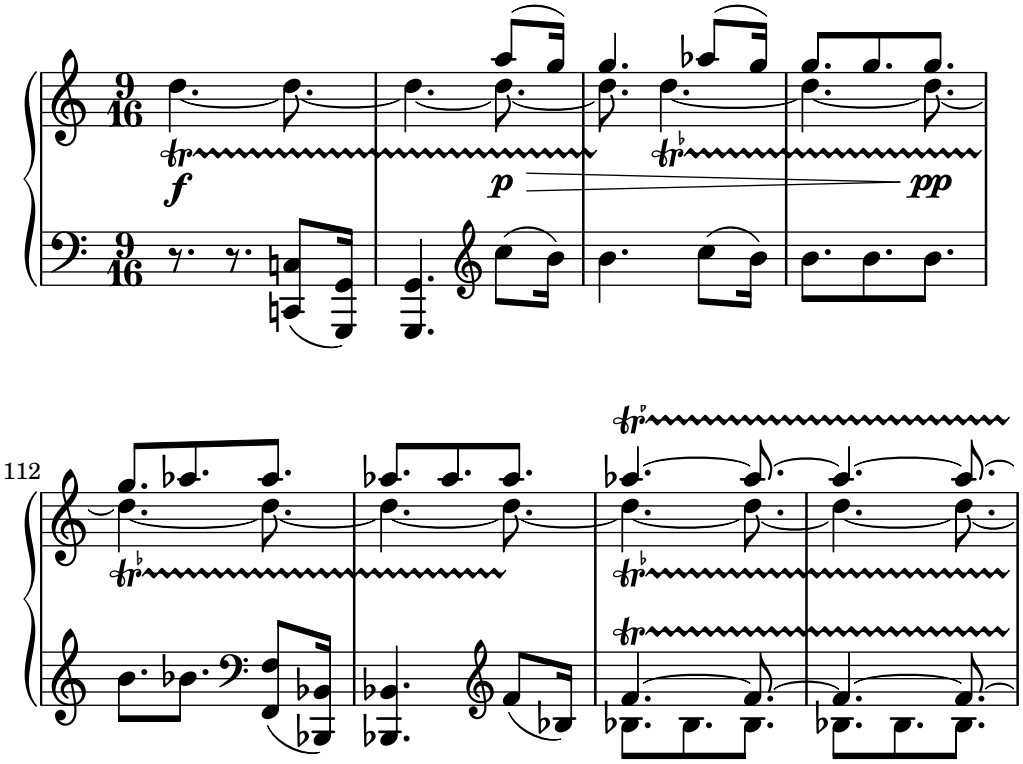
```
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```
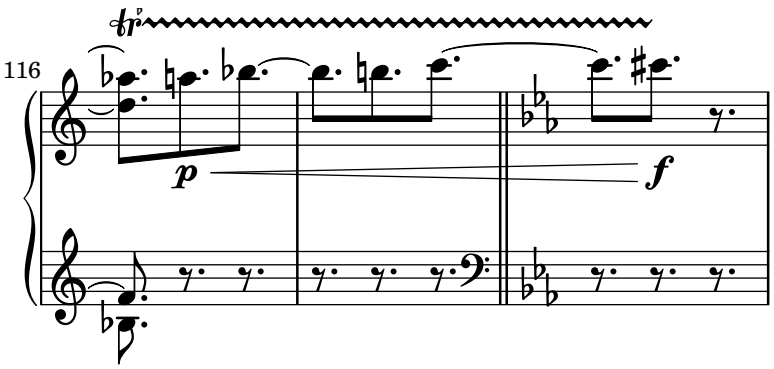


Voir aussi

Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

1.5 Notes simultanées





La notion musicale de polyphonie fait référence au fait d'avoir plus d'une voix simultanément dans une pièce. Dans LilyPond, la notion de polyphonie fait référence au fait d'avoir plus d'une voix sur la même portée.

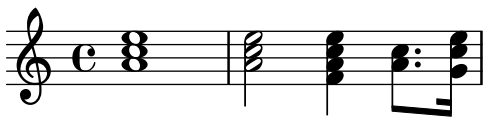
1.5.1 Monophonie

Nous allons voir ici comment gérer plusieurs notes simultanées dans une même voix.

Notes en accords

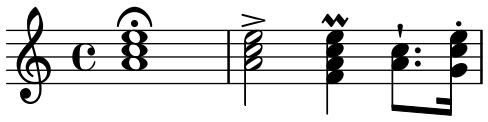
Un accord est formé en mettant une série de hauteurs entre < et >. Un accord peut être suivi d'une durée comme une simple note.

```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```



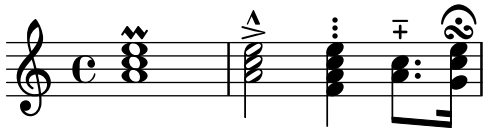
Un accord peut être suivi d'une indication d'articulation comme une simple note.

```
\relative {
  <a' c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
}
```



Certaines notes, au sein même d'un accord, peuvent être affectées d'une articulation ou d'un ornement :

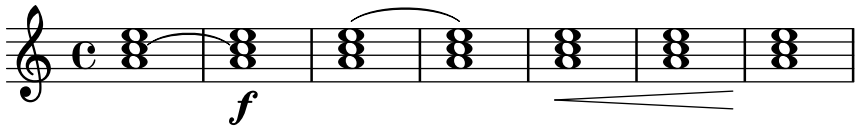
```
\relative {
  <a' c\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4
  <a-+ c-->8. <g\fermata c e\turn>16
}
```



Certains éléments de notation, tels que nuances et soufflets ne peuvent se rattacher qu'à un accord et non aux notes qui le composent, sous peine de ne les voir s'imprimer. D'autres
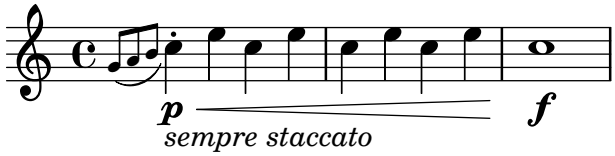
éléments, tels que doigtés ou liaisons, seront placés différemment selon qu'ils sont rattachés à des notes composant un accord, à un accord dans sa globalité ou à des notes individuelles.

```
\relative {
  <a'\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>)
  <a c e>\< <a c e> <a c e>\!
}
```



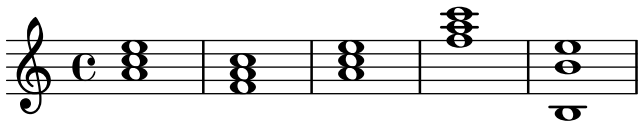
Un accord peut se voir comme un conteneur de notes, articulations et autres éléments rattachés. Par voie de conséquence, un accord dépourvu de note n'a pas de durée ; toute articulation qui lui serait attachée interviendra au même moment que la note ou l'accord qui le suit et lui sera donc combiné – pour des combinaisons plus complexes, voir [Expressions simultanées], page 177.

```
\relative {
  \grace { g'8( a b }
  <> ) \p \< -. -\markup \italic "sempre staccato"
  \repeat unfold 4 { c4 e }  c1\f
}
```



Les accords peuvent être saisis en mode relatif. Dans un accord, l'octave de chaque hauteur saisie est fonction de la précédente, à l'exception de la première qui, elle, sera positionnée en fonction de la première hauteur mentionnée dans l'accord précédent ou de la note individuelle précédente. Les autres notes au sein de l'accord se positionnent relativement à la précédente *dans cet accord*.

```
\relative {
  <a' c e>1 <f a c> <a c e> <f' a c> <b, e b,>
}
```



Pour plus d'information à propos des accords, voir Section 2.7 [Notation des accords], page 444.

Voir aussi

Glossaire musicologique : Section "accord" dans *Glossaire*.

Manuel d'initiation : Section "Combinaison de notes en accords" dans *Manuel d'initiation*.

Manuel de notation : [Articulations et ornements], page 127, Section 2.7 [Notation des accords], page 444. [Octaves relatives], page 2, Section 1.5.2 [Plusieurs voix], page 179.

Morceaux choisis : Section "Notes simultanées" dans *Morceaux choisis*.

Problèmes connus et avertissements

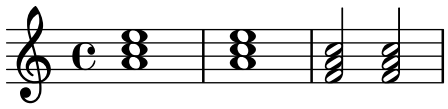
Un accord comportant plus de deux notes dans le même « espace de portée » – tel que '`<e f! fis!>`' – conduit immanquablement à des chevauchements. En fonction de la situation, un meilleur rendu peut nécessiter de recourir à

- l'utilisation temporaire de Section 1.5.2 [Plusieurs voix], page 179, '`<< f! \\ <e fis!> >>`',
- une transcription enharmonique d'une ou plusieurs hauteurs, '`<e f ges>`', ou
- des [Clusters], page 178.

Répétition d'accords

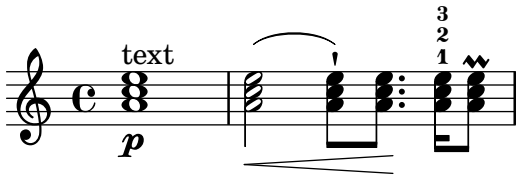
Dans le but de vous épargner de la saisie, LilyPond dispose d'un raccourci – symbolisé par la lettre `q` – qui a pour effet de répéter le dernier accord saisi :

```
\relative {
  <a' c e>1 q <f a c>2 q
}
```



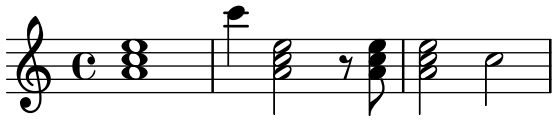
À l'instar de n'importe quel accord, le symbole de répétition peut être affublé d'une durée, de signes d'articulation, *markups*, liaisons, ligatures... En fait, c'est la structure du dernier accord qui est dupliquée.

```
\relative {
  <a' c e>1\p^"text" q2\<( q8)[-! q8.]\! q16-1-2-3 q8\prall
}
```



Dans la mesure où le symbole de répétition d'accord enregistre la structure du dernier accord construit, il est tout à fait possible de l'utiliser même après une succession de notes individuelles et de silences :
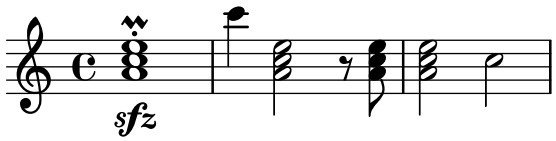
```
\relative {
  <a' c e>1 c'4 q2 r8 q8 |
  q2 c, |
}
```



Le symbole de répétition d'accord ne prend en charge que les hauteurs, en aucun cas les nuances, articulations ou ornementations, qu'elles aient été attachées aux notes le composant ou à l'ensemble.

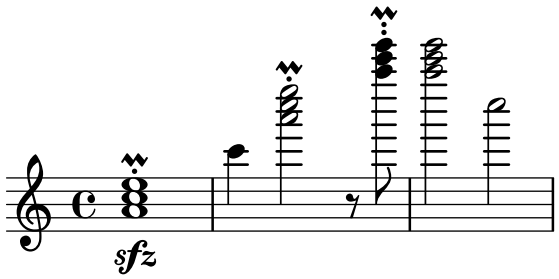
```
\relative {
  <a'-. c\prall e>1\sfz c'4 q2 r8 q8 |
  q2 c, |
```

```
}
```



Le seul moyen de les reproduire consiste à utiliser explicitement la fonction `\chordRepeats`, en lui adjoignant un argument supplémentaire qui recense les *types d'événement* à répéter et qui seraient absents de l'accord construit par un `q`.
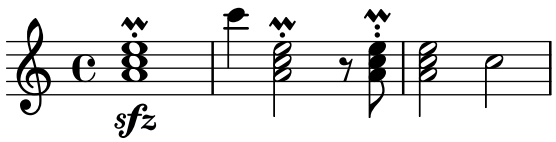
```
\relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c, |
}
```



Comme vous pouvez le constater, l'utilisation de `\chordRepeats` au sein d'un bloc `\relative` ne produit pas le résultat escompté : les événements de l'accord expansés sont identiques à la saisie traditionnelle d'un accord, ce qui a pour conséquence que l'octave affectée par `\relative` repose sur le contexte en cours.

Dans la mesure où l'imbrication de clauses `\relative` n'est pas source d'interférence, l'ajout d'un `\relative` à ce qui sera concerné par l'instruction `\chordRepeats` permet d'établir une relation d'octave entre les accords dès avant leur expansion. Dans le cas présent, l'intégralité du bloc `\relative` intérieur n'affecte en rien ce qui l'entoure, ce qui explique la spécification d'octave attachée à la dernière note :

```
\new Voice
\relative {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c'' |
}
```



Les interactions avec `\relative` ne se produisent que lors d'un appel explicite de `\chordRepeats` : l'expansion implicite en début de saisie intervient à un moment où toutes les instances de `\relative` ont déjà été interprétées.

Voir aussi

Manuel de notation : [Articulations et ornements], page 127, Section 2.7 [Notation des accords], page 444.
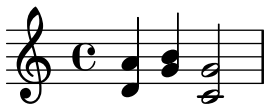
Fichiers d'initialisation : `ly/chord-repetition-init.ly`.

Expressions simultanées
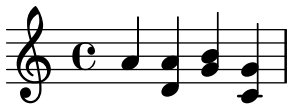
Lorsqu'une ou plusieurs expressions musicales sont encadrées par des doubles chevrons, elles sont considérées comme étant simultanées. Si la première expression débute par une note unique ou si l'intégralité de l'expression simultanée est explicitement rattachée à une voix en particulier, elle sera placée sur une seule portée. Dans le cas contraire, les éléments d'une expression simultanée seront placés sur des portées distinctes.

Voici deux exemples d'expression simultanée sur une même portée :

```
\new Voice {  % explicit single voice
  << \relative { a'4 b g2 }
     \relative { d'4 g c,2 } >>
}
```
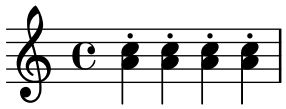


```
\relative {
  % single first note
  a' << \relative { a'4 b g }
        \relative { d'4 g c, } >>
}
```



Cette manière de procéder est utile lorsque les éléments de l'expression ont des rythmes identiques. Dès que vous tenterez d'attacher sur une même hampe des notes de durée différente, vous générerez des erreurs. Notes, articulations et modifications de propriétés au sein d'un même `Voice` sont enregistrées et gravées selon l'ordre musical :
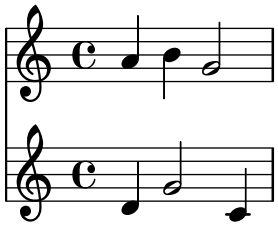
```
\relative {
  <a' c>4-.  <>-. << c a >>  << { c-. <c a> } { a s-. } >>
}
```



La présence de plusieurs hampes, ligatures, durées ou propriétés au même instant musical nécessite l'utilisation de plusieurs voix.

Dans l'exemple suivant, l'expression simultanée génère implicitement plusieurs portées :

```
% no single first note
<< \relative { a'4 b g2 }
   \relative { d'4 g2 c,4 } >>
```



En pareil cas, des rythmes différents ne sont source d'aucun problème puisqu'ils sont interprétés dans des voix différentes.

Problèmes connus et avertissements

Des notes, bien qu'appartenant à des voix différentes, mais dont les hampes ont la même orientation, peuvent se retrouver au même endroit sur la portée, ce quelque soit le décalage que vous auriez pu leur appliquer. Ceci ne manque pas de faire apparaître un message

```
warning: This voice needs a \voiceXx or \shiftXx setting

en français :
Avertissement : Cette voix requiert un voiceXx ou un réglage \shiftXx
```

lors de la compilation. Le déclenchement de cet avertissement peut être désactivé par une clause

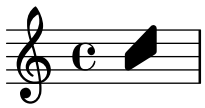
```
\override NoteColumn.ignore-collision = ##t
```

Ceci n'aura pas pour seule conséquence que ce message ne sera plus émis ; les procédures d'évitement de collision de quelqu'ordre que ce soit seront désactivées, ce qui peut conduire à quelques effets inattendus (voir aussi *Problèmes connus et avertissements* à la rubrique [Résolution des collisions], page 183).

Clusters

Un cluster indique un agrégat de sons. On peut le représenter par une plage limitée par un *ambitus* (notes extrêmes). On obtient une telle notation en appliquant la fonction `\makeClusters` à une séquence d'accords, comme

```
\relative \makeClusters { <g' b>2 <c g'> }
```



Des notes ordinaires et des clusters peuvent cohabiter sur une même portée, y compris simultanément – en pareil cas, rien ne sera fait pour tenter d'empêcher les chevauchements entre notes et clusters.

Voir aussi

Glossaire musicologique : Section "cluster" dans *Glossaire*.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "ClusterSpanner" dans *Référence des propriétés internes*, Section "ClusterSpannerBeacon" dans *Référence des propriétés internes*, Section "Cluster_spanner_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L'apparence d'un cluster sera extrêmement resserrée s'il ne comporte au moins deux accords.

Dans la mesure où un cluster ne possède pas de hampe, il n'y a aucun moyen d'en connaître la durée ; cependant la longueur du signe imprimé dépend directement de la durée affectée aux accords qui le définissent.

Seul un silence peut séparer deux clusters.

Les clusters ne sont pas reproduits en MIDI.

1.5.2 Plusieurs voix
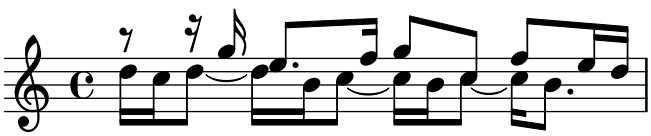
Nous allons nous intéresser, dans les paragraphes qui suivent, à la gestion de notes simultanées réparties sur plusieurs voix ou plusieurs portées.

Polyphonie sur une portée

Instanciation explicite des voix

La manière la plus facile d'entrer des fragments avec plus d'une voix sur une portée est la suivante :

```
\new Staff <<
  \new Voice = "first"
    \relative { \voiceOne r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    \relative { \voiceTwo d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Vous constaterez que les voix sont créées explicitement et qu'elles sont nommées. Les commandes `\voiceOne` ... `\voiceFour` déterminent les voix de telle sorte que les première et troisième auront des hampes vers le haut, et les deuxième et quatrième vers le bas. D'autre part, les notes des troisième et quatrième voix seront quelque peu décalées, tout comme leurs silences, afin d'éviter les collisions. La commande `\oneVoice` permet de retrouver les réglages par défaut.

Polyphonie temporaire
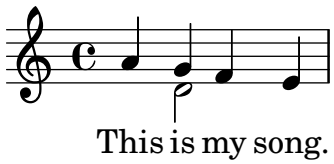
Un fragment temporairement polyphonique se construit de la manière suivante :

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

En fait, la première expression d'une polyphonie temporaire reste dans le même contexte `Voice` que celui existant auparavant et qui perdurera après ce fragment. Les autres expressions entre doubles chevrons seront assignées à des voix temporaires distinctes. C'est la raison pour laquelle les paroles qui suivaient la voix avant la polyphonie continueront à le faire durant ce passage polyphonique et après lui :

```
\relative <<
  \new Voice = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
```

```
    }
  >>
  \oneVoice
  e4
}
\new Lyrics \lyricsto "melody" {
This is my song.
}
>>
```
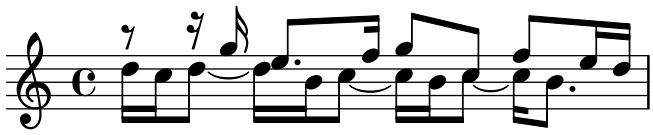


Vous remarquerez que les commandes `\voiceOne` et `\voiceTwo` permettent d'obtenir des réglages différents pour chacune des voix.

La construction avec un double antislash

Une construction de la forme `<< {...} \\ {...} >>`, dans laquelle plusieurs expressions sont séparées par des doubles obliques inversées, se comporte différemment de celle sans séparateur : **tous** les membres de cette construction seront assignés à de nouveaux contextes de voix. Ces contextes de voix, créés implicitement, portent les noms `"1"`, `"2"`, etc. Dans chacun de ces contextes, le positionnement des liaisons, la direction des hampes, etc. sont réglés de manière appropriée. En voici un exemple :
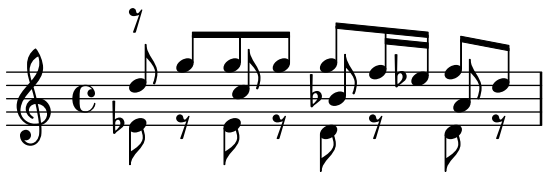
```
<<
  \relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \\
  \relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Cette syntaxe peut être utilisée dans la mesure où la création puis la disparition de voix temporaires sont sans conséquence. Les réglages de ces voix créées implicitement sont les mêmes que si elles avaient été créées à l'aide des commandes `\voiceOne` à `\voiceFour`, dans leur ordre d'apparition.

Dans l'exemple qui suit, la voix intermédiaire a des hampes vers le haut. Nous la plaçons donc en troisième position, de telle sorte qu'elle adopte les réglages de `\voiceThree` qui correspondent à ce que nous voulons. Grâce à des espaces invisibles, nous évitons de surcharger la portée avec des demis soupirs.

```
<<
  \relative { r8 g'' g  g g f16 ees f8 d }
  \\
  \relative { ees'8 r ees r d r d r }
  \\
  \relative { d''8 s c s bes s a s }
>>
```



En dehors des cas les plus simples, nous vous invitons à toujours créer les contextes de voix de manière explicite. Voir à ce sujet Section “Contextes et graveurs” dans *Manuel d'initiation* et Section “Instanciation explicite des voix” dans *Manuel d'initiation*.
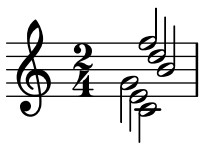
Ordre des voix

L'ordre dans lequel doivent apparaître les voix d'une construction simultanée suit le schéma suivant :

```
Voix 1 : la plus haute
Voix 2 : la plus basse
Voix 3 : deuxième plus haute
Voix 4 : deuxième plus basse
Voix 5 : troisième plus haute
Voix 6 : troisième plus basse
etc.
```

Cette présentation en entonnoir peut sembler quelque peu contre-intuitive ; elle simplifie cependant grandement le processus de mise en forme. Vous noterez que les hampes des voix au numéro impair vont vers le haut, celles des voix paires vers le bas :
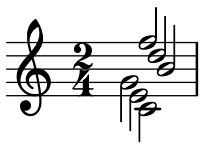
```
\new Staff <<
  \time 2/4
  { f''2 }  % 1: highest
  \\
  { c'2  }  % 2: lowest
  \\
  { d''2 }  % 3: second-highest
  \\
  { e'2  }  % 4: second-lowest
  \\
  { b'2  }  % 5: third-highest
  \\
  { g'2  }  % 6: third-lowest
>>
```



La commande `\voices` permet de gérer l'ordre de saisie des voix :

```
\new Staff \voices 1,3,5,6,4,2 <<
  \time 2/4
  { f''2 }  % 1: highest
  \\
  { d''2 }  % 3: second-highest
  \\
  { b'2  }  % 5: third-highest
  \\
  { g'2  }  % 6: third-lowest
  \\
```

```
  { e'2  }  % 4: second-lowest
  \\
  { c'2  }  % 2: lowest
>>
```
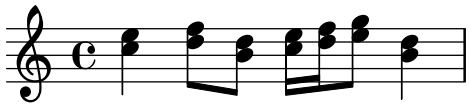


Note : Paroles et objets étendus (liaisons, soufflets, etc.) ne peuvent passer d'une voix à l'autre.

Identité rythmique

Lorsque l'on doit saisir des fragments de musique parallèle qui ont le même rythme, on peut les combiner dans un contexte de voix unique et par voie de conséquence former des accords. Il suffit pour cela de les regrouper dans une construction de musique simultanée simple au sein d'une voix explicite :

```
\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>
```



Prenez garde que les différents éléments doivent impérativement avoir la même structure rythmique, sous peine de ligature aléatoire et de messages d'avertissement.

Commandes prédéfinies

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

Voir aussi

Manuel d'initiation : Section "Instanciation explicite des voix" dans *Manuel d'initiation*, Section "Les voix contiennent la musique" dans *Manuel d'initiation*.

Manuel de notation : [Hampes], page 245, [Portées de percussion], page 425, [Silences invisibles], page 63.

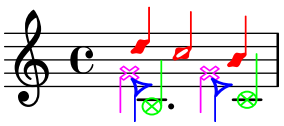
Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Styles de voix

Opter pour des couleurs et des têtes de notes spécifiques selon la voix permet de les identifier plus facilement :

```
<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \\
  \relative { \voiceTwoStyle e'2 e }
  \\
  \relative { \voiceThreeStyle b2. c4 }
```

```
  \\
  \relative { \voiceFourStyle g'2 g }
>>
```



La commande `\voiceNeutralStyle` permet de revenir à une présentation normale.

Commandes prédéfinies

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

Voir aussi
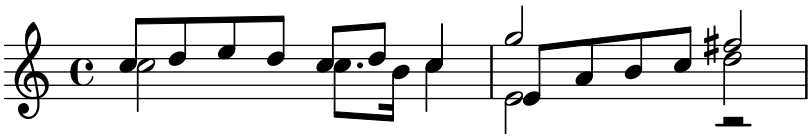
Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "J'entends des Voix" dans *Manuel d'initiation*.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Résolution des collisions

Les notes de hauteur identique appartenant à des voix différentes, même si leur hampe sont opposées, verront leur tête automatiquement fusionner. Les notes dont la tête diffère ou bien qui ont la hampe dans la même direction ne seront pas automatiquement fusionnées. Les silences, lorsqu'ils sont dans une autre voix et à l'opposé des hampes seront décalés verticalement. Vous constaterez, dans l'exemple suivant, que la fusion échoue aux premier et troisième temps de la première mesure, ainsi qu'au premier temps de la deuxième mesure.
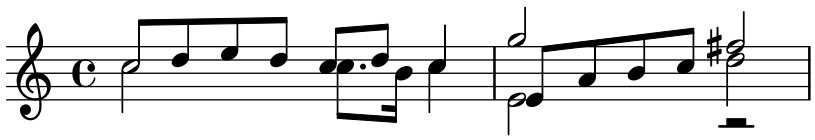
```
<<
  \relative {
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Cependant, vous pouvez fusionner une tête de blanche avec une tête de croche – jamais avec une noire. Les têtes du premier temps de la première mesure ont bien fusionné :
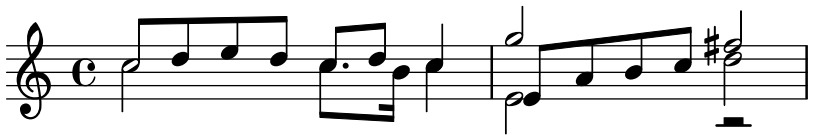
```
<<
  \relative {
```

```
    \mergeDifferentlyHeadedOn
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



De même, vous pouvez fusionner les têtes de notes pointées et non pointées comme au troisième temps de la première mesure :
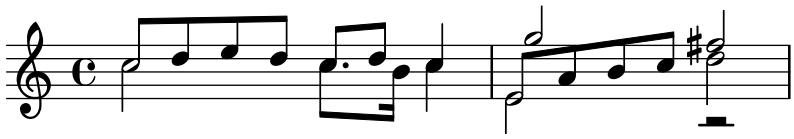
```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Lorsque trois notes ou plus s'agglutinent dans un même empilement, `\mergeDifferentlyHeadedOn` ne peut mener à bien la fusion des deux notes qui devraient l'être. Pour obtenir une fusion optimale, appliquez un décalage (`\shift`) à la note qui ne devrait pas fusionner. Ici, on applique un `\shiftOn` pour décaler le *sol* de l'empilement ; le rendement de `\mergeDifferentlyHeadedOn` est alors comme il faut.

```
<<
  \relative {
```

```
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



La commande `\shiftOn` permet, sans pour autant le forcer, un décalage des notes d'une voix en particulier. Une note ou un accord appartenant à cette voix ne seront décalés que si leur hampe menaçait d'entrer en collision avec une hampe appartenant à une autre voix allant dans la même direction. La commande `\shiftOff` interdit l'apparition de décalage.
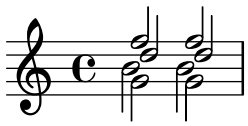
Les voix externes – habituellement les voix une et deux – sont affectées de `\shiftOff`, alors que les voix internes – trois et quatre – sont affectées de `\shiftOn`. Lorsqu'un décalage intervient, les notes dont les hampes sont ascendantes (voix impaire) iront vers la droite, et les notes à hampe descendante (voix paire) iront vers la gauche.

Voici un exemple qui vous permettra de bien visualiser ce qui se passe en interne.

> **Note :** Lorsqu'il y a trois voix ou plus, prenez garde au fait que l'ordre d'apparition des voix dans votre fichier ne correspond pas à l'ordre vertical des voix tel qu'il apparaîtra sur la portée.

```
\new Staff \relative {
  %% saisie abrégée
  <<
    { f''2  }  % 1 : extrême haute
    \\
    { g,2 }  % 2 : extrême basse
    \\
    { d'2 }  % 3 : intermédiaire haute
    \\
    { b2  }  % 4 : intermédiaire basse
  >>
  %% expansion en interne de ce qui précède
  <<
    \new Voice = "1" { \voiceOne   \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo   \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn  d'2 } % décale à droite
```

```
    \new Voice = "4" { \voiceFour  \shiftOn  b2  } % décale à gauche
  >>
}
```



Les commandes `\shiftOnn` et `\shiftOnnn` sont des niveaux supplémentaires de décalage qui peuvent s'adopter temporairement dans certaines situations complexes – voir Section "Exemple concret" dans *Manuel d'initiation*.

Les têtes de notes ne fusionneront que dans la mesure où leur hampe sont opposées – implicitement parce qu'appartenant aux voix une ou deux, ou bien explicitement.

Commandes prédéfinies

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.

Morceaux choisis
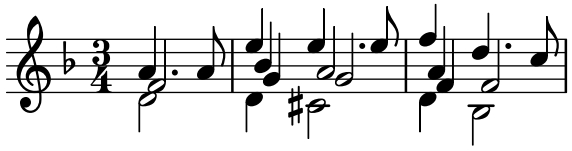
Ajout de voix pour éviter les collisions

Dans certains cas de musique polyphonique complexe, une voix supplémentaire peut permettre d'éviter les risques de collision. Lorsque quatre voix parallèles ne suffisent pas, la fonction Scheme `context-spec-music` permet d'ajouter encore d'autres voix.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

\relative c'' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice  {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \new Voice {
      \voiceTwo
      d,2
      d4 cis2
      d4 bes2
    }
    \new Voice {
      \voiceThree
      f'2
      bes4 a2
      a4 s2
    }
    \new Voice {
```
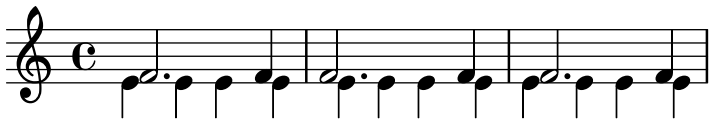
```
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
  >>
}
```



Déplacement des notes pointées dans une polyphonie

Une note pointée appartenant à la voix supérieure d'une portée polyphonique sera par défaut décalée vers la droite afin d'éviter les collisions avec les autres voix. Ce comportement peut être outrepassé à l'aide de la propriété `prefer-dotted-right` de `NoteCollision`.

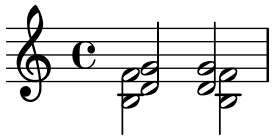
```
\new Staff \relative c' <<
  {
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##f
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##t
    f2. f4
  }
  \\
  { e4 e e e e e e e e e e }
>>
```



Décalage horizontal forcé

Quand LilyPond est dépassé, la propriété `force-hshift` de l'objet `NoteColumn` et des silences à hauteur déterminée peuvent s'avérer utiles pour dicter au programme les choix de placement. On travaille ici en espace de portée.

```
\relative c' <<
  {
    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn.force-hshift = #1.7
    <b f'>2
  }
>>
```



Voir aussi

Glossaire musicologique : Section “polyphonie” dans *Glossaire*.

Manuel d’initiation : Section “Exemple concret” dans *Manuel d’initiation*, Section “Les voix contiennent la musique” dans *Manuel d’initiation*, Section “Notes simultanées” dans *Manuel d’initiation*.

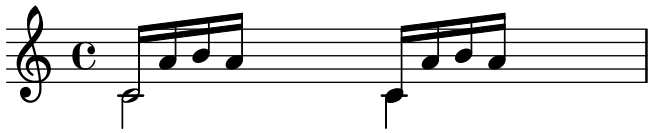
Morceaux choisis : Section “Notation simultanée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “NoteColumn” dans *Référence des propriétés internes*, Section “NoteCollision” dans *Référence des propriétés internes*, Section “RestCollision” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Une clause `\override NoteColumn.ignore-collision = ##t` aura pour effet une fusion incorrecte des têtes de note différentes à partir de voix différentes.
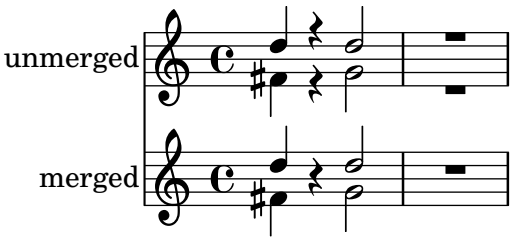
```
\mergeDifferentlyHeadedOn
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
\override NoteColumn.ignore-collision = ##t
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
```



Fusion de silences

Il est d’usage, lorsque plusieurs voix cohabitent, de fusionner les silences qui apparaissent simultanément dans différentes parties. Ceci peut s’obtenir à l’aide du `Merge_rests_engraver`.

```
voiceA = \relative { d''4 r d2 | R1 | }
voiceB = \relative { fis'4 r g2 | R1 | }
\score {
  <<
    \new Staff \with {
      instrumentName = "unmerged"
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
    \new Staff \with {
      instrumentName = "merged"
      \consists "Merge_rests_engraver"
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
  >>
}
```



La propriété de contexte `suspendRestMerging`, lorsque activée par un `##t`, permet de suspendre temporairement la fusion des silences.

Regroupement automatique de parties

Le regroupement automatique de parties vous permet de fusionner deux pupitres sur une seule portée, ceci dans le but de créer des partitions d'orchestre. Lorsque les deux parties sont identiques sur une certaine durée, une seule s'affiche. Lorsqu'elles diffèrent, deux voix séparées apparaissent, avec des hampes dont la direction est gérée automatiquement. Vous pouvez aussi identifier et faire ressortir les solos et parties *a due*.

Voici la syntaxe qui permet de combiner des parties :
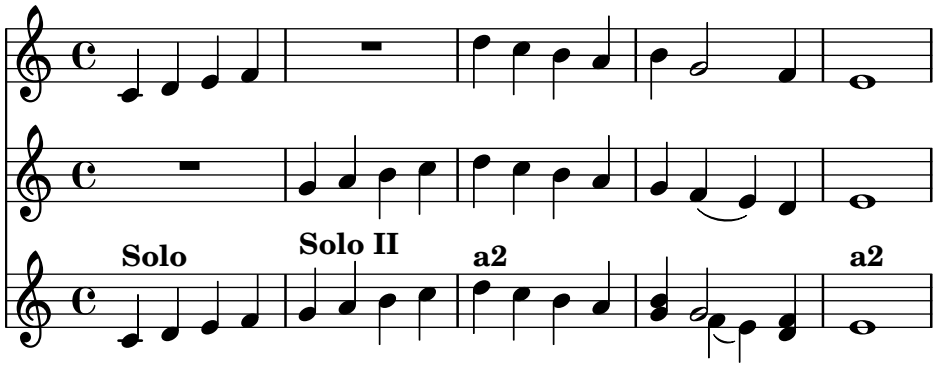
`\partCombine` *`expression_musicale_1 expression_musicale_2`*

L'exemple suivant illustre les fonctionnalités élémentaires du combinateur de parties : positionner les parties sur une portée, gérer la direction des hampes et de la polyphonie. Les identifiants sont les mêmes pour la combinaison et les parties séparées.

```
instrumentOne = \relative {
  c'4 d e f |
  R1 |
  d'4 c b a |
  b4 g2 f4 |
  e1 |
}

instrumentTwo = \relative {
  R1 |
  g'4 a b c |
  d4 c b a |
  g4 f( e) d |
  e1 |
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partCombine \instrumentOne \instrumentTwo
>>
```



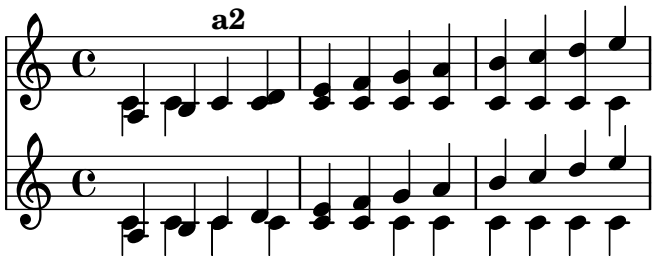
Les notes de la troisième mesure n'apparaissent qu'une seule fois, alors qu'elles ont été spécifiées deux fois (une fois dans chacune des parties). La direction des hampes et des liaisons de tenue ou de phrasé est gérée automatiquement, selon qu'il s'agisse d'un solo ou d'un unisson. La première partie, dont le contexte s'appellera `one`, aura toujours ses hampes dirigées vers le haut et sera notée « Solo », alors que la deuxième, appelée `two`, aura des hampes vers le bas et sera notée « Solo II ». Les parties à l'unisson seront par défaut estampillées d'un « a2 ».

Par défaut, le combinateur fusionnera deux notes de même hauteur en une note *a due*, regroupera en accord les notes de même rythme et dont l'intervalle est inférieur à une neuvième, enfin isolera les notes séparées de plus d'une neuvième (ou si les voix se croisent) dans des voix distinctes. Ceci peut s'adapter à l'aide d'une paire de nombres fournie en argument optionnel à la commande `\partCombine` : le premier nombre spécifie l'intervalle à partir duquel les notes seront combinées (0 par défaut) et le second celui à partir duquel les notes seront placées dans des voix séparées. Un second élément de cette paire à zéro obligera le combinateur à séparer les notes dès la seconde ; s'il est à un, elles seront séparées à partir de la tierce, et ainsi de suite.

```
instrumentOne = \relative {
  a4 b c d |
  e f g a |
  b c d e |
}

instrumentTwo = \relative {
  c'4 c c c |
  c c c c |
  c c c c |
}

<<
  \new Staff \partCombine \instrumentOne \instrumentTwo
  \new Staff \partCombine #'(2 . 3) \instrumentOne \instrumentTwo
>>
```



LilyPond interprète dans un contexte `Voice` chacun des arguments fournis à `\partCombine`. Si vous travaillez avec des octaves relatives, spécifiez `\relative` dans chacune des expressions musicales, comme ceci :

```
\partCombine
  \relative ... expression_musicale_1
  \relative ... expression_musicale_2
```

Une section `\relative` à l'extérieur du `\partCombine` restera sans effet sur les hauteurs de *`expression_musicale_1`* ou de *`expression_musicale_2`*.

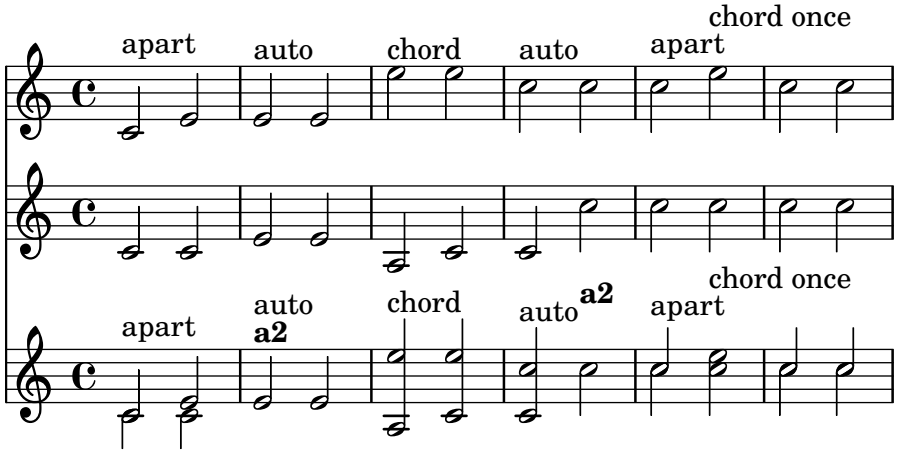
En matière d'édition professionnelle, les voix sont souvent maintenues séparément et sur une durée conséquente, bien que les notes des différentes voix soient les mêmes et pourraient donc être présentées à l'unisson. Dans la mesure où `\partCombine` considère les notes séparément, combiner des notes en accord ou indiquer un solo ne serait pas optimal en pareil cas. LilyPond

dispose alors de certaines commandes qui permettent d'influencer le comportement de la fonction `\partCombine`. Ces commandes peuvent se voir préfixées d'un `\once` de sorte à n'affecter que la note qui les suit directement dans l'expression musicale.

- `\partCombineApart` maintient les notes dans des voix séparées même si elles peuvent se combiner en accord ou en unisson.
- `\partCombineChords` combine les notes en accords.
- `\partCombineUnisono` combine les voix en un « unisson ».
- `\partCombineSoloI` affiche exclusivement la première voix et l'affuble d'un « Solo ».
- `\partCombineSoloII` affiche exclusivement la deuxième voix et l'affuble d'un « Solo ».
- `\partCombineAutomatic` annule les effets des dérogations précédentes et active le comportement standard de la fonction `\partCombine`.

```
instrumentOne = \relative c' {
  \partCombineApart c2^"apart" e |
  \partCombineAutomatic e2^"auto" e |
  \partCombineChords e'2^"chord" e |
  \partCombineAutomatic c2^"auto" c |
  \partCombineApart c2^"apart"
    \once \partCombineChords e^"chord once" |
  c2 c |
}
instrumentTwo = \relative {
  c'2 c |
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partCombine \instrumentOne \instrumentTwo }
>>
```



Utilisation de \partCombine *et paroles*

La commande \partCombine n'est pas conçue pour traiter des paroles ; si l'une des voix est explicitement nommée, afin de lui adjoindre des paroles, le combinateur de parties cessera de fonctionner. Toutefois, le recours à un contexte NullVoice permet d'obtenir les effets escomptés – voir [Polyphonie et paroles communes], page 318.

Morceaux choisis

Combinaison de deux parties sur une même portée

L'outil de combinaison de parties (la commande \partCombine) permet d'avoir deux parties différentes sur une même portée. LilyPond ajoute automatiquement des indications textuelles, telles que « solo » ou « a2 ». Si votre intention n'est que de fusionner les parties, sans ajouter de texte, assignez faux à la propriété printPartCombineTexts. Dans le cas de partitions vocales, et plus particulièrement d'hymnes, ces « solo/a2 » ne sont d'aucune utilité, aussi vaut-il mieux les désactiver. Dans le cas où il y aurait alternance entre *solo* et *tutti*, il vaut mieux faire appel à de la musique polyphonique standard.

Voici trois moyens d'imprimer deux parties sur une même portée : en polyphonie normale, avec \partCombine sans indication supplémentaire, et avec \partCombine commentée.

```
%% Combining pedal notes with clef changes

musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

\score {
    <<
    \new Staff \with { instrumentName = "Standard polyphony" }
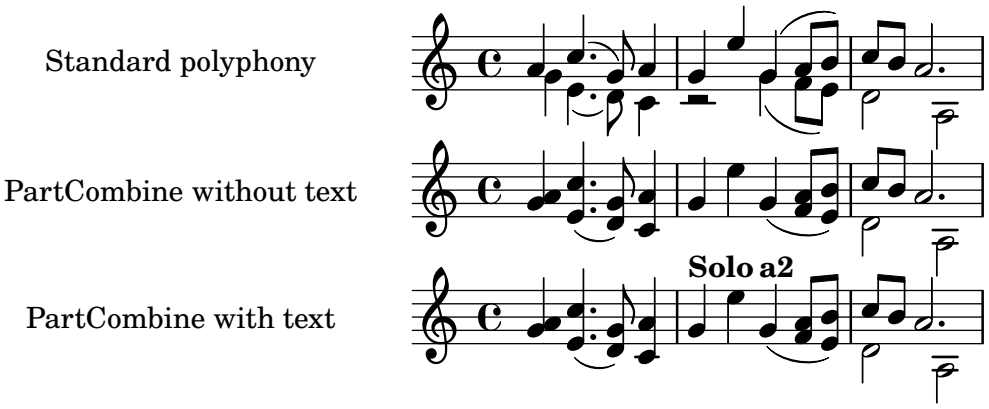
      << \musicUp \\ \musicDown >>

    \new Staff \with {
      instrumentName = "PartCombine without text"
      printPartCombineTexts = ##f
    }

    \partCombine \musicUp \musicDown

    \new Staff \with { instrumentName = "PartCombine with text" }
      \partCombine \musicUp \musicDown
    >>
  \layout {
    indent = 6.0\cm
    \context {
```
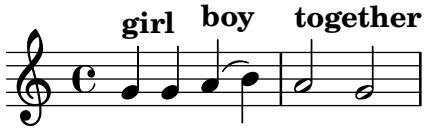
```
      \Score
      \override SystemStartBar.collapse-height = #30
    }
  }
}
```



Modification des indications de parties combinées

Lorsque vous regroupez automatiquement des parties, vous pouvez modifier le texte qui sera affiché pour les solos et pour les parties à l'unisson :

```
\new Staff <<
  \set Staff.soloText = #"girl"
  \set Staff.soloIIText = #"boy"
  \set Staff.aDueText = #"together"
  \partCombine
    \relative c'' {
      g4 g r r
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
>>
```



Voir aussi

Glossaire musicologique : Section "a due" dans *Glossaire*, Section "partie" dans *Glossaire*.

Manuel de notation : Section 1.6.3 [Écriture de parties séparées], page 217.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PartCombineMusic" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les différentes commandes `\partCombine...` ne prennent en charge que deux voix. De la même manière, le combinateur n'est pas conçu pour travailler avec des paroles ; il s'arrête dès qu'il est explicitement fait appel à l'une des voix pour y attacher des paroles.

`\partCombine...` ne peut s'inscrire ni dans un bloc `\tuplet` ni dans un bloc `\relative`.

Lorsque `printPartCombineTexts` est actif et que les deux voix jouent souvent les mêmes notes, le combinateur peut afficher `a2` plus d'une fois par mesure.

`\partCombine` n'examine que l'attaque des notes. Il n'est donc pas en mesure de déterminer si une note attaquée précédemment est encore jouée ou non, ce qui peut engendrer quelques problèmes, entre autres des indications de « Solo » ou « Unison » incorrectement placées.

`\partCombine` conserve les objets étendus (liaisons, soufflets, etc.) dans la même voix de sorte à éviter qu'ils soient improprement ou pas du tout imprimés lorsque leur départ ou terminaison est dans une voix différente.

En interne, `\partCombine` interprète les deux arguments en tant que `Voice`s, dénommées `one` et `two`, puis décide de quand les parties seront fusionnées. Par conséquent, si les arguments changent pour d'autres noms de contexte `Voice`, les événements qu'ils contiendraient seront ignorés.

Certaines considérations apparaissent aussi dans les chapitres [Tablatures par défaut], page 371, et [Barres de ligature automatiques], page 87.

Saisie de musique en parallèle
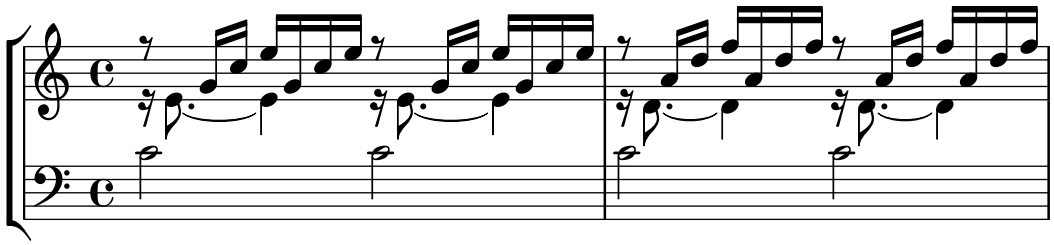
On peut écrire plusieurs voix de façon entremêlée. La fonction `\parallelMusic` prend en charge une liste des variables à créer, ainsi qu'une expression musicale. Le contenu des différentes mesures de l'expression musicale deviennent les valeurs des variables respectives que vous pourrez ensuite utiliser pour imprimer la partition.

> **Note :** Les contrôles de barre de mesure `|` sont obligatoires et les mesures doivent être de longueur identique.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Bar 1
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~   4              r16 e'8.~   4              |
  c'2                        c'2                        |

  % Bar 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
  r16 d'8.~   4              r16 d'8.~   4              |
  c'2                        c'2                        |

}
\new StaffGroup <<
  \new Staff << \voiceA \\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```
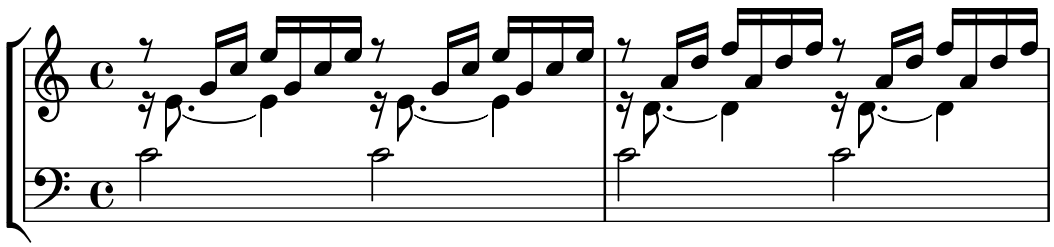


Vous pouvez travailler en mode relatif. Notez cependant que la commande `\relative` n'apparaît pas au sein du bloc `\parallelMusic`. Le calcul des hauteurs relatives s'effectue voix par voix, et non au fil des lignes saisies ; en d'autres termes, les notes de la `voiceA` ignorent tout de celles de la `voiceB`.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ 4        r16 e8.~  4         |
  c2                c                   |

  % Bar 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  4        r16 d8.~  4        |
  c2                 c                  |

 }
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
  \new Staff \relative c' { \clef bass \voiceC }
>>
```



Ceci fonctionne bien avec la musique pour piano. L'exemple suivant affecte quatre mesures à quatre variables :

```
global = {
  \key g \major
  \time 2/4
}
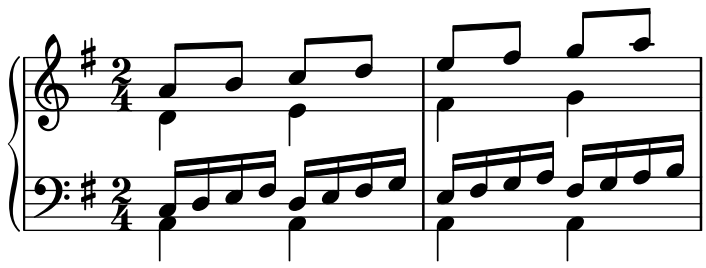
\parallelMusic voiceA,voiceB,voiceC,voiceD {
  % Bar 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g |
  a4          a         |

  % Bar 2
  e8      fis  g     a   |
  fis4         g         |
  e16 fis g  a fis g a b |
```

```
  a4           a         |

  % Bar 3 ...
}

\score {
  \new PianoStaff <<
     \new Staff {
       \global
       <<
         \relative c'' \voiceA
         \\
         \relative c'  \voiceB
       >>
     }
     \new Staff {
       \global \clef bass
       <<
         \relative c \voiceC
         \\
         \relative c \voiceD
       >>
     }
  >>
}
```
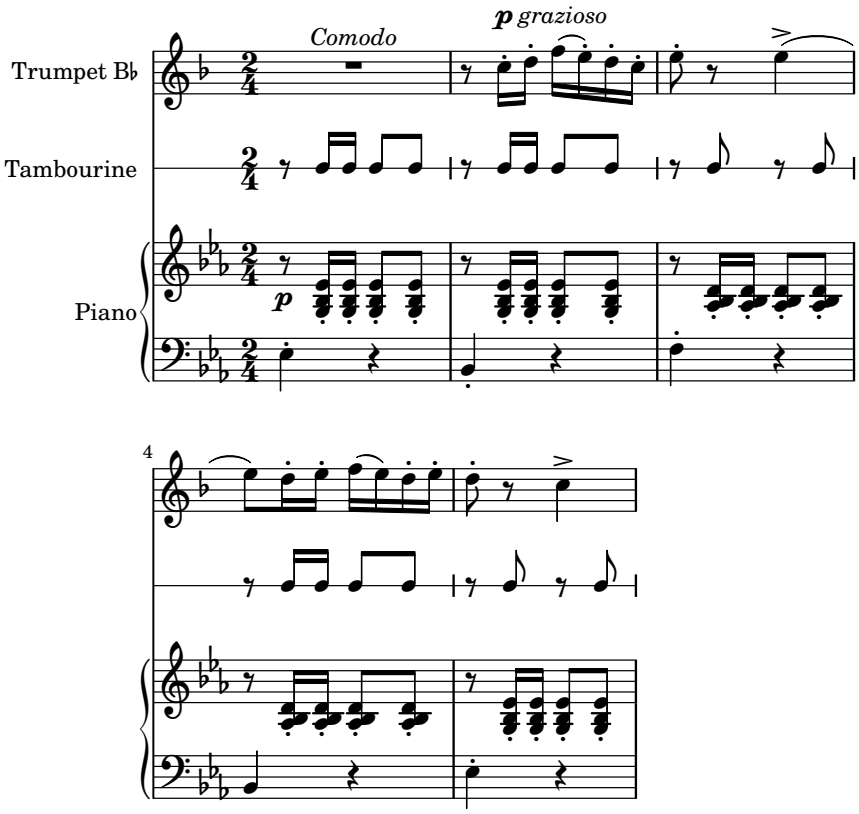


Voir aussi

Manuel d'initiation : Section “Organisation du code source avec des variables” dans *Manuel d'initiation*.

Morceaux choisis : Section “Notation simultanée” dans *Morceaux choisis*.

1.6 Notation sur la portée



Cette section aborde les détails de gravure de la portée, la réalisation de partitions comprenant plusieurs portées et l'ajout d'indications globales d'exécution, présentes sur certaines portées seulement.

1.6.1 Gravure des portées
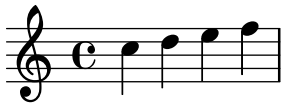
Nous allons voir ici comment créer des portées et comment les regrouper.

Initialisation de nouvelles portées

Les *portées* – en anglais *staff* (*staves* au pluriel) – sont créées à l'aide des commandes `\new` ou `\context`. Pour de plus amples détails, consultez Section 5.1.2 [Création et référencement d'un contexte], page 630.

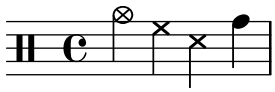
Le contexte de portée standard s'appelle `Staff` :

```
\new Staff \relative { c''4 d e f }
```



Le contexte `DrumStaff` crée une portée à cinq lignes correspondant à une batterie traditionnelle et chacun des instruments est représenté par un symbole spécifique. Les éléments sont saisis en mode batterie, initialisé par la commande `\drummode`, chaque composante étant spécifiée par son nom. Pour de plus amples détails, consultez [Portées de percussion], page 425.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



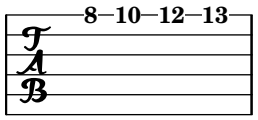
Un `RhythmicStaff` est composé d'une portée à ligne unique chargée de présenter les valeurs rythmiques saisies. Seules sont imprimées les durées. Pour de plus amples détails, consultez [Gravure de lignes rythmiques], page 84.

```
\new RhythmicStaff { c4 d e f }
```



Un `TabStaff` crée une portée de tablature correspondant aux six cordes d'une guitare standard. Pour de plus amples détails, consultez [Tablatures par défaut], page 371.
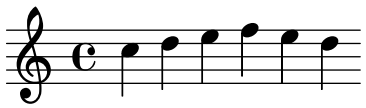
```
\new TabStaff \relative { c''4 d e f }
```



LilyPond dispose aussi de deux contextes dédiés à la musique ancienne : `MensuralStaff` et `VaticanaStaff`. Ils sont abordés plus en détails au chapitre [Contextes prédéfinis], page 470.

Le contexte `GregorianTranscriptionStaff` permet d'obtenir des éditions modernes du grégorien. Bien entendu, il est dépourvu de barres de mesure.

```
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }
```



Vous pourrez toujours définir d'autres contextes de portée selon vos besoins, en suivant les indications fournies au chapitre Section 5.1.6 [Définition de nouveaux contextes], page 644.

Voir aussi

Glossaire musicologique : Section "staff" dans *Glossaire*, Section "portées" dans *Glossaire*.

Manuel de notation : [Contextes de musique mensurale], page 472, [Contextes du chant grégorien], page 479, [Contextes prédéfinis], page 470, Section 5.1.2 [Création et référencement d'un contexte], page 630, Section 5.1.6 [Définition de nouveaux contextes], page 644, [Gravure de lignes rythmiques], page 84, [Portées de percussion], page 425, [Symbole de la portée], page 205, [Tablatures par défaut], page 371.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

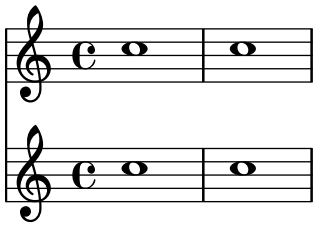
Référence des propriétés internes : Section "Staff" dans *Référence des propriétés internes*, Section "DrumStaff" dans *Référence des propriétés internes*, Section "GregorianTranscriptionStaff" dans *Référence des propriétés internes*, Section "RhythmicStaff" dans *Référence des propriétés internes*, Section "TabStaff" dans *Référence des propriétés internes*, Section "MensuralStaff" dans *Référence des propriétés internes*, Section "VaticanaStaff" dans *Référence des propriétés internes*, Section "StaffSymbol" dans *Référence des propriétés internes*.

Regroupement de portées

LilyPond dispose de différents contextes permettant de regrouper des portées individuelles et d'obtenir ainsi des « systèmes ». Chacun de ces contextes définira le style de regroupement, avec son signe particulier en début de portée et ses règles de gestion des barres de mesure.

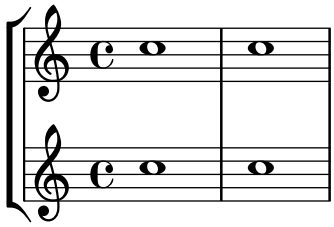
Lorsqu'aucun contexte particulier n'est spécifié, les propriétés suivantes s'appliqueront par défaut : les portées du groupe ne sont pas reliées, hormis par une simple ligne verticale en début de ligne, et les barres de mesure sont indépendantes.

```
<<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



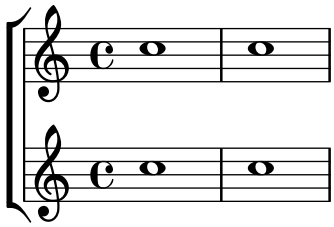
Dans un `StaffGroup`, le groupe de portées est signifié par un crochet, et les barres de mesure sont d'un seul tenant.

```
\new StaffGroup <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Dans un `ChoirStaff`, le groupe de portées est signifié par un crochet sur la gauche, et les barres de mesure sont individuelles.

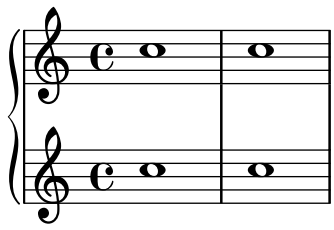
```
\new ChoirStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Dans un `GrandStaff`, le groupe de portées est signifié par une accolade sur la gauche, et les barres de mesure sont d'un seul tenant.

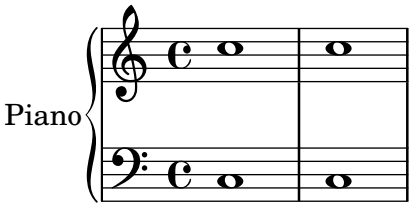
```
\new GrandStaff <<
```

```
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Le contexte `PianoStaff` est identique au `GrandStaff`, sauf qu'il gère automatiquement l'affichage du nom d'instrument – voir [Noms d'instrument], page 217, pour plus de détails.

```
\new PianoStaff \with { instrumentName = "Piano" }
  <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { \clef bass c1 c }
>>
```



Les barres de mesure au début de chaque système adoptent l'un des styles `SystemStartBar`, `SystemStartBrace` ou `SystemStartBracket`. Dans chaque contexte, seul l'un de ces styles est utilisé, et c'est la propriété `systemStartDelimiter` qui détermine lequel. Un quatrième style, `SystemStartSquare`, doit quant à lui être spécifié de manière explicite.

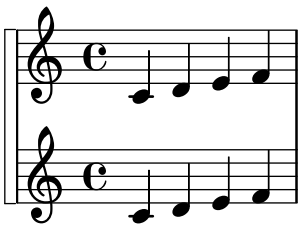
Vous pouvez aussi créer vos propres contextes de regroupement, comme l'explique Section 5.1.6 [Définition de nouveaux contextes], page 644.

Morceaux choisis

Indication de regroupement de portées par un rectangle

Un regroupement de portées sera indiqué par un simple rectangle – `SystemStartSquare` – en début de ligne dès lors que vous le mentionnerez explicitement au sein d'un contexte `StaffGroup` ou `ChoirStaff`.

```
\score {
  \new StaffGroup { <<
  \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```

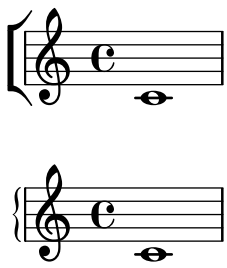


Indicateur de regroupement et portée unique

Lorsque, dans des regroupements de type `ChoirStaff` ou `StaffGroup`, une seule portée est active, aucune indication n'est donnée en début de ligne. Affecter à la propriété `collapse-height` un nombre de lignes inférieur à celui de la portée permet de modifier ce comportement par défaut.

Notez bien que dans le cas des `PianoStaff` et `GrandStaff`, pour lesquels le délimiteur de système est une accolade et non un crochet, il ne s'agit pas de la même propriété – voir le deuxième système de l'exemple.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
```



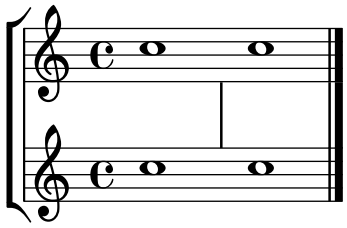
Présentation à l'ancienne (barres de mesure entre les portées)

En musique mensurale, les barres de mesure ne traversent pas les portées. Pour obtenir ce résultat avec un `StaffGroup` plutôt qu'en utilisant un `ChoirStaff`, il faudra masquer les portions de barre qui recouvrent les portées à l'aide d'un `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c'' {
  <<
```

```
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “accolade” dans *Glossaire*, Section “crochet” dans *Glossaire*, Section “système” dans *Glossaire*.

Manuel de notation : Section 5.1.6 [Définition de nouveaux contextes], page 644, [Noms d’instrument], page 217.
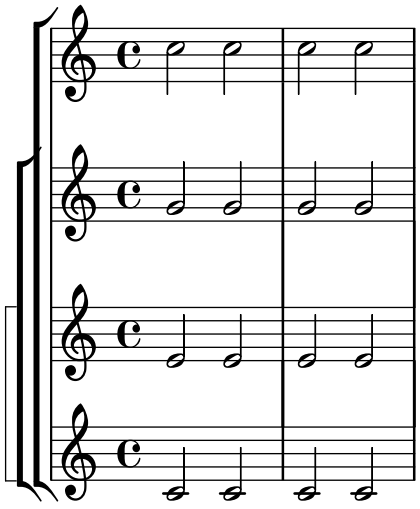
Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Staff” dans *Référence des propriétés internes*, Section “StaffGroup” dans *Référence des propriétés internes*, Section “ChoirStaff” dans *Référence des propriétés internes*, Section “GrandStaff” dans *Référence des propriétés internes*, Section “PianoStaff” dans *Référence des propriétés internes*, Section “SystemStartBar” dans *Référence des propriétés internes*, Section “SystemStartBrace” dans *Référence des propriétés internes*, Section “SystemStartBracket” dans *Référence des propriétés internes*, Section “SystemStartSquare” dans *Référence des propriétés internes*.

Imbrication de regroupements de portées

Les accolades et crochets qui délimitent les systèmes peuvent être imbriqués en profondeur. Chaque niveau inférieur aura son propre délimiteur, en plus de celui du niveau supérieur.

```
\new StaffGroup <<
  \new Staff \relative { c''2 c | c2 c }
  \new StaffGroup <<
    \new Staff \relative { g'2 g | g2 g }
    \new StaffGroup \with {
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff \relative { e'2 e | e2 e }
      \new Staff \relative { c'2 c | c2 c }
    >>
  >>
>>
```



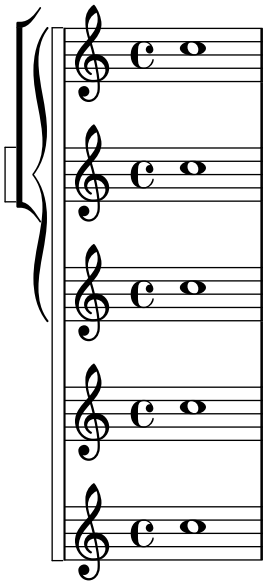
Vous pouvez aussi créer vos propres contextes d'imbrication, comme l'explique Section 5.1.6 [Définition de nouveaux contextes], page 644.

Morceaux choisis

Imbrications de regroupements de portées

La propriété `systemStartDelimiterHierarchy` permet de créer des regroupements imbriqués complexes. La commande `\set StaffGroup.systemStartDelimiterHierarchy` prend en argument la liste alphabétique des sous-groupes à hiérarchiser. Chaque sous-groupe peut être affublé d'un délimiteur particulier. Chacun des regroupements intermédiaires doit être borné par des parenthèses. Bien que des éléments de la liste puissent être omis, le premier délimiteur embrassera toujours l'intégralité des portées. Vous disposez des quatre délimiteurs `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` et `SystemStartSquare`.

```
\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = #4
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                            (SystemStartSquare b)  ) c ) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



Voir aussi

Manuel de notation : Section 5.1.6 [Définition de nouveaux contextes], page 644, [Noms d'instrument], page 217, [Regroupement de portées], page 199.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

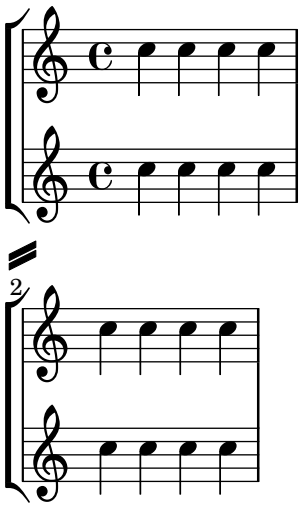
Référence des propriétés internes : Section "StaffGroup" dans *Référence des propriétés internes*, Section "ChoirStaff" dans *Référence des propriétés internes*, Section "SystemStartBar" dans *Référence des propriétés internes*, Section "SystemStartBrace" dans *Référence des propriétés internes*, Section "SystemStartBracket" dans *Référence des propriétés internes*, Section "SystemStartSquare" dans *Référence des propriétés internes*.

Séparation des systèmes

Le nombre de systèmes peut varier d'une page à l'autre ; vous pouvez, en pareil cas, rendre plus évidente la séparation entre les systèmes en l'indiquant visuellement. Ce « séparateur » est absent par défaut, mais vous pouvez l'activer par une option au sein du bloc `\paper`.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
    % following commands are needed only to format this documentation
```

```
    paper-width = 100\mm
    paper-height = 100\mm
    tagline = ##f
  }
}
```



Voir aussi

Manuel de notation : Section 4.1 [Mise en forme de la page], page 571.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.
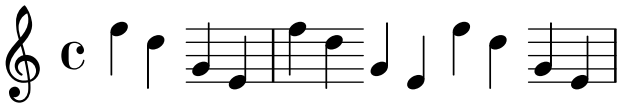
1.6.2 Modification de portées individuelles

Cette section explique le réglage de la gravure de chaque portée, comme la taille de portée ou le nombre de lignes ; sont aussi décrits la suspension et la reprise de portées, ainsi que les portées d’*ossia*.

Symbole de la portée

Les commandes `\stopStaff` et `\startStaff` permettent respectivement de clôturer et (re)démarrer une portée n’importe où dans une partition.

```
\relative {
  \stopStaff f''4 d \startStaff g, e
  f'4 d \stopStaff g, e
  f'4 d \startStaff g, e
}
```



Commandes prédéfinies
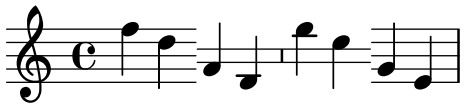
`\startStaff`, `\stopStaff`.

Les notes, nuances, etc. sont regroupées dans un assemblage de lignes horizontales, que l’on nomme la portée (en anglais *staff*, et *staves* au pluriel). Dans LilyPond, ces lignes sont dessinées au moyen d’un objet graphique (*grob*) à part entière, nommé `StaffSymbol` – symbole de portée. Modifier les propriétés d’un `StaffSymbol` changera l’apparence de la portée, dès lors qu’elles auront été définies avant de créer la portée en question.

Vous pouvez modifier le nombre de lignes d'une portée :
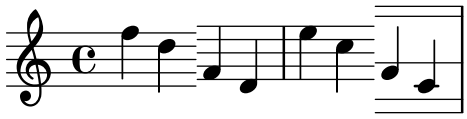
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-count = #2
  \startStaff g, e |

  f'4 d \stopStaff
  \revert Staff.StaffSymbol.line-count
  \startStaff g, e |
}
```



Le positionnement de chacune des lignes de la portée est modifiable. Une liste de nombres détermine le positionnement de chaque ligne. Le `0` correspond à la ligne médiane d'une portée habituelle, pour laquelle la liste est donc `(-4 -2 0 2 4)`. Une ligne sera donc imprimée pour chaque valeur exprimée ; le nombre de lignes, ainsi que leur position dans la portée, peut donc se modifier à l'aide d'une seule commande.
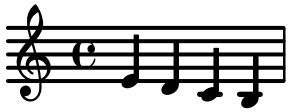
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
  \startStaff g, e |
  f'4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
  \startStaff g, e |
}
```



Afin de préserver l'orientation habituelle des hampes – ascendantes dans la partie inférieure de la portée, descendantes dans la partie supérieure – la ligne (ou l'interligne) centrale de la portée personnalisée devra être en phase avec avec la ligne médiane d'une portée classique (0). La position de la clef et celle du do médium demanderont parfois un ajustement afin d'être en phase avec cette nouvelle portée. Pour plus d'explications, reportez-vous aux exemples du chapitre [Clefs], page 18.
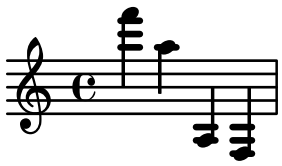
Lorsque vous modifierez l'épaisseur des lignes, gardez à l'esprit que les lignes supplémentaires et les hampes seront aussi modifiées.

```
\new Staff \with {
  \override StaffSymbol.thickness = #3
}
{ e4 d c b }
```



L'épaisseur des lignes supplémentaires (*ledger lines*) peut être déterminée indépendamment des lignes de la portée.

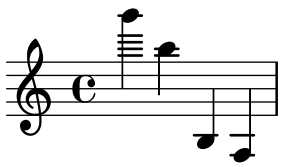
```
\new Staff \with {
  \override StaffSymbol.thickness = #2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f'''4 a, a,, f
}
```



La première valeur est multipliée par l'épaisseur de ligne de portée, la seconde par l'espace d'interligne ; ces deux valeurs sont alors combinées pour donner la nouvelle épaisseur des lignes supplémentaires.
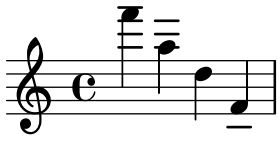
L'emplacement des lignes supplémentaires est réglable :

```
\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f'''4 a, a,, f
}
```



Vous pouvez faire apparaître des lignes supplémentaires additionnelles au-dessus ou en dessous des têtes de note selon leur positionnement relatif aux autres notes, qu'elles aient ou non elles-mêmes des lignes supplémentaires.
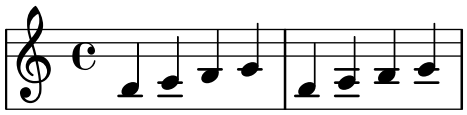
```
\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
} \relative {
  f'''4 a, d, f,
}
```



Des lignes supplémentaires peuvent apparaître y compris au sein d'une portée, notamment lorsque vous l'avez personnalisée. L'exemple suivant illustre deux cas de figure quant au positionnement des lignes supplémentaires selon que la propriété `legder-position` est définie explicitement ou non. La présence du `\stopStaff` est ici rendue nécessaire pour annuler les effets de la commande `\override` qui s'applique à l'intégralité du `StaffSymbol`.

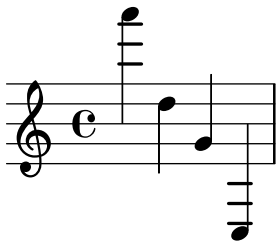
```
\override Staff.StaffSymbol.line-positions =   #'(-8 0 2 4)
d4 e f g
```

```
\stopStaff
\startStaff
\override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
d4 e f g
```



Modifier l'équidistance des lignes de la portée affectera aussi les lignes supplémentaires.

```
\new Staff \with {
  \override StaffSymbol.staff-space = #1.5
} \relative {
  f'''4 d, g, e,
}
```
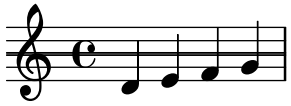


Morceaux choisis

Empâtement de certaines lignes d'une portée

Vous pourriez avoir envie, dans un but pédagogique, de rendre certaines lignes d'une portée plus épaisses que les autres, comme la ligne médiane, ou bien pour mettre en exergue la ligne portant la clef de sol. Il suffit pour cela d'ajouter une ligne qui sera accolée à celle qui doit être mise en évidence, grâce à la propriété `line-positions` de l'objet `StaffSymbol`.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



Voir aussi

Glossaire musicologique : Section "ligne" dans *Glossaire*, Section "ligne supplémentaire" dans *Glossaire*, Section "portée" dans *Glossaire*.

Manuel de notation : [Clefs], page 18.
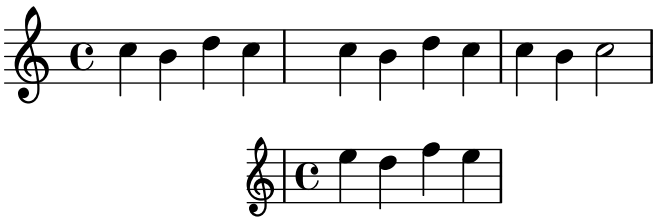
Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "StaffSymbol" dans *Référence des propriétés internes*, Section "staff-symbol-interface" dans *Référence des propriétés internes*.

Portées d'ossia

Une portée d'*ossia* – ou de variante – s'obtient en créant, à l'endroit approprié, une nouvelle portée simultanée :

```
\new Staff \relative {
  c''4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```

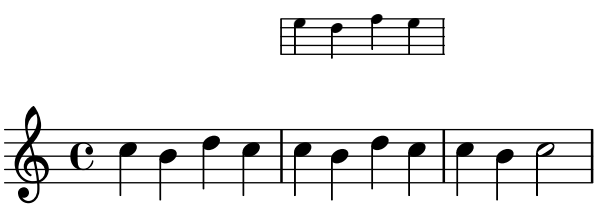


L'exemple ci-dessus n'est probablement pas ce qui vous conviendra le plus. Afin que cette ossia se place au-dessus de la portée à laquelle elle se réfère, étant par ailleurs dépourvue de métrique et de clef, et d'une taille légèrement inférieure, vous devrez avoir recours à quelques retouches. Le manuel d'initiation aborde une technique particulière pour obtenir ce résultat, au chapitre Section "Expressions musicales imbriquées" dans *Manuel d'initiation*.

L'exemple qui suit utilise, pour aligner la portée d'ossia, la propriété `alignAboveContext`. Cette méthode est tout à fait appropriée lorsqu'il y a un nombre restreint d'ossias.

```
\new Staff = "main" \relative {
  c''4 b d c
  <<
    { c4 b d c }

    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = "main"
      \magnifyStaff #2/3
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```
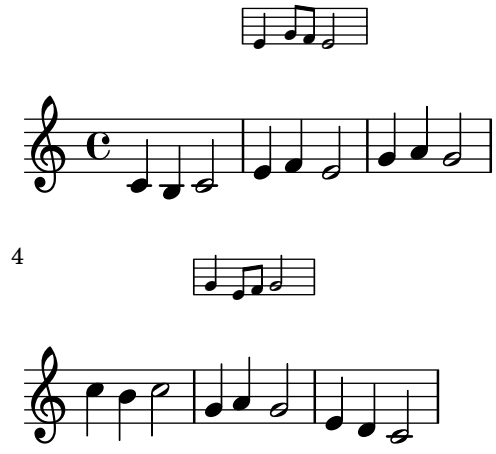


Dans le cas où de nombreux et courts fragments d'ossia affecteraient une même portée, il est judicieux de créer un contexte `Staff` vide auquel sera attribué un *identifiant*. Il suffira alors, pour

entamer un fragment d'ossia, de « faire appel » à ce contexte grâce aux commandes `\startStaff` et `\stopStaff`. Vous vous rendrez compte à l'utilisation des avantages que procure cette façon de procéder, bien plus que dans l'exemple suivant.

```
<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    \magnifyStaff #2/3
  }
  { \stopStaff s1*6 }
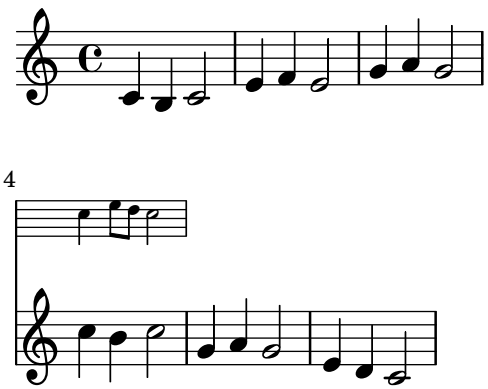
  \new Staff \relative {
    c'4 b c2
    <<
      { e4 f e2 }
      \context Staff = "ossia" {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = "ossia" {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```



Vous pourriez aussi recourir à la commande `\RemoveAllEmptyStaves` pour créer votre portée d'ossia. Cependant, cette méthode reste limitée à l'apparition de ces ossias en début de ligne. Pour plus d'information au sujet de la commande `\RemoveAllEmptyStaves`, reportez-vous au chapitre [Masquage de portées], page 212.

```
<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
```

```
    \hide Clef
    \magnifyStaff #2/3
    \RemoveAllEmptyStaves
  } \relative {
    R1*3
    c''4 e8 d c2
  }
  \new Staff \relative {
    c'4 b c2
    e4 f e2
    g4 a g2 \break
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>
```



Morceaux choisis
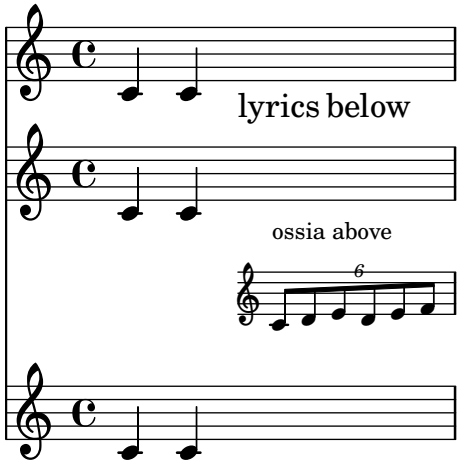
Positionnement d'une ossia et des paroles

Cet exemple illustre la manière de positionner une portée d'ossia et des paroles à l'aide des propriétés de contexte `alignBelowContext` et `alignAboveContext`.

```
\paper {
  ragged-right = ##t
}

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
```

```
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3
          c8[^"ossia above" d e d e f]
        }
      }
    >>
  }
>>
```



Voir aussi

Glossaire musicologique : Section “ossia” dans *Glossaire*, Section “portée” dans *Glossaire*, Section “Frenched staff” dans *Glossaire*.

Manuel d’initiation : Section “Expressions musicales imbriquées” dans *Manuel d’initiation*, Section “Longueur et épaisseur des objets” dans *Manuel d’initiation*, Section “Taille des objets” dans *Manuel d’initiation*.

Manuel de notation : [Masquage de portées], page 212.

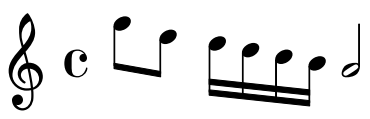
Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “StaffSymbol” dans *Référence des propriétés internes*.

Masquage de portées

Désactiver le graveur `Staff_symbol_engraver` dans un contexte `Staff` permet de masquer des lignes. La commande `\stopStaff` aura le même effet.

```
\new Staff \with {
  \remove "Staff_symbol_engraver"
}
\relative { a''8 f e16 d c b a2 }
```

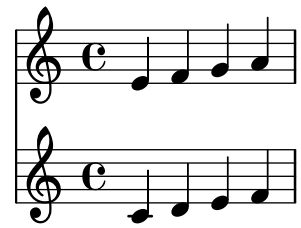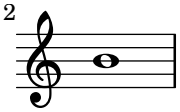


L’instruction `\RemoveEmptyStaves` placée dans un bloc `\layout` ou dans une clause `\with` affectant une portée particulière, aura pour effet de masquer toute portée qui ne contient rien. Dans

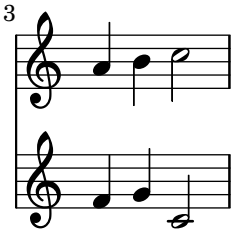
les partitions d'orchestre, les portées qui n'ont que des silences sont habituellement masquées afin de gagner de la place. Ce style d'édition s'appelle en anglais « Frenched Score ». Cette fonctionnalité masque et supprime toutes les portées vides d'une partition, hormis celles du premier système. Le premier système sera lui aussi concerné dès lors que sera utilisée l'instruction `\RemoveAllEmptyStaves`.

```
\layout {
  \context {
    \Staff
    \RemoveEmptyStaves
  }
}

\relative <<
  \new Staff {
    e'4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
>>
```







Une portée est considérée comme vide dès lors qu'elle ne contient que des silences multimesures, des silences visibles ou invisibles (ou d'espacement – les *\skip*) ou bien une combinaison de ces éléments. **Tous** les autres objets musicaux, ceux qui font qu'une portée ne sera pas considérée vide, sont répertoriés dans la propriété de contexte `keepAliveInterfaces` telle que définie dans le fichier `ly/engraver-init.ly`.
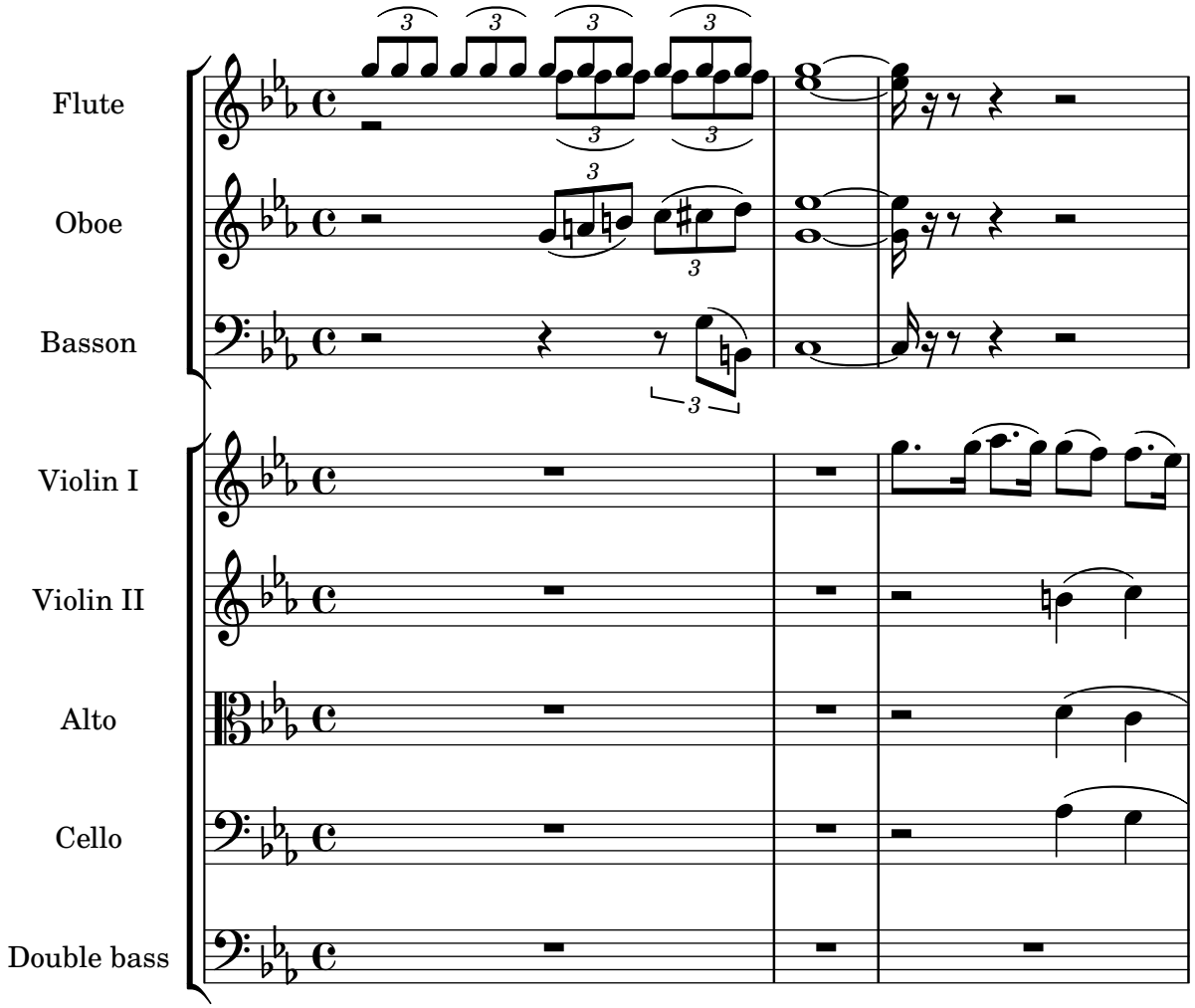
`\RemoveEmptyStaves` et `\RemoveAllEmptyStaves` sont tous deux des racourcis prédéfinis qui déterminent les propriétés telles que `remove-empty` et `remove-first`, attachées à l'objet
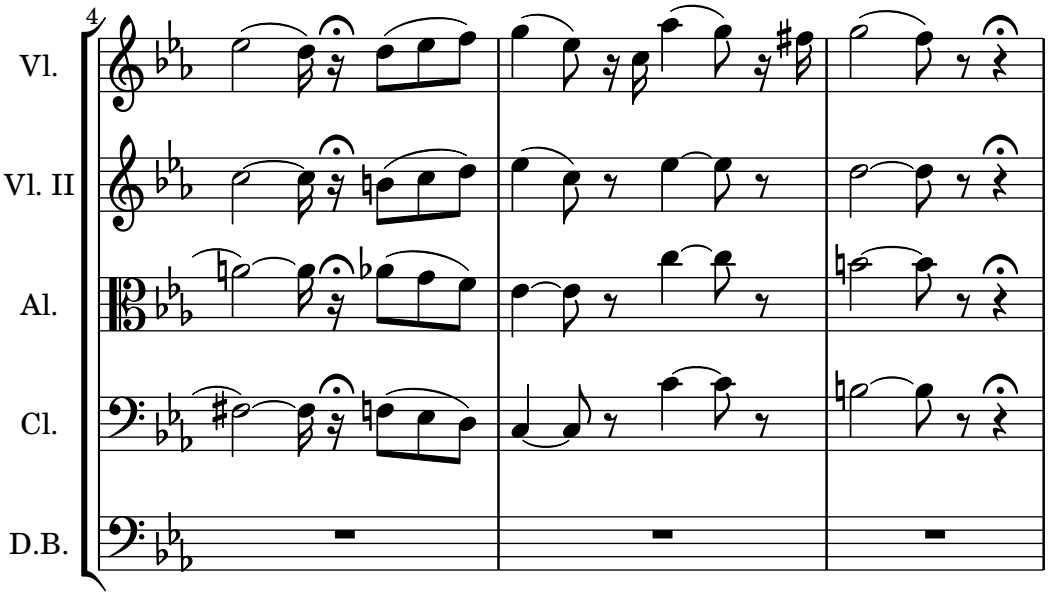
`VerticalAxisGroup`, comme indiqué dans Section A.20 [Identificateurs de modification de contexte], page 860.

Le `Keep_alive_together_engraver` permet de masquer l'intégralité d'un regroupement de portées plutôt que des portées individuelles. Il est normalement attaché au contexte `PianoStaff` : un système pianistique sera masqué uniquement dans le cas où les deux portées qui le composent sont vides. De la même manière, il est d'usage pour un conducteur d'orchestre de masquer les regroupements de portées plutôt que de manière individuelle. Ceci s'obtient en ajoutant le `Keep_alive_together_engraver` au regroupement de portées considéré, comme indiqué dans Section 5.1.4 [Modification des greffons de contexte], page 636, – voir [Regroupement de portées], page 199, pour leur dénomination.

```
\layout {
  \context {
    \StaffGroup
    \RemoveEmptyStaves
    \consists "Keep_alive_together_engraver"
  }
}
```

Dans l'exemple suivant, les portées des instruments à vent sont absentes du deuxième système, alors que celle de la contrebasse demeure présente puisqu'elle est rattachée au regroupement des cordes qui, elles, continuent de jouer.





En interne, le `Keep_alive_together_engraver` recourt à la propriété `remove-layer` du `VerticalAxisGroup` d'une portée pour décider de son impression lorsqu'elle est considérée vide. Cette propriété peut aussi se définir manuellement, auquel cas elle agit sous forme d'index de priorité : des valeurs proches de zéro seront prioritaires sur celles plus élevées, ce qui aura pour conséquence de masquer d'abord les portées dont le `remove-layer` est plus grand.

Ceci se révèle tout particulièrement utile pour une « division » de portée, lorsque plusieurs parties indépendantes – voir Section 1.6.3 [Écriture de parties séparées], page 217, – doivent momentanément se répartir sur plusieurs portées. Dans l'exemple qui suit, deux parties sont réparties sur *trois* portées ; ces trois portées n'apparaissent toutefois jamais simultanément :

- au premier système, une seule d'entre elles est affichée, dans la mesure où la propriété `keepAliveInterfaces` a été réglée sur une liste vide – de fait, les deux autres portées sont considérées comme vides et donc masquées, en dépit de ce qu'elles pourraient contenir ;
- lorsque cette propriété est désactivée – et retrouve donc ses réglages par défaut – elle n'empêche plus les deux autres portées d'être affichées. Cependant, et parce que le réglage de leur `remove-layer` est inférieur à celui de la portée unique, ces deux portées seront alors imprimées à la place.
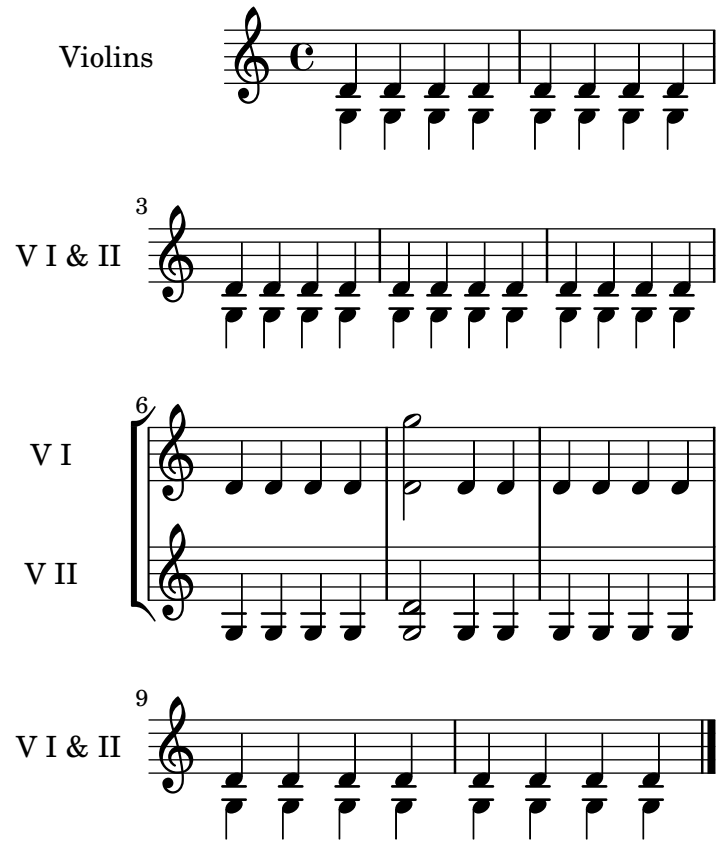
De telles substitutions ne s'appliquent pas seulement aux notes, accords ou autres éléments musicaux intervenant dès après le nouveau réglage, mais à l'intégralité du système où il est mentionné.

```
\layout {
  short-indent = 2\cm
  indent = 3\cm
  \context {
    \Staff
    keepAliveInterfaces = #'()
  }
}

violI = {
  \repeat unfold 24 { d'4 }
  \once \unset Staff.keepAliveInterfaces
  <d' g''>2
  \repeat unfold 14 { d'4 }
  \bar "|."
}
```

```
violII = {
  \repeat unfold 24 { g4 }
  <g d'>2
  \repeat unfold 14 { g4 }
  \bar "|."
}

\new StaffGroup \with { \consists "Keep_alive_together_engraver" } <<
  \new Staff \with {
    instrumentName = "Violins"
    shortInstrumentName = "V I & II"
    \override VerticalAxisGroup.remove-layer = 2
  } <<  \violI \\ \violII  >>
  \new Staff \with {
    instrumentName = "Violin I"
    shortInstrumentName = "V I"
    \RemoveAllEmptyStaves
    \override VerticalAxisGroup.remove-layer = 1
  } \violI
  \new Staff \with {
    instrumentName = "Violin II"
    shortInstrumentName = "V II"
    \RemoveAllEmptyStaves
    \override VerticalAxisGroup.remove-layer = 1
  } \violII
>>
```



`\RemoveAllEmptyStaves` permet aussi de gérer des fragments d'ossia attachés à une portée. Pour plus de détails, voir [Portées d'ossia], page 209.

Commandes prédéfinies

`\RemoveEmptyStaves`, `\RemoveAllEmptyStaves`,

Voir aussi

Glossaire musicologique : Section "Frenched staff" dans *Glossaire*.

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : [Dictée à trous], page 241, Section A.20 [Identificateurs de modification de contexte], page 860, Section 5.1.4 [Modification des greffons de contexte], page 636, Section 5.1.5 [Modification des réglages par défaut d'un contexte], page 638, [Portées d'ossia], page 209, [Regroupement de portées], page 199, [Silences invisibles], page 63, [Symbole de la portée], page 205, Section 5.4.7 [Visibilité des objets], page 675.

Fichiers d'initialisation : `ly/engraver-init.ly`.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Axis_group_engraver" dans *Référence des propriétés internes*, Section "ChordNames" dans *Référence des propriétés internes*, Section "FiguredBass" dans *Référence des propriétés internes*, Section "Keep_alive_together_engraver" dans *Référence des propriétés internes*. Section "Lyrics" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*, Section "Staff_symbol_engraver" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Supprimer le `Staff_symbol_engraver` aura pour effet de masquer les barres de mesure. Forcer leur visibilité peut entraîner des problèmes de formatage. En pareil cas il vaut mieux, au lieu de supprimer le graveur, recourir aux dérogations suivantes :

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

En ce qui concerne les désagréments et messages liés à l'utilisation de l'instruction `\RemoveEmptyStaves`, consultez Section 5.1.5 [Modification des réglages par défaut d'un contexte], page 638.

1.6.3 Écriture de parties séparées

Nous allons voir, au fil des lignes qui suivent, comment préparer un « matériel » pour orchestre ou ensemble musical, ce qui requiert souvent d'insérer des noms d'instrument dans une partition. Nous aborderons aussi la citation d'autres voix, et comment la mettre en forme, ainsi que le moyen de contracter une succession de mesures vides dans une partition de pupitre.
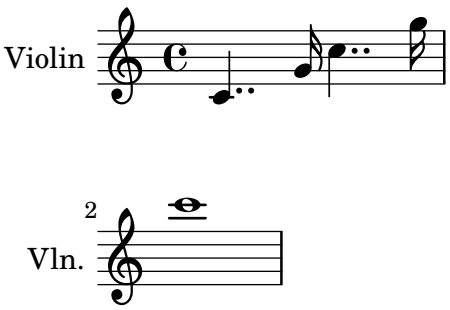
Par ailleurs, une méthode pour gérer la division d'un pupitre est abordée à la rubrique [Masquage de portées], page 212.

Noms d'instrument

Dans un conducteur, les noms d'instrument sont portés en regard de chacune des portées, qu'il s'agisse d'un contexte `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` ou `ChoirStaff` . La première ligne affichera la valeur de `instrumentName`, et les suivantes celle de `shortInstrumentName`.
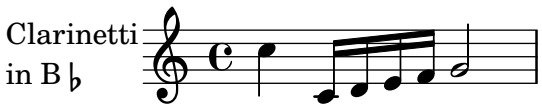
```
\new Staff \with {
  instrumentName = "Violin "
```

```
  shortInstrumentName = "Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}
```
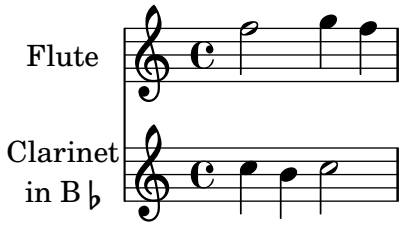


Le recours à la commande `\markup` permet de construire des noms d'instrument particuliers, tels que

```
\new Staff \with {
  instrumentName = \markup {
    \column { "Clarinetti"
      \line { "in B" \smaller \flat }
    }
  }
} \relative {
  c''4 c,16 d e f g2
}
```



Lorsque plusieurs contextes de portée sont regroupés, les noms d'instrument, que ce soit sous leur forme développée ou abrégée, sont par défaut centrés. Si l'un d'entre eux est libellé sur plusieurs lignes, il faudra recourir à l'instruction `\center-column` :
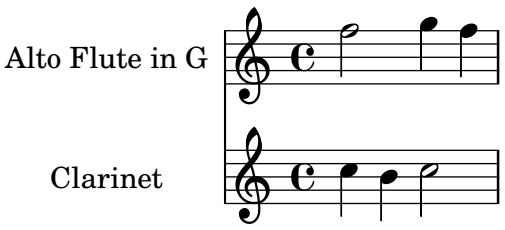
```
<<
  \new Staff \with {
    instrumentName = "Flute"
  }
  { f2 g4 f }
  \new Staff \with {
    instrumentName = \markup {
      \center-column { "Clarinet"
      \line { "in B" \smaller \flat }
      }
    }
  }
  { c4 b c2 }
>>
```
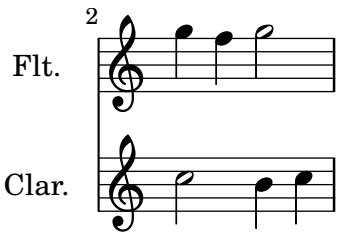


Lorsque le nom d'un instrument est relativement long, il est judicieux d'augmenter les retraits – *indent* – au sein du bloc `\layout` à l'aide des commandes `indent` et `short-indent`. Pour plus de plus amples détails sur ces réglages, reportez-vous au chapitre [Variables d'indentation et de décalage], page 579.

```
<<
  \new Staff \with {
    instrumentName = "Alto Flute in G"
    shortInstrumentName = "Flt."
  } \relative {
    f''2 g4 f \break
    g4 f g2
  }
  \new Staff \with {
    instrumentName = "Clarinet"
    shortInstrumentName = "Clar."
  } \relative {
    c''4 b c2 \break
    c2 b4 c
  }
>>

\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm
}
```





Des noms d'instrument peuvent s'utiliser dans d'autres contextes, tels que `ChordNames` ou `FiguredBass`, dès lors qu'on leur adjoint le graveur `Instrument_name_engraver`. Pour de plus amples informations sur la manière d'activer ou désactiver un graveur, voir Section 5.1.4 [Modification des greffons de contexte], page 636.

Le nom abrégé d'un instrument (`shortInstrumentName`) peut changer en cours de morceau, en même temps que les autres réglages propres au nouvel instrument. Notez cependant que la valeur de `instrumentName` ne s'affichera que sur la première portée :

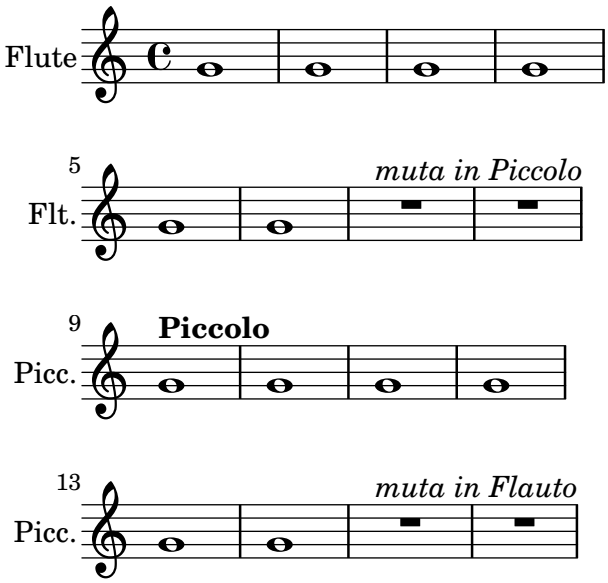
```
prepPiccolo = <>^\markup \italic { muta in Piccolo }

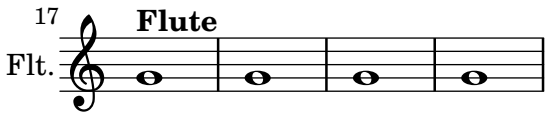
prepFlute = <>^\markup \italic { muta in Flauto }

setPiccolo = {
  <>^\markup \bold { Piccolo }
  \transposition c''
}

setFlute = {
  <>^\markup \bold { Flute }
  \transposition c'
}

\new Staff \with {
  instrumentName = "Flute"
  shortInstrumentName = "Flt."
}
\relative {
  g'1 g g g \break
  g1 g \prepPiccolo R R \break
  \set Staff.instrumentName = "Piccolo"
  \set Staff.shortInstrumentName = "Picc."
  \setPiccolo
  g1 g g g \break
  g1 g \prepFlute R R \break
  \set Staff.instrumentName = "Flute"
  \set Staff.shortInstrumentName = "Flt."
  \setFlute
  g1 g g g
}
```





Voir aussi

Manuel de notation : Section 5.1.4 [Modification des greffons de contexte], page 636, [Variables d'indentation et de décalage], page 579.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "InstrumentName" dans *Référence des propriétés internes*, Section "PianoStaff" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*.

Citation d'autres voix

Il est assez courant qu'une voix soit doublée par une autre. Par exemple, les premiers et seconds violons peuvent jouer les mêmes notes durant un moment. LilyPond gère parfaitement ces situations où une voix est la *réplique* d'une autre, sans devoir ressaisir la musique en question.

L'instruction `\addQuote`, placée au niveau le plus haut – c'est à dire en dehors de tout bloc de musique – définit la musique dont il sera possible de répliquer des fragments.
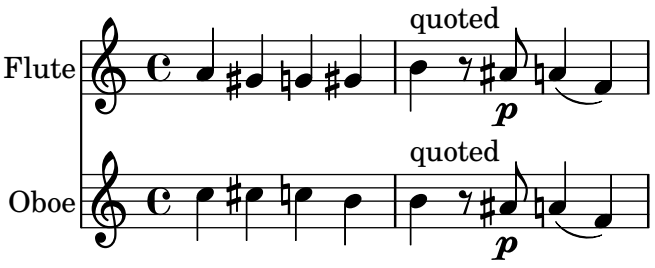
Au cours d'une partie, des extraits de répliques peuvent être cités en utilisant la commande `\quoteDuring`. Cette commande prend deux arguments : le nom de la voix reproduite, tel que défini par `\addQuote`, et une expression musicale qui indique la durée de cette citation.

```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
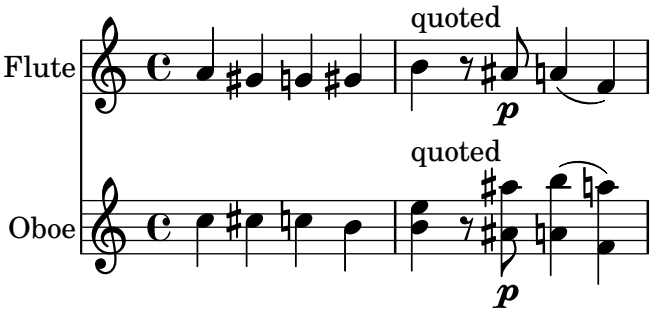


Si l'expression musicale utilisée pour `\quoteDuring` contenait autre chose que du silence, il en résulterait une situation polyphonique, ce qui n'est pas le but recherché :

```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
 }
```

```
oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { e4 r8 ais b4 a }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```



Lorsqu'une commande `\unfoldRepeats` est requise dans une expression musicale de telle sorte que la musique soit réimprimée par `\quoteDuring`, l'instruction `\addQuote` doit elle-mêeme contenir la commande `\unfoldRepeats` :

```
fluteNotes = \relative {
  \repeat volta 2 { a'4 gis g gis }
}

oboeNotesDW = \relative {
  \repeat volta 2 \quoteDuring "incorrect" { s1 }
}
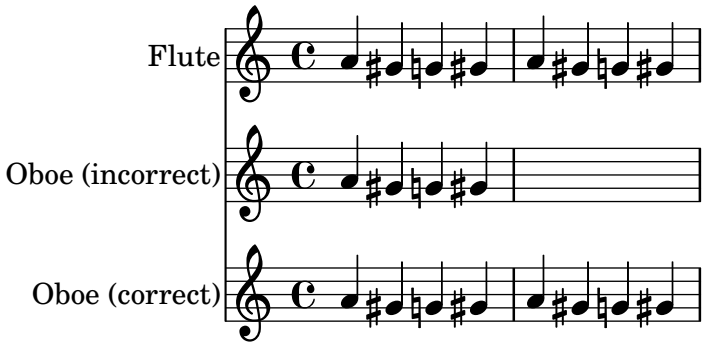
oboeNotesW = \relative {
  \repeat volta 2 \quoteDuring "correct" { s1 }
}


\addQuote "incorrect" { \fluteNotes }

\addQuote "correct" { \unfoldRepeats \fluteNotes }

\score {
  \unfoldRepeats
  <<
    \new Staff \with { instrumentName = "Flute" }
    \fluteNotes
    \new Staff \with { instrumentName = "Oboe (incorrect)" }
    \oboeNotesDW
    \new Staff \with { instrumentName = "Oboe (correct)" }
    \oboeNotesW
```
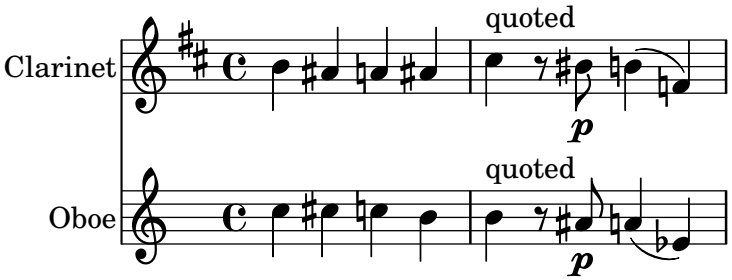
```
  >>
}
```



L'instruction `\quoteDuring` prendra en compte les réglages d'une commande `\transposition`, qu'elle apparaisse au niveau de la voix répliquée ou dans celle qui réplique.

```
clarinetNotes = \relative c'' {
  \transposition bes
  \key d \major
  b4 ais a ais | cis4^"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "clarinet" { s1 }
 }
\addQuote "clarinet" { \clarinetNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```



LilyPond répliquera, par défaut, tous les éléments – articulations, nuances, *markups*, etc. La propriété `quotedEventTypes` permet de définir plus précisément quels éléments de la voix originelle seront reproduits.
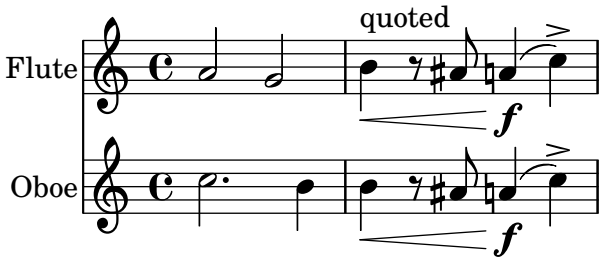
```
fluteNotes = \relative {
  a'2 g2 |
  b4\<^"quoted" r8 ais a4\f( c->)
 }

oboeNotes = \relative {
```

```
  c''2. b4 |
  \quoteDuring "flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \set Score.quotedEventTypes = #'(note-event articulation-event
                                     crescendo-event rest-event
                                     slur-event dynamic-event)
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
 }
```



Les citations peuvent être « balisées » par un nom particulier afin de les utiliser de différentes manières. Pour de plus amples détails à ce propos, consultez le chapitre [Utilisation de balises], page 544.

Voir aussi

Manuel de notation : [Instruments transpositeurs], page 28, [Utilisation de balises], page 544.

Fichiers d'initialisation : `scm/define-event-classes.scm`.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Music classes" dans *Référence des propriétés internes*, Section "QuoteMusic" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Seul le contenu de la première `Voice` rencontrée dans la partie marquée d'une commande `\addQuote` pourra être retenu. Par voie de conséquence, ***musique*** ne saurait comprendre de `\new` ou une instance `\context Voice` qui la ferait passer à une autre voix.

Citer des notes d'ornement ne fonctionne pas, et peut même entraîner un blocage de LilyPond.
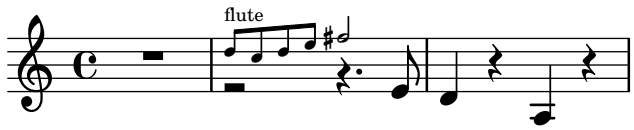
Citer des triolets imbriqués peut entraîner un résultat de piètre qualité.

Mise en forme d'une citation

Le moyen le plus simple pour mettre en forme des notes provenant d'une autre voix consiste à déclarer explicitement un contexte `CueVoice` au sein de la voix où elle apparaît.
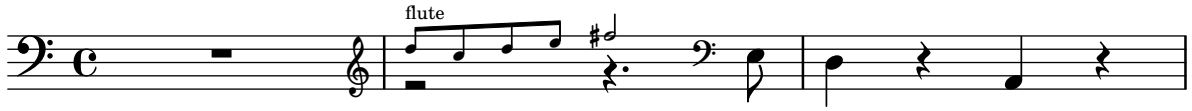
```
\relative {
  R1
  <<
    { e'2\rest r4. e8 }
```

```
    \new CueVoice {
      \stemUp d'8^"flute" c d e fis2
    }
  >>
  d,4 r a r
}
```
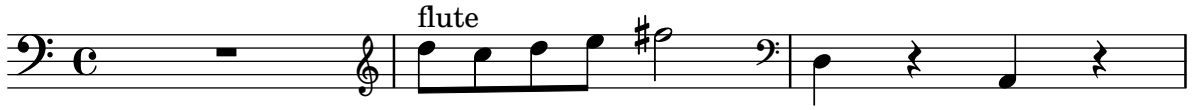


L'instruction `\cueClef`, utilisée conjointement à un contexte `CueVoice` explicite permet d'indiquer la clef, dans une taille réduite, propre à la voix citée. Le retour à la clef d'origine s'effectue à l'aide de l'instruction `\cueClefUnset`.

```
\relative {
  \clef "bass"
  R1
  <<
    { e'2\rest r4. \cueClefUnset e,8 }
    \new CueVoice {
      \cueClef "treble" \stemUp d''8^"flute" c d e fis2
    }
  >>
  d,,4 r a r
}
```



Notez que les deux instructions `\cueClef` et `\cueClefUnset` sont disponibles si nécessaire en dehors d'un `CueVoice`.

```
\relative {
  \clef "bass"
  R1
  \cueClef "treble"
  d''8^"flute" c d e fis2
  \cueClefUnset
  d,,4 r a r
}
```



Lorsque la situation est plus complexe, instrument transpositeur ou citations de plusieurs sources, vous disposez des instructions `\cueDuring` et `\cueDuringWithClef`, versions spécifiques de la commande `\quoteDuring` – voir la rubrique précédente ([Citation d'autres voix], page 221).

Leur syntaxe est :

`\cueDuring` *`origine #position musique`*

et

\cueDuringWithClef *origine #position #clef musique*

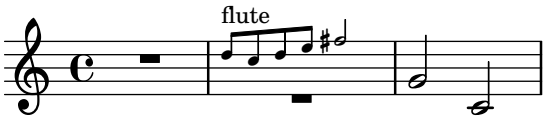
Des mesures issues de la partie d'*origine* seront recopiées dans un contexte de `CueVoice` et synchronisées avec *musique* – habituellement un silence. L'apparition des petites notes initialise une polyphonie temporaire pour la portée concernée. L'argument *position* détermine si ces petites notes seront attachées à la première ou à la seconde voix – `UP` pour la première, `DOWN` pour la seconde.

```
fluteNotes = \relative {
  r2. c''4 | d8 c d e fis2 | g2 d |
}

oboeNotes = \relative c'' {
  R1
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}
```

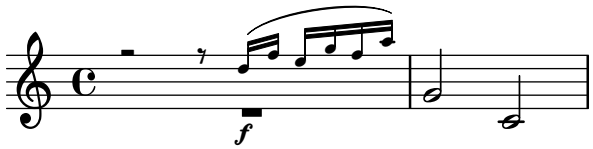


La propriété `quotedCueEventTypes` permet de définir précisément quels éléments de la voix originelle seront reproduits. Sa valeur par défaut est `'(note-event rest-event tie-event beam-event tuplet-span-event)`. LilyPond reproduira donc les notes, silences, liaisons de prolongation, ligatures et n-olets, mais pas les articulations, annotations ni nuances.

> **Note :** Dans l'exemple précédent, il était nécessaire de déclarer explicitement le contexte `Voice`, sinon l'intégralité de l'expression musicale se serait retrouvée dans le contexte `CueVoice`.

```
oboeNotes = \relative {
  r2 r8 d''16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboeNotes }

\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring "oboe" #UP { R1 }
  g2 c,
}
```
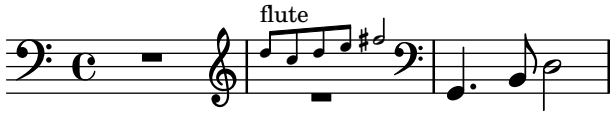


Le nom de l'instrument qui est répliqué peut s'indiquer à l'aide d'un *markup*. Par ailleurs, si la citation nécessite l'apparition d'une clef différente, celle-ci devra être introduite manuellement, tout comme l'originale qui devra être rappelée en fin de citation.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \clef treble
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



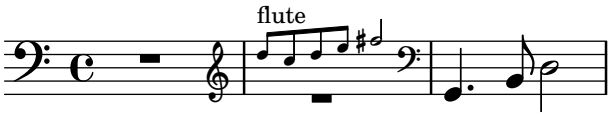
L'instruction `\cueDuringWithClef` se chargera quant à elle, et grâce à un argument supplémentaire, de gérer le changement de clef nécessaire à la citation et le retour à la clef originelle.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  <>^\markup { \tiny "flute" }
  \cueDuringWithClef "flute" #UP "treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



L'instruction `\cueDuring`, à l'instar de la commande `\quoteDuring`, tient compte des instruments transpositeurs. La citation s'effectue aux hauteurs correspondant à l'instrument où elles apparaissent.

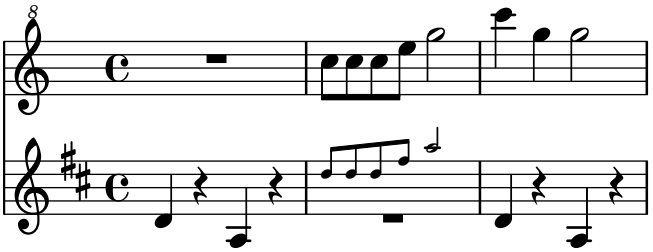
L'instruction `\transposedCueDuring` est particulièrement adaptée pour des instrument ayant une tessiture éloignée, comme dans le cas d'un piccolo cité dans une partie de clarinette basse. Sa syntaxe est identique à celle de `\cueDuring`, à ceci près qu'elle nécessite un argument supplémentaire afin de spécifier la transposition à effectuer en hauteur absolue.

```
piccoloNotes = \relative {
  \clef "treble^8"
  R1
  c'''8 c c e g2
  c4 g g2
}

bassClarinetNotes = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring "piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccoloNotes }

<<
  \new Staff \piccoloNotes
  \new Staff \bassClarinetNotes
>>
```



La commande `killCues` permet de supprimer les notes d'une citation. Ceci est utile lorsque cette citation n'est pas imprimée dans le conducteur entre autres. `killCues` supprimera les notes et autres événements pris en charge par `\cueDuring`. Pour les autres annotations telles que changement de clef ou instrument concerné, faites appel à des balises – voir [Utilisation de balises], page 544, à ce sujet.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

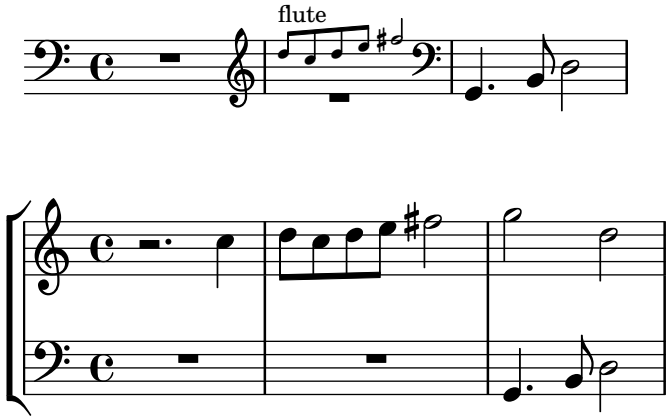
bassoonNotes = \relative c {
  \clef bass
  R1
  \tag #'part {
```

```
    \clef treble
    <>^\markup \tiny { flute }
  }
  \cueDuring "flute" #UP { R1 }
  \tag #'part \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>
```



Voir aussi

Manuel de notation : [Citation d'autres voix], page 221, [Citation-repère], page 335, [Clefs], page 18, [Instruments transpositeurs], page 28, [Noms d'instrument], page 217, [Utilisation de balises], page 544.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "CueVoice" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

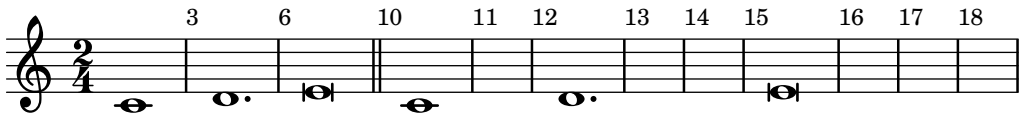
La commande `\cueDuring` ne sait pas gérer les collisions de silence entre les contextes `Voice` et `CueVoice`.

Dans le cadre d'un `\cueDuringWithClef` ou d'un `\transposedCueDuring`, l'argument supplémentaire doit intervenir après l'origine et la position.

Compression de mesures vides

Toutes les mesures sont par défaut imprimées, même si elles sont vides ; ceci peut se produire dans le cas où un événement rythmique (note, silence ou saut) dure au point de s'étaler sur plusieurs mesures. Ce comportement peut se modifier de sorte que les mesures vides sont compressées en une seule mesure comme ici (la deuxième partie de cet exemple, où les mesures sont expansées, retrouve en fait le comportement par défaut) :

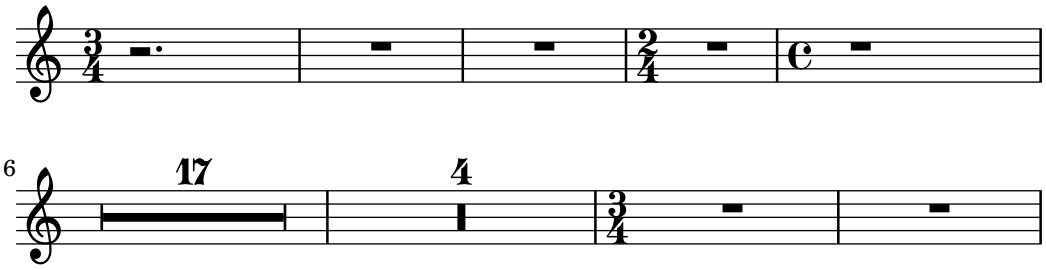
```
\override Score.BarNumber.break-visibility = ##(#f #t #t)
\time 2/4
\compressEmptyMeasures
c'1 d'1. e'\breve
\bar "||"
\expandEmptyMeasures
c'1 d'1. e'\breve
```



Bien que la syntaxe de cette notation soit correcte, elle peut être déroutante d'un point de vue musical, ce qu'illustre l'exemple ci-dessus. C'est pourquoi l'impression des numéros de mesure devient nécessaire, en suivant la syntaxe indiquée dans [Utilisation de break-visibility], page 677.

Une telle notation peut toutefois s'avérer utile lorsqu'elle s'applique à des Section "silences valant une mesure" dans *Manuel de notation*. Un silence de plusieurs mesures sera alors affiché sous la forme d'une seule mesure contenant un symbole de silence multimesure surplombé du nombre de mesures de silence :

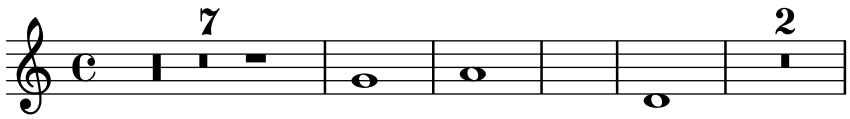
```
% Comportement par défaut
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Mesures de silence comprimées en une seule mesure
\compressEmptyMeasures
r1 | R1*17 | R1*4 |
\expandEmptyMeasures
% Mesures de silence à nouveau expansées
\time 3/4
R2.*2 |
```



Contrairement à `\compressEmptyMeasures`, la fonction musicale `\compressMMRests` s'applique exclusivement aux silences tout en laissant les autres événements expansés. En sa qualité de fonction, et non de réglage de propriété, sa syntaxe diffère quelque peu, en ceci qu'elle doit être suivie d'une expression musicale :

```
\compressMMRests {
```

```
  % Les silences sont comprimés...
  R1*7
  % ... mais les notes peuvent s'étaler sur plusieurs mesures.
  g'1 a'1*2 d'1
  R1*2
}
```



Toutes les commandes décrites dans ces paragraphes reposent en fait sur la propriété interne `skipBars`, qui se définit au sein du contexte `Score` comme indiqué dans Section 5.3.2 [La commande de fixation \set], page 652.

Commandes prédéfinies

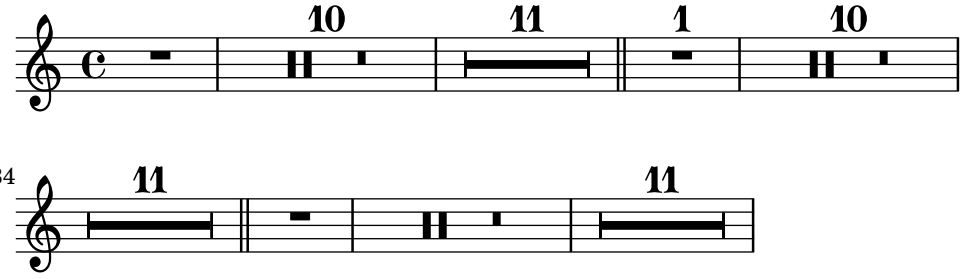
`\compressEmptyMeasures`, `\expandEmptyMeasures`, `\compressMMRests`.

Morceaux choisis

Dénombrer une unique mesure de silence

Les silences multimesures affichent leur longueur sauf s'il n'y a qu'une seule mesure. Ceci peut se modifier en réglant `restNumberThreshold`.

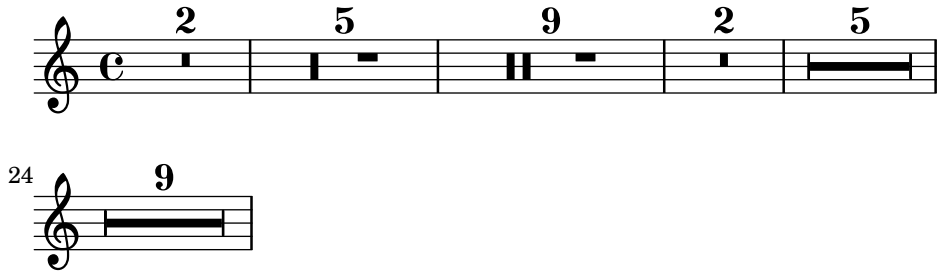
```
{
  \compressEmptyMeasures
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 0
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 10
  R1 R1*10 R1*11
}
```



Modifier l'apparence d'un silence multimesure

Dans le cas où ce silence dure moins de dix mesures, LilyPond imprime sur la portée des « silences d'église » – *Kirchenpause* en allemand – et qui sont une simple suite de rectangles. La propriété `expand-limit` permet d'obtenir un silence unique :

```
\relative c'' {
  \compressMMRests {
    R1*2 | R1*5 | R1*9
    \override MultiMeasureRest.expand-limit = #3
    R1*2 | R1*5 | R1*9
  }
}
```

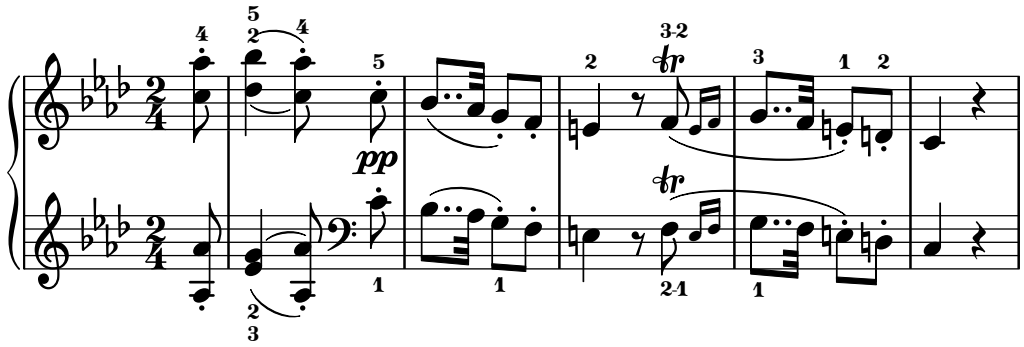


Voir aussi

Manuel de notation : Section 5.3.2 [La commande de fixation `\set`], page 652, [Silences valant une mesure], page 65, [Utilisation de break-visibility], page 677.

Référence des propriétés internes : Section “MultiMeasureRest” dans *Référence des propriétés internes*, Section “MultiMeasureRestNumber” dans *Référence des propriétés internes*, Section “MultiMeasureRestScript” dans *Référence des propriétés internes*, Section “MultiMeasureRestText” dans *Référence des propriétés internes*.

1.7 Annotations éditoriales



Ce chapitre traite de la manière de modifier l’apparence des notes dans un but pédagogique ou d’analyse.

1.7.1 Dans la portée

Nous allons voir ici comment mettre en exergue certains éléments à l’intérieur même de la portée.
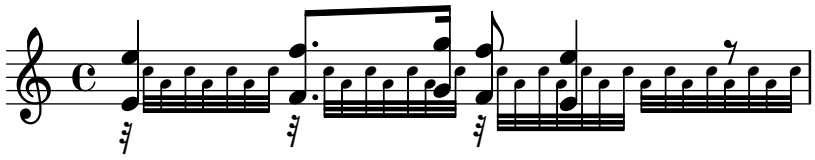
Indication de la taille de fonte musicale

> **Note :**
> Pour les tailles de police textuelle, voir [Sélection de la fonte et de la taille], page 265.
> Pour la taille des portées, voir Section 4.2.2 [Définition de la taille de portée], page 585.
> Pour les petites notes, voir [Mise en forme d’une citation], page 224.
> Pour les variantes, voir [Portées d’ossia], page 209.

Le plus sûr moyen de régler la taille des éléments de notation sans modifier la taille de la portée consiste à utiliser la commande `\magnifyMusic` :

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
```

```
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```

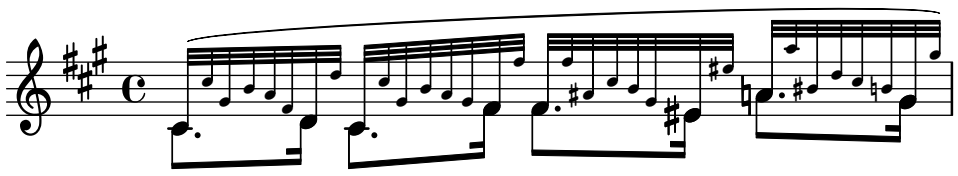


La présence d'un `\override` dans cet exemple permet de contourner une bogue – voir « Problèmes connus et avertissements » en fin de section.

Dans le cas de fusion d'une tête de note normale avec une autre de taille inférieure, la taille de la petite note peut nécessiter une réinitialisation – à l'aide d'un '`\once \normalsize`' – de telle sorte que les hampes et altérations s'alignent correctement :

```
\new Staff <<
  \key fis \minor
  \mergeDifferentlyDottedOn
  \new Voice \relative {
    \voiceOne
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment =
        #(* 1.2 0.63)

      \once \normalsize cis'32( cis' gis b a fis
        \once \normalsize d d'
      \once \normalsize cis, cis' gis b a gis
        \once \normalsize fis fis'
      \once \normalsize fis, fis' ais, cis b gis
        \once \normalsize eis eis'
      \once \normalsize a, a' bis, d cis b
        \once \normalsize gis gis')
    }
  }
  \new Voice \relative {
    \voiceTwo
    cis'8. d16 cis8. fis16 fis8. eis16 a8. gis16
  }
>>
```



La commande `\magnifyMusic` n'est pas conçue pour gérer les citations, notes d'ornement ou portées d'ossia – des moyens spécifiques sont déjà disponibles en la matière. Elle est par contre
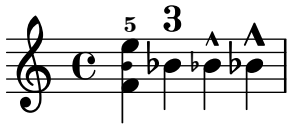
tout à fait adaptée dans le cas d'un instrument particulier disposant de sa propre portée là où des notes d'ornement seraient inappropriées, comme pour une pseudo-cadence ou les exemples ci-dessus. L'attribution d'une valeur de 0,63 à `\magnifyMusic` duplique les dimensions d'un contexte `CueVoice`.

> **Note :** La commande `\magnifyMusic` n'est pas censée intervenir en complément d'une modification de la taille de portée – voir Section 4.2.2 [Définition de la taille de portée], page 585.

Redimmensionnement individuel d'objets de rendu

L'ajustement de la propriété `font-size` à l'aide des commandes `\tweak` ou `\override` permet de retailler un objet de rendu particulier :

```
\relative {
  % resize a note head
  <f' \tweak font-size -4 b e>-5
  % resize a fingering
  bes-\tweak font-size 0 -3
  % resize an accidental
  \once \override Accidental.font-size = -4 bes!-^
  % resize an articulation
  \once \override Script.font-size = 4 bes!-^
}
```
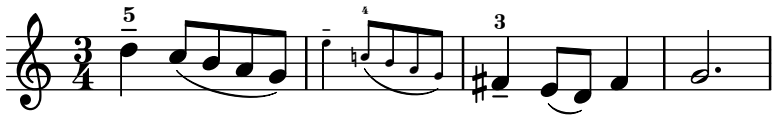


La valeur par défaut de `font-size` est mentionnée, pour chaque objet de rendu, dans la référence des propriétés internes. La propriété `font-size` ne peut intervenir que pour les objets qui utilisent des polices, autrement dit ceux qui disposent de l'interface de rendu `font-interface`. L'abence de `font-size` dans la liste des « réglages par défaut » d'un objet signifie qu'elle est de 0 – voir Section "All layout objects" dans *Référence des propriétés internes (Tous les objets de rendu).*

La propriété `fontSize`

La propriété `fontSize` d'un contexte a pour effet de définir la taille proportionnelle de tout élément de notation basé sur un glyphe dans ce contexte :

```
\relative {
  \time 3/4
  d''4---5 c8( b a g) |
  \set fontSize = -6
  e'4-- c!8-4( b a g) |
  \set fontSize = 0
  fis4---3 e8( d) fis4 |
  g2.
}
```



La valeur de `fontSize` est un nombre indiquant la taille relative par rapport à la hauteur standard de la portée en cours. La valeur par défaut de `fontSize` est de 0. Un pas de six aboutit exactement au doublement de la taille ; un pas de moins six la réduit de moitié. Chaque incrément correspond à une augmentation d'environ 12 % de la taille de la police.

La fonction Scheme `magnification->font-size` permet de s'affranchir de l'échelle logarithmique peu intuitive des unités de la propriété `font-size`. Ainsi, l'ajustement à 75 % de la notation musicale par rapport à la taille de la portée peut se libeller :

```
\set fontSize = #(magnification->font-size 0.75)
```

La fonction Scheme `magstep` quant à elle effectue l'inverse : elle convertit le nombre affecté à `font-size` en facteur d'échelle.
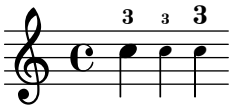
La propriété `fontSize` affecte uniquement les éléments de notation reposant sur des glyphes, tels les têtes de note, altérations, scripts, etc. Elle n'aura aucun effet sur la taille de la portée, la hauteur des hampes ou la longueur des ligatures ni sur l'espacement horizontal. L'échelonnement des hampes, ligature et espacement horizontal, couplé à la taille de la notation (sans modification de la taille de la portée), s'obtient à l'aide de la commande `\magnifyMusic` comme nous venons de le voir. La modification de l'ensemble, y compris la taille de portée, est abordé dans Section 4.2.2 [Définition de la taille de portée], page 585.

Dès lors que la **propriété de contexte** `fontSize` est définie, sa valeur est ajoutée individuellement à la valeur de la **propriété de** *grob* `font-size` des objets de rendu. Ceci peut être source de confusion lorsque des propriétés `font-size` individuelles sont réglées alors que `fontSize` est déjà fixé :

```
% the default font-size for NoteHead is 0
% the default font-size for Fingering is -5
c''4-3

\set fontSize = -3
% the effective font size for NoteHead is now -3
% the effective font size for Fingering is now -8
c''4-3

\override Fingering.font-size = 0
% the effective font size for Fingering is now -3
c''4-3
```
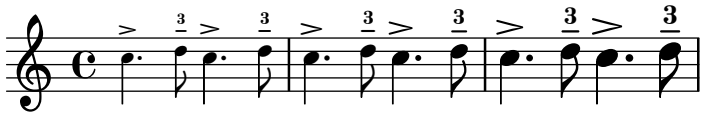


LilyPond dispose d'un certain nombre de raccourcis :

Commande	**Équivalent**	**Taille relative**
`\teeny`	`\set fontSize = -3`	71 %
`\tiny`	`\set fontSize = -2`	79 %
`\small`	`\set fontSize = -1`	89 %
`\normalsize`	`\set fontSize = 0`	100 %
`\large`	`\set fontSize = 1`	112 %
`\huge`	`\set fontSize = 2`	126 %

```
\relative c'' {
  \teeny
  c4.-> d8---3
  \tiny
```

```
  c4.-> d8---3
  \small
  c4.-> d8---3
  \normalsize
  c4.-> d8---3
  \large
  c4.-> d8---3
  \huge
  c4.-> d8---3
}
```



Pour changer la taille des caractères, LilyPond met à l'échelle la fonte dont la taille est la plus proche de la taille voulue. La taille standard (pour laquelle `font-size = 0`) dépend de la hauteur de la portée. À une portée de 20 points correspond une police de 11 points.

Commandes prédéfinies

`\magnifyMusic`, `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Voir aussi

Manuel de notation : Section 4.2.2 [Définition de la taille de portée], page 585, [Mise en forme d'une citation], page 224, [Portées d'ossia], page 209, [Sélection de la fonte et de la taille], page 265.

Fichiers d'initialisation : `ly/music-functions-init.ly`, `ly/property-init.ly`.

Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "font-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements
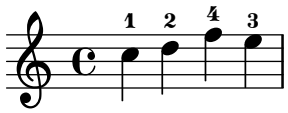
Deux bogues actuellement répertoriés font obstacle à un espacement horizontal correct avec `\magnifyMusic`. La seule façon de les contourner n'est cependant pas garantie dans toutes les situations. Dans l'exemple suivant, il vous faudra adapter la valeur de la variable *mag*. Vous pourriez aussi tenter de supprimer une ou les deux commandes `\newSpacingSection`, ou les commandes `\override` et `\revert` :

```
\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [musique]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}
```

Doigtés
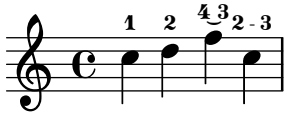
Les doigtés peuvent être indiqués comme suit : *note-chiffre_du_doigt*

```
\relative { c''4-1 d-2 f-4 e-3 }
```



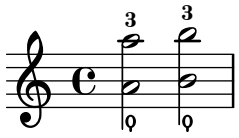
Pour les substitutions de doigts, on a recours à une indication textuelle (commande `\markup`) de doigté (commande `\finger`).

```
\relative {
  c''4-1 d-2 f\finger \markup \tied-lyric "4~3" c\finger "2 - 3"
}
```



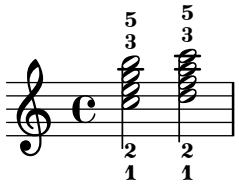
La commande `\thumb` peut être utilisée pour indiquer, par exemple dans une partition de violoncelle, si une note doit être jouée avec le pouce (*thumb* en anglais).

```
\relative { <a'_\thumb a'-3>2 <b_\thumb b'-3> }
```



Les doigtés des accords peuvent être saisis note par note, en les indiquant après chaque hauteur de note.

```
\relative {
  <c''-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
}
```



Les indications de doigtés peuvent se placer au-dessus ou en dessous de la portée – voir Section 5.4.2 [Direction et positionnement], page 667, à ce sujet.

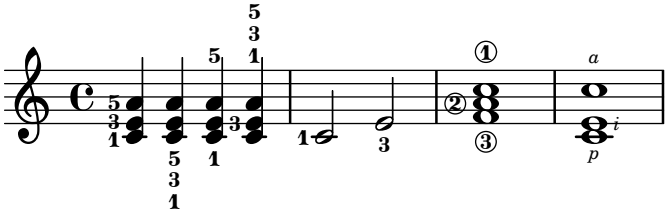
Morceaux choisis

Contrôler la position des doigtés dans un accord

Le positionnement des doigtés peut être contrôlé de manière très précise. Afin que l'orientation soit prise en compte, il est nécessaire d'utiliser une syntaxe d'accord `< >`, même s'il ne s'agit que d'une seule note. Le positionnement des numéros de corde et doigtés main droite se règle de manière analogue.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
```
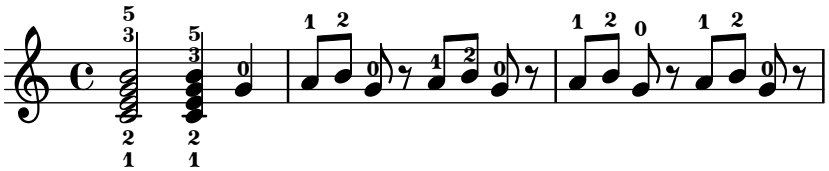
```
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger #1 e\rightHandFinger #2 c'\rightHandFinger #4 >
}
```



Impression des doigtés à l'intérieur de la portée

L'empilement des indications de doigté se fait par défaut à l'extérieur de la portée. Il est néanmoins possible d'annuler ce comportement. Une attention particulière doit toutefois être portée dans les cas où doigté et hampe vont dans la même direction : les indications de doigté n'évitent les hampe qu'en présence de ligature. Ce réglage peut s'adapter pour éviter toutes les hampes ou aucune d'elles. L'exemple suivant illustre ces deux options, ainsi que la manière de revenir au comportement par défaut.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 g'-0
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
}
```



Voir aussi

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667.

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

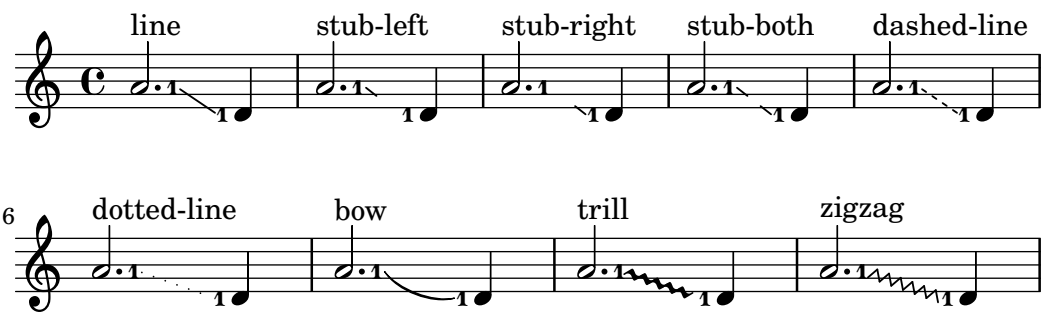
Référence des propriétés internes : Section “Fingering” dans *Référence des propriétés internes*, Section “Fingering_engraver” dans *Référence des propriétés internes*, Section “fingering-event” dans *Référence des propriétés internes*, Section “FingeringEvent” dans *Référence des propriétés internes*, Section “New_fingering_engraver” dans *Référence des propriétés internes*.

Glissement de doigt

En matière d'instrument à cordes, un doigt que l'on fait glisser s'indique souvent à l'aide d'un trait reliant le même doigt utilisé pour plusieurs notes jouées à des positions différentes sur une même corde. Cette ligne s'entame par un `\glide` placé avant l'instruction de doigté et se terminera à l'occurrence suivante du même doigt. Cette ligne peut adopter différents aspects.

```
mus = {
  \set fingeringOrientations = #'(right)
  <a'\glide-1>2.
  \set fingeringOrientations = #'(left)
  <d'-1>4
}

{
  <>^"line"
  \mus
  <>^"stub-left"
  \override FingerGlideSpanner.style = #'stub-left
  \mus
  <>^"stub-right"
  \override FingerGlideSpanner.style = #'stub-right
  \mus
  <>^"stub-both"
  \override FingerGlideSpanner.style = #'stub-both
  \mus
  <>^"dashed-line"
  \override  FingerGlideSpanner.style = #'dashed-line
  \mus
  \break
  <>^"dotted-line"
  \override FingerGlideSpanner.style = #'dotted-line
  \mus
  <>^"bow"
  \override FingerGlideSpanner.style = #'bow
  \mus
  <>^"trill"
  \override FingerGlideSpanner.style = #'trill
  \mus
  <>^"zigzag"
  \override FingerGlideSpanner.style = #'zigzag
  \mus
}
```

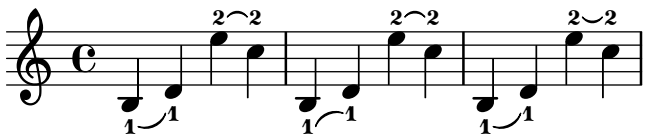


Dès lors que le `style` est défini à `'bow`, le positionnement de l'arc s'ajuste à l'aide des modificateurs d'orientation.

```
{
  \override FingerGlideSpanner.style = #'bow
  \set fingeringOrientations = #'(down)
  <b\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''\glide-2> <c''-2>

  \set fingeringOrientations = #'(down)
  <b^\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''^\glide-2> <c''-2>

  \set fingeringOrientations = #'(down)
  <b_\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''_\glide-2> <c''-2>
}
```



Dès lors que le `Finger_glide_engraver` est déplacé dans le contexte `Staff`, les objets `Fingering` de différents contextes `Voice` pourront être reliés.

```
\score {
  \new Staff <<
    \new Voice {
      \voiceOne
      \set fingeringOrientations = #'(right)
      <e''-3>2
      \set fingeringOrientations = #'(left)
      <d''-\tweak bound-details.left.padding #2.5 \glide-2>
      <c''-2>
      \bar "||"
    }
    \new Voice {
      \voiceTwo
      \set fingeringOrientations = #'(right)
      <c''\glide-2>
      \set fingeringOrientations = #'(left)
      <b'-\tweak bound-details.left.padding #2.5 \glide-4>
      <a'-4>
    }
  >>
  \layout {
    ragged-right = ##f
    \context {
      \Voice
      \remove "Finger_glide_engraver"
```

```
    }
    \context {
      \Staff
      \consists "Finger_glide_engraver"
    }
  }
}
```



Voir aussi

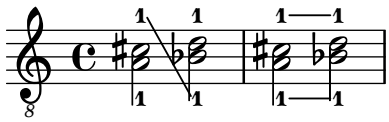
Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667.

Manuel des références internes : Section “FingeringGlideEvent” dans *Référence des propriétés internes*, Section “fingering-glide-event” dans *Référence des propriétés internes*, Section “Finger_glide_engraver” dans *Référence des propriétés internes*, Section “finger-glide-interface” dans *Référence des propriétés internes*, Section “FingerGlideSpanner” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un glissé mutiple sur un même doigt n’est pas autorisé et peut entraîner un résultat inattendu. Une méthode de contournement consiste à utiliser des doigts différents alliés à l’instruction `\tweak text`.
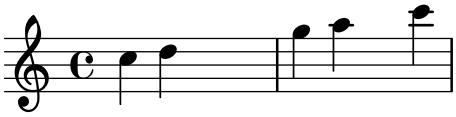
```
{
    \clef "G_8"
    <a\glide-1 cis'\glide-1>2 <bes-1 d'-1>
    <a\glide-1 cis'\glide-\tweak text "1"-2>2
    <bes-1 d'-\tweak text "1"-2>
}
```



Dictée à trous
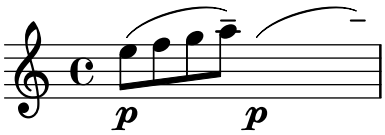
Les notes masquées – ou invisibles ou encore transparentes – sont utiles dans le cadre d’exercices de théorie ou de composition.

```
\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
  b
  \unHideNotes
  c
}
```



Têtes de note, hampes, crochets et silences sont invisibles. Une ligature sera invisible si elle démarre sur une note invisible. Les objets de notation attachés à une note invisible ne seront pas masqués pour autant.

```
\relative c'' {
  e8(\p f g a)--
  \hideNotes
  e8(\p f g a)--
}
```



Commandes prédéfinies

`\hideNotes`, `\unHideNotes`.

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : [Masquage de portées], page 212, [Silences invisibles], page 63, Section 5.4.7 [Visibilité des objets], page 675.

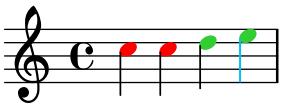
Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Note_spacing_engraver" dans *Référence des propriétés internes*, Section "NoteSpacing" dans *Référence des propriétés internes*.

Coloration d'objets
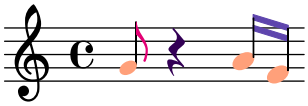
Des objets peuvent être colorisés individuellement. Une liste des noms des couleurs disponibles se trouve à l'annexe Section A.7 [Liste des couleurs], page 716.

```
\override NoteHead.color = #red
c''4 c''
\override NoteHead.color = #(x11-color 'LimeGreen)
d''
\override Stem.color = "deepskyblue"
e''
```



En plus du jeu limité de couleurs de base prédéfini sous forme de variable – voir les « couleurs normales » dans Section A.7 [Liste des couleurs], page 716, – n'importe quelle couleur peut être spécifiée sous forme de chaîne, qu'il s'agisse d'un nom de couleur prédéfini dans le style CSS (`https://www.w3.org/Style/CSS/`) ou d'un code hexadécimal préfixé d'un `#` (le tout entre guillemets informatiques) :

```
\override NoteHead.color = "lightsalmon"
\override Flag.color = "#E30074"
\override Beam.color = "#5e45ad"
\override Rest.color = "#3058"
g'8 \huge r4 a'16 f'
```
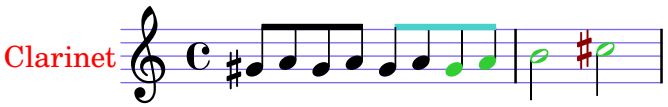


Lorsque ce code couleur comporte un canal alpha pour la transparence, spécifier un code sur huit caractères "*#RRGGBBAA*", ou sa forme abrégée "*#RGBA*", produira ses effets pour une sortie SVG, mais pas pour du PostScript ni du PDF. Dans l'exemple précédent, on peut voir à travers le silence dès lors qu'il est compilé avec le moteur SVG, comme expliqué dans Section 3.4.3 [Formats de sortie alternatifs], page 553.

De manière différente, l'intégralité de la palette des couleurs définies pour X11 (`https://en.wikipedia.org/wiki/X11_color_names`) est accessible par la fonction Scheme `x11-color`. Cette fonction prend en argument une expression symbolique de la forme `'DarkSeaGreen4` ou bien une chaîne de caractères comme `"DarkSeaGreen4"`. La première formulation est à la fois plus rapide à écrire et aussi plus efficace. Néanmoins, la deuxième forme permet d'accéder aux noms composés des couleurs de X11 comme ici `"dark sea green 4"`.

Lorsque la fonction `x11-color` ne trouve pas le paramètre fourni, elle revient à la couleur par défaut, le noir. Le problème ressort de façon évidente au vu de la partition finale.

```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
    }
  }
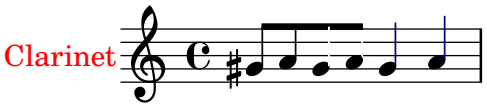
  \relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  gis8 a
  \override Beam.color = #(x11-color "medium turquoise")
  gis a
  \override Accidental.color = #(x11-color 'DarkRed)
  gis a
  \override NoteHead.color = #(x11-color "LimeGreen")
  gis a
  % this is deliberate nonsense; note that the stems remain black
  \override Stem.color = #(x11-color 'Boggle)
  b2 cis
}
```



Un autre moyen consiste à fournir à la fonction Scheme `rgb-color` les composantes de la couleur exacte au format rouge-vert-bleu (*RGB*) – chacune étant exprimée en en fraction de 256 (le 0.5 de l'exemple suivant correspond à 128). ainsi qu'éventuellement un nombre définissant le niveau de transparence. Une fois de plus, cette transparence n'est effective que dans le cadre du format SVG, où la clef de l'extrait ci-dessous sera affectée.

```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
    }
  \override Clef.color = #(rgb-color 0 0 0 0.5)
  }
\relative c'' {
```

```
  \override Stem.color = #(rgb-color 0 0 0)
  gis8 a
  \override Stem.color = #(rgb-color 1 1 1)
  gis8 a
  \override Stem.color = #(rgb-color 0 0 0.5)
  gis4 a
}
```



Voir aussi

Manuel de notation : Section 5.3.4 [La commande d'affinage `\tweak`], page 656, Section A.7 [Liste des couleurs], page 716.

Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Problèmes connus et avertissements

Une couleur x11 n'aura pas forcément le même rendu qu'une couleur normale ayant un nom similaire.
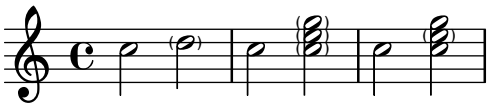
Les couleurs de X11 ne sont pas toutes perceptibles dans un navigateur internet. Aussi nous vous recommandons, dans le cadre d'une présentation multimédia, d'utiliser des couleurs CSS.

Vous ne pouvez pas coloriser individuellement des notes à l'intérieur d'un accord avec `\override`. Si besoin est, utilisez `\tweak` ou `\single\override` devant la note en question. Pour plus de détails, consultez Section 5.3.4 [La commande d'affinage `\tweak`], page 656.

Parenthèses
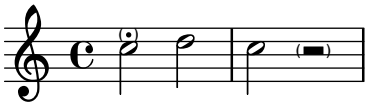
Des objets peuvent être mis entre parenthèses en saisissant `\parenthesize` juste avant l'événement musical. Si l'instruction préfixe un accord, chaque note le composant se présentera entre parenthèses. Vous pouvez aussi mettre individuellement entre parenthèses les notes d'un accord.

```
\relative {
  c''2 \parenthesize d
  c2 \parenthesize <c e g>
  c2 <c \parenthesize e g>
}
```



Les objets autres que des notes peuvent aussi être entre parenthèses. En ce qui concerne les articulations, l'instruction `\parenthesize` doit cependant être précédée d'un tiret.

```
\relative {
  c''2-\parenthesize -. d
  c2 \parenthesize r
}
```



Voir aussi

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Parenthesis_engraver” dans *Référence des propriétés internes*, Section “ParenthesesItem” dans *Référence des propriétés internes*, Section “parentheses-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Lorsqu'un accord est mis entre parenthèses, celles-ci s'appliquent individuellement à chacune des notes le composant, alors qu'on attendrait une seule paire de parenthèses encadrant tout l'accord.

Hampes

Dès qu'une note est rencontrée, un objet `Stem` se crée automatiquement. Pour les rondes et les silences, ils sont aussi créés, mais en mode invisible.

L'orientation des hampes peut être définie manuellement – voir Section 5.4.2 [Direction et positionnement], page 667, à ce sujet.

Commandes prédéfinies
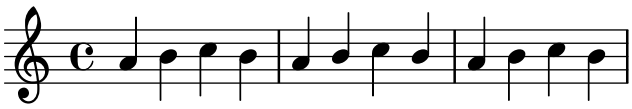
`\stemUp`, `\stemDown`, `\stemNeutral`.

Morceaux choisis

Direction par défaut des hampes de la ligne médiane

La direction des hampes des notes placées sur la ligne médiane de la portée est gérée par la propriété `neutral-direction` de l'objet `Stem`.

```
\relative c'' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}
```
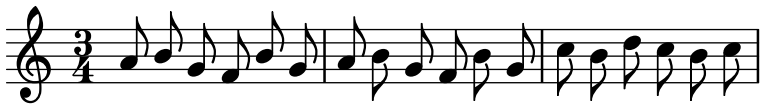


Changement automatique de l'orientation de hampe de la note médiane selon la mélodie

Afin de suivre la ligne mélodique, LilyPond peut inverser l'orientation de hampe de la note médiane, dès lors qu'aura été ajouté au contexte de voix le graveur `Melody_engraver`.

La propriété de contexte `suspendMelodyDecisions` permet, si besoin, de désactiver temporairement ce comportement.

```
\relative c'' {
  \time 3/4
  a8 b g f b g |
  \set suspendMelodyDecisions = ##t
  a  b g f b g |
  \unset suspendMelodyDecisions
  c  b d c b c |
}
```

```
\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
  }
}
```



Voir aussi

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667.

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Stem_engraver” dans *Référence des propriétés internes*, Section “Stem” dans *Référence des propriétés internes*, Section “stem-interface” dans *Référence des propriétés internes*.

1.7.2 Hors de la portée
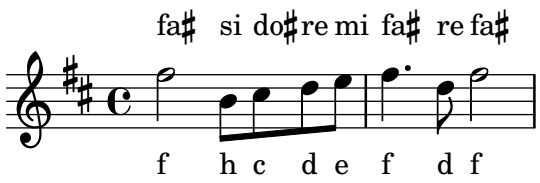
Nous allons nous intéresser ici à souligner des éléments inscrits dans la portée par des éléments qui lui seront externes.

Nom des notes

Le nom des notes peut s’imprimer sous forme de texte à l’aide du contexte `NoteNames`. Utilisé conjointement à une portée standard, chaque hauteur sera synchronisée avec son nom, qu’il soit imprimé au-dessus ou en dessous de la portée.

```
\language "italiano"
melody = \relative do'' {
  fad2 si,8 dod re mi fad4. re8 fad2
}

<<
  \new NoteNames { \melody }
  \new Staff { \key si \minor \melody }
  \new NoteNames {
    \set printNotesLanguage = "deutsch"
    \set printAccidentalNames = ##f
    \melody
  }
>>
```



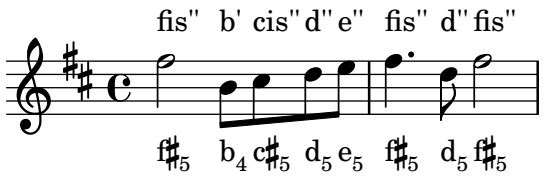
Le nom des notes est imprimé par défaut dans la langue utilisée pour la saisie ; la propriété `printNotesLanguage` permet cependant de sélectionner l’une des autres langues disponibles –

voir [Nom des notes dans d'autres langues], page 8. La présentation des altérations se gère à l'aide de la propriété `printAccidentalNames`.

L'utilisation conjointe de ces deux propriétés et l'activation de `printOctaveNames` donnera une représentation comparable à la syntaxe de saisie de LilyPond. Pour une représentation plus généraliste, on peut utiliser des numéros d'octave.

```
melody = \relative c'' {
  fis2 b,8 cis d e fis4. d8 fis2
}
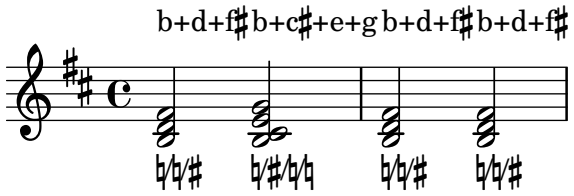
<<
  \new NoteNames {
    \set printOctaveNames = ##t
    \set printAccidentalNames = #'lily
    \melody
  }
  \new Staff { \key b \minor \melody }
  \new NoteNames {
    \set printOctaveNames = #'scientific
    \melody
  }
>>
```



La propriété `noteNameSeparator` définit comment les accords seront représentés. D'autres fonctions de formatage peuvent se définir en tant que `noteNameFunction`. Une telle fonction doit attendre une hauteur (`pitch`) et un contexte en tant qu'arguments même si l'un peut être ignoré.

```
somechords = \relative c' {
  <b d fis>2 <b cis e g> <b d fis> q
}

<<
  \new NoteNames {
    \set noteNameSeparator = "+"
    \somechords
  }
  \new Staff { \key b \minor \somechords }
  \new NoteNames {
    \set noteNameFunction =
      #(lambda (pitch ctx)
         (alteration->text-accidental-markup
          (ly:pitch-alteration pitch)))
    \somechords
  }
>>
```



Voir aussi

Manuel de notation : [Nom des notes dans d'autres langues], page 8.

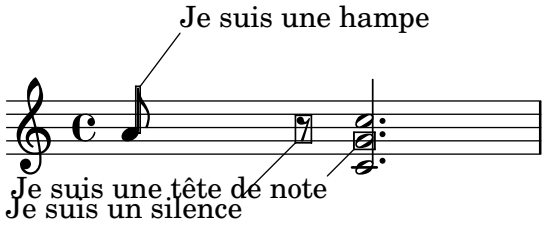
Référence des propriété internes : Section "NoteName" dans *Référence des propriétés internes*, Section "NoteNames" dans *Référence des propriétés internes*, Section "Note_name_engraver" dans *Référence des propriétés internes*.

Info-bulle

Vous pouvez marquer et nommer des éléments de notation à l'aide de bulles. L'objectif premier de cette fonctionnalité est d'expliquer la notation.

En voici un exemple :

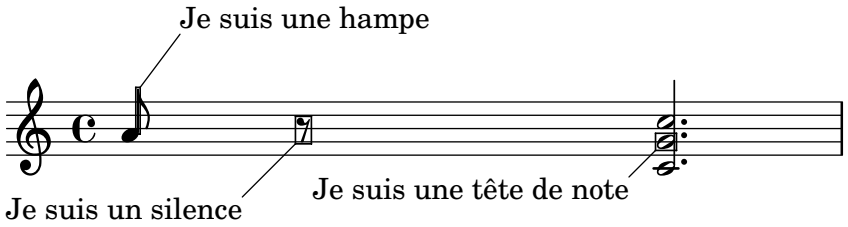
```
\new Voice \with { \consists "Balloon_engraver" }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Je suis un silence" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "Je suis une tête de note" } c>2.
}
```



Vous disposez de deux fonctions musicales, `balloonGrobText` et `balloonText`. `balloonGrobText` prend en argument l'objet graphique à agrémenter et s'utilise comme `\once \override`. `balloonText`, quant à lui, s'utilise comme une simple articulation et fonctionne comme `\tweak` pour attacher du texte à l'une des notes d'un accord. Les autres arguments sont le décalage et le texte de la bulle.

Les info-bulles n'ont aucune influence sur l'espacement des notes ; on peut toutefois le gérer ainsi :

```
\new Voice \with { \consists "Balloon_engraver" }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Je suis un silence" }
  r
  \balloonLengthOn
  <c, g'-\balloonText #'(-2 . -2) \markup { "Je suis une tête de note" } c>2.
}
```



Commandes prédéfinies

`\balloonLengthOn`, `\balloonLengthOff`.

Voir aussi

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Balloon_engraver” dans *Référence des propriétés internes*, Section “BalloonTextItem” dans *Référence des propriétés internes*, Section “balloon-interface” dans *Référence des propriétés internes*.

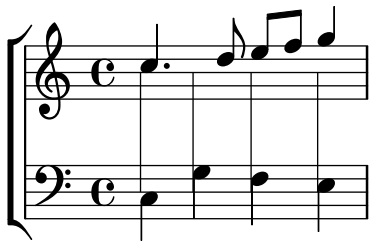
Quadrillage temporel

Vous pouvez tracer des lignes entre les portées, synchronisées avec les notes.

LilyPond a recours à deux graveurs distincts afin d’afficher le quadrillage : le `Grid_point_engraver` se charge de déterminer l’envergure du crochet, alors que le `Grid_line_span_engraver` se consacrera au tracé des lignes. Les lignes sont par défaut centrées horizontalement sous les notes et alignées sur la gauche des têtes. La propriété `gridInterval` spécifie quant à elle l’espace de temps entre chaque ligne.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver" %% active les guides
    gridInterval = #(ly:make-moment 1/4)
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
     %% centre les lignes guides horizontalement sous les notes
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative {
      \stemUp
      c''4. d8 e8 f g4
    }
    \new Staff \relative {
    %% centre les lignes guides verticalement
      \clef bass
      \stemDown
      c4  g'  f  e
    }
  >>
}
```

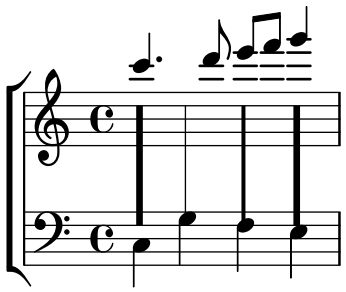


Morceaux choisis

Apparence du quadrillage temporel

Modifier certaines des propriétés du quadrillage temporel aura pour effet d'en changer l'apparence.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \relative c'' {
        \stemUp
        c'4. d8 e8 f g4
      }
    }
    \new Staff {
      \relative c {
        % this moves them up one staff space from the default position
        \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
        \stemDown
        \clef bass
        \once \override Score.GridLine.thickness = #5.0
        c4
        \once \override Score.GridLine.thickness = #1.0
        g'4
        \once \override Score.GridLine.thickness = #3.0
        f4
        \once \override Score.GridLine.thickness = #5.0
        e4
      }
    }
  >>
  \layout {
    \context {
      \Staff
      % set up grids
      \consists "Grid_point_engraver"
      % set the grid interval to one quarter note
      gridInterval = #(ly:make-moment 1/4)
    }
    \context {
      \Score
      \consists "Grid_line_span_engraver"
      % this moves them to the right half a staff space
      \override NoteColumn.X-offset = #-0.5
    }
  }
}
```



Voir aussi
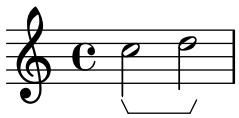
Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Grid_line_span_engraver” dans *Référence des propriétés internes*, Section “Grid_point_engraver” dans *Référence des propriétés internes*, Section “GridLine” dans *Référence des propriétés internes*, Section “GridPoint” dans *Référence des propriétés internes*, Section “grid-line-interface” dans *Référence des propriétés internes*, Section “grid-point-interface” dans *Référence des propriétés internes*.

Crochets d’analyse

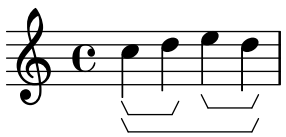
On utilise des crochets en analyse musicale, pour indiquer la structure d’une pièce.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}
```



Les crochets d’analyses sont susceptibles d’être imbriqués :

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```
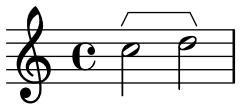


Morceaux choisis

Crochets d'analyse au-dessus de la portée

Les crochets d'analyse viennent par défaut se positionner au-dessous de la portée. L'exemple suivant vous indique comment les faire apparaître en surplomb de la portée.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}

\relative c'' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```

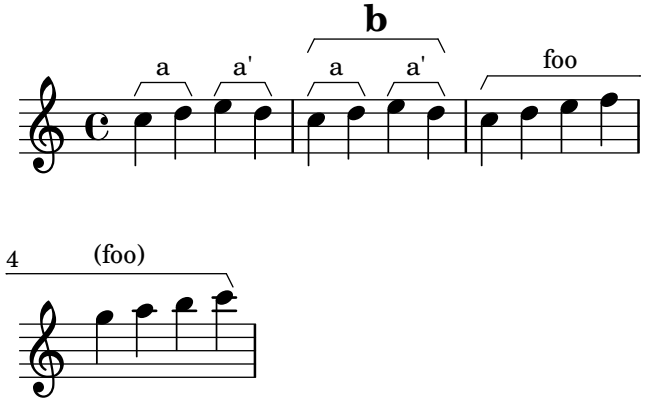


Crochet d'analyse avec texte

Du texte peut venir s'ajouter aux crochets d'analyse grâce à la propriété `texte` de l'objet graphique `HorizontalBracketText`. Plusieurs crochets présents en un même moment requièrent d'utiliser la commande `\tweak`. Le texte ajouté sera répété, entre parenthèse, après un saut de ligne.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}
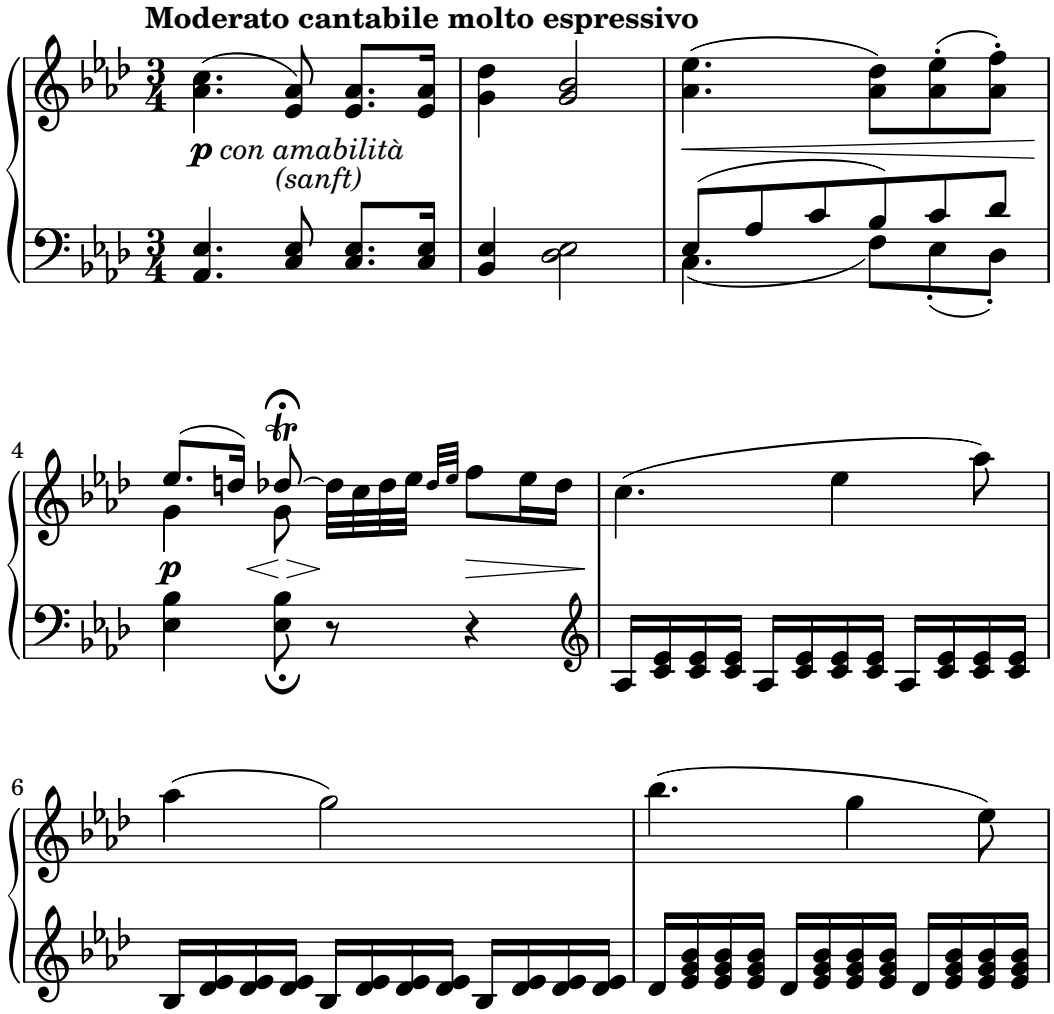
{
  \once\override HorizontalBracketText.text = "a"
    c''\startGroup d''\stopGroup
    \once\override HorizontalBracketText.text = "a'"
    e''\startGroup d''\stopGroup |
  c''-\tweak HorizontalBracketText.text
        \markup \bold \huge "b" \startGroup
     -\tweak HorizontalBracketText.text "a" \startGroup
    d''\stopGroup
    e''-\tweak HorizontalBracketText.text "a'" \startGroup
    d''\stopGroup\stopGroup |
  c''-\tweak HorizontalBracketText.text foo \startGroup
    d'' e'' f'' | \break
  g'' a'' b'' c'''\stopGroup
}
```



Voir aussi

Référence des propriétés internes : Section “Horizontal_bracket_engraver” dans *Référence des propriétés internes*, Section “HorizontalBracket” dans *Référence des propriétés internes*, Section “horizontal-bracket-interface” dans *Référence des propriétés internes*, Section “HorizontalBracketText” dans *Référence des propriétés internes*, Section “horizontal-bracket-text-interface” dans *Référence des propriétés internes*, Section “Staff” dans *Référence des propriétés internes*.

1.8 Texte



Nous allons voir ici comment insérer du texte dans une partition, ainsi que différentes manières de le mettre en forme.

1.8.1 Ajout de texte

Cette partie constitue une introduction aux différentes manières d'ajouter du texte à une partition.

> **Note :** Pour écrire des accents et autres caractères spéciaux, il suffit de les insérer directement dans votre fichier LilyPond. Ce fichier devra être sauvegardé avec l'encodage UTF-8. Pour plus d'informations, voir [Codage du texte], page 549.

Vue d'ensemble des objets textuels

Les objets textuels se saisissent soit comme de simples chaînes de caractères entre quillemets informatique, soit dans un bloc `\markup` qui acceptte une mise en forme fine du texte et des réhaussements graphiques comme indiqué dans Section 1.8.2 [Mise en forme du texte], page 263.

En tant que tels, les blocs *markup* peuvent s'utiliser :

- dans tout objet `TextScript` attaché à des notes par `-`, `^` ou `_` ; voir see [Commentaires textuels], page 256 ;
- en tant que bandeau (*spanner*) lorsqu'une indication se prolongge sur plusieurs temps ou mesures ; voir [Indication textuelle avec extension], page 257 ;
- dans toute marque affichée au-dessus de la partition, tels les objets `RehearsalMark` ou `MetronomeMark` introduits respectivement par les mots-clé `\mark` ou `\tempo` ; voir [Indications textuelles], page 259 ;
- en tant que bloc *markup* indépendant, saisi en dehors de tout bloc `\score` – dans ce cas précis la commande `\markup {...}` est obligatoire et ne saurait être remplacée par du simple texte borné par des guillemets informatiques ; voir [Texte indépendant], page 262 ;
- dans toute définition qu sein d'un bloc `\header` (par ex. `title`, `subtitle`, `composer`), ou dans des éléments spécifiques au bloc `\paper` tel que `evenHeaderMarkup` pour les numéros de page. Ceci est abordé dans Section 3.2 [Titres et entêtes], page 515.

De nombreux autres objets basés sur du texte peuvent se saisir sous forme de bloc *markup* même si ce n'esp pas leur usage premier.

- Les doigtés peuvent se remplacer par des blocs de *markup* dès lors qu'ils sont introduits par la commande `\finger` ; voir [Doigtés], page 236.
- Les syllabes des paroles peuvent être formatées avec une commande `\markup` ; voir Section 2.1.1 [Vue d'ensemble de la musique vocale], page 287.
- Les noms des accords sont en fait définis sous forme de blocs de *markup* et peuvent donc être redéfinis de la même manière pour adapter les modificateurs d'accord ou les exceptions ; voir Section 2.7.2 [Gravure des accords], page 450.
- Les nuances se saisissent de la manière la plus simple. Il est néanmoins possible de définir sa Section "propre indication" dans *Manuel de notation* sous forme d'objet *markup*. Certaines nuances, comme un crescendo, sont affichés sous forme de bandeau et peuvent être redéfinies au travers de certaines propriétés telle que `crescendoText` ; voir [Nuances], page 129.
- Des objets moins courants sont constitués de blocs *markup*, tels les indications en [Infobulle], page 248.

Il est en fait possible d'utiliser `\markup` pour personnaliser l'apparence de pratiquement n'importe quel objet graphique (*grob*) en appliquant une dérogation soit à sa propriété `text` s'il en dispose ou de sa propriété `stencil`. Une partie de la logique qui rend ceci possible est expliquée dans Section "Flexibilité architecturale" dans *Essai*.

L'exemple qui suit illustre l'ubiquité des blocs *markup* non seulement comme les objets ci-dessus présentés, mais aussi en remplaçant des objets musicaux par des objets textuels de différentes manières.

```
\header { title = \markup "Header" }

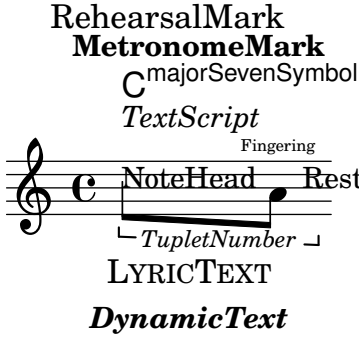
dyn =
#(make-dynamic-script #{ \markup \text "DynamicText" #})

\markup \box "Top-level markup"

\score {
  <<
    \new ChordNames
    \with {
      majorSevenSymbol = \markup "majorSevenSymbol"
    }
    \chordmode { c1:maj7 }
    \new Staff {
      \tempo \markup "MetronomeMark"
      \mark \markup \smaller "RehearsalMark"
      \once \override TupletNumber.text =
        \markup "TupletNumber"
      \tuplet 3/2 {
        \once \override NoteHead.stencil =
          #ly:text-interface::print
        \once \override NoteHead.text =
          \markup \lower #0.5 "NoteHead"
        c''8^\markup \italic "TextScript"
        a'\finger \markup \text "Fingering"
        \once \override Rest.stencil =
          #(lambda (grob)
             (grob-interpret-markup grob #{
               \markup  "Rest"
               #}))
        r
      }
    }
    \new Lyrics \lyricmode {
      \markup \smallCaps "LyricText" 1
    }
    \new Dynamics { s1\dyn }
  >>
}
```

Header

Top-level markup



Voir aussi

Manuel de notation : [Commentaires textuels], page 256, [Doigtés], page 236, Section 2.7.2 [Gravure des accords], page 450, [Indication textuelle avec extension], page 257, [Indications textuelles], page 259, [Info-bulle], page 248, Section 1.8.2 [Mise en forme du texte], page 263, [Nuances], page 129, [Personnalisation des indications de nuance], page 136, [Texte indépendant], page 262, Section 2.1.1 [Vue d'ensemble de la musique vocale], page 287.
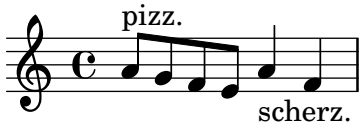
Essai sur la gravure musicale automatisée : Section "Flexibilité architecturale" dans *Essai*.

Morceaux choisis : Section "Text" dans *Morceaux choisis*.

Commentaires textuels
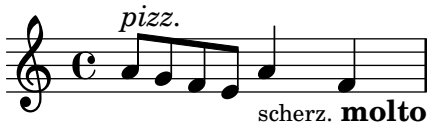
Vous pouvez ajouter à une partition des indications sous forme textuelle, comme dans l'exemple suivant. Ces indications se placeront manuellement au-dessus ou au-dessous de la portée selon la syntaxe utilisée – cf. Section 5.4.2 [Direction et positionnement], page 667.

```
\relative { a'8^"pizz." g f e a4-"scherz." f }
```
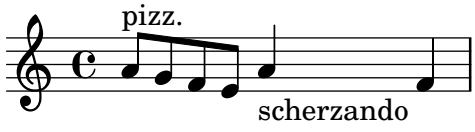


Cette syntaxe est en fait un raccourci. Des constructions plus élaborées d'annotation peuvent être obtenues en ayant recours à un bloc `\markup` et selon les préceptes énoncés dans Section 1.8.2 [Mise en forme du texte], page 263.

```
\relative {
  a'8^\markup { \italic pizz. } g f e
  a4_\markup { \tiny scherz. \bold molto } f }
```



Par défaut, ces indications n'affectent en rien l'espacement des notes. Leur longueur peut néanmoins être prise en considération : dans l'exemple qui suit, le premier commentaire n'influe pas sur l'espacement, à l'inverse du second.

```
\relative {
  a'8^"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}
```



En plus d'indications textuelles, les notes peuvent se voir attacher des articulations, comme indiqué au chapitre [Articulations et ornements], page 127.

Pour de plus amples détails sur la manière de combiner indications textuelles et articulations, reportez-vous au chapitre Section "Positionnement des objets" dans *Manuel d'initiation*.

Commandes prédéfinies

`\textLengthOn`, `\textLengthOff`.

Voir aussi

Manuel d'initiation : Section "Positionnement des objets" dans *Manuel d'initiation*.

Manuel de notation : [Articulations et ornements], page 127, Section 5.4.2 [Direction et positionnement], page 667, Section 1.8.2 [Mise en forme du texte], page 263.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

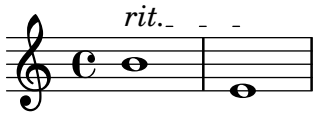
S'assurer que tous les éléments textuels et les paroles respectent les marges du document requiert des calculs relativement lourds ; vous pouvez toutefois vous en affranchir en ajoutant

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Indication textuelle avec extension
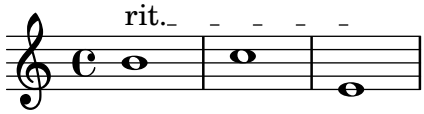
Certaines indications d'interprétation comme *rallentando*, *accelerando* ou *trilles*, s'inscrivent textuellement et se prolongent sur plusieurs notes à l'aide d'une ligne pleine, pointillée ou ondulée. Ces objets, que l'on appelle « extenseurs », se dessinent entre deux notes à l'aide de la syntaxe suivante :

```
\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
```



Le texte à imprimer est spécifié en tant que propriété de l'objet `TextSpanner`. Il apparaîtra par défaut en italique ; cependant, rien ne s'oppose à un autre graphisme dès lors que vous faites appel à un bloc `\markup` – voir Section 1.8.2 [Mise en forme du texte], page 263.

```
\relative {
  \override TextSpanner.bound-details.left.text =
    \markup { \upright "rit." }
  b'1\startTextSpan c
  e,\stopTextSpan
}
```



Le style de ligne se définit lui aussi comme une propriété de l'objet. Les détails concernant la syntaxe à utiliser sont expliqués au chapitre Section 5.4.8 [Styles de ligne], page 682.

Commandes prédéfinies

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.
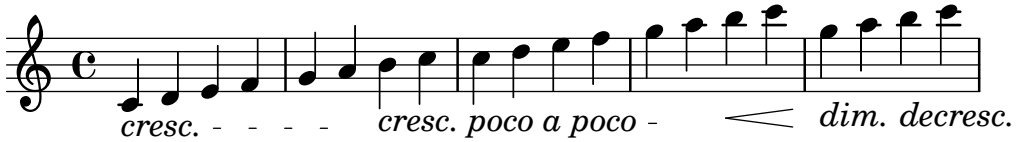
Morceaux choisis

Extensions de nuance postfix

Les lignes d'extension des commandes `\cresc`, `\dim` et `\decresc` peuvent désormais être personnalisées facilement sous forme d'opérateurs postfix. Soufflets et (de)crescendos peuvent cohabiter. `\<` et `\>` produiront par défaut des soufflets, alors que `\cresc`, etc. produiront une indication textuelle avec extension.

```
% Some sample text dynamic spanners, to be used as postfix operators
crpoco =
#(make-music 'CrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "cresc. poco a poco")

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decresc c4\!
}
```



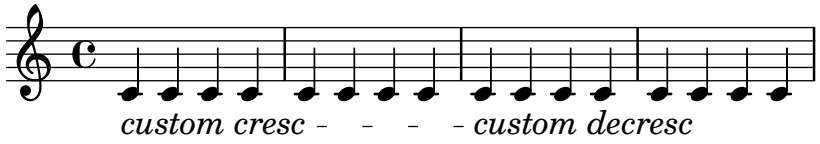
Personnalisation des extenseurs de nuance postfix

Il s'agit de fonctions postfix pour personnaliser l'extension des crescendos textuels. L'extension devrait débuter sur la première note de la mesure. Il faut utiliser `-\mycresc` – comme une articulation – sous peine que le départ de l'extension n'apparaisse qu'à la note suivante.

```
% Two functions for (de)crescendo spanners where you can explicitly
% give the spanner text.
mycresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'DecrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
```

```
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4\! c4 c4
}
```



Voir aussi

Manuel de notation : Section 1.8.2 [Mise en forme du texte], page 263, [Nuances], page 129, Section 5.4.8 [Styles de ligne], page 682.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextSpanner” dans *Référence des propriétés internes*.

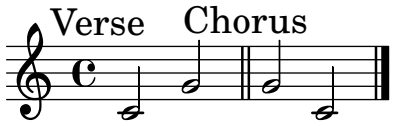
Problèmes connus et avertissements

LilyPond ne peut traiter qu’un seul extenseur à la fois par voix.

Indications textuelles
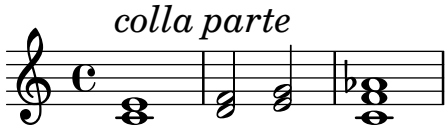
La commande `\mark` est tout d’abord conçue pour les [Indications de repère], page 117.

```
\relative {
  \mark "Verse"
  c'2 g'
  \bar "||"
  \mark "Chorus"
  g2 c,
  \bar "|."
}
```
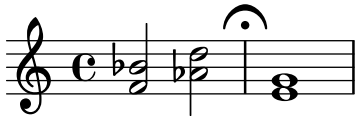


Cette syntaxe rend possible l’adjonction de n’importe quel texte à une barre de mesure. Ce texte peut être mis en forme de différentes manières dès lors qu’est utilisé un bloc `\markup`, comme indiqué au chapitre Section 1.8.2 [Mise en forme du texte], page 263.

```
\relative {
  <c' e>1
  \mark \markup { \italic { colla parte } }
  <d f>2 <e g>
  <c f aes>1
}
```
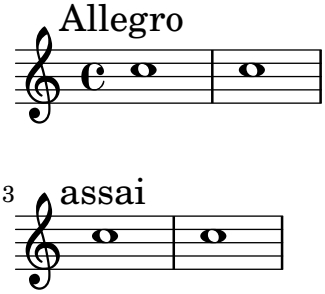


Elle peut aussi servir à insérer des signes de *coda* ou de *segno*, ou bien un point d'orgue, au-dessus d'une barre de mesure. Couplez-la alors à la commande `\markup` pour avoir accès au symbole approprié, selon les indications contenues au chapitre [Notation musicale dans du texte formaté], page 275.

```
\relative {
  <bes' f>2 <aes d>
  \mark \markup { \musicglyph "scripts.ufermata" }
  <e g>1
}
```



Le résultat de `\mark` n'apparaîtra que sur la portée supérieure d'un système. Si vous introduisez la commande `\mark` au moment d'une barre de mesure, la marque se placera au-dessus de la barre. Si vous y faites appel au milieu d'une mesure, cette marque sera positionnée entre les notes. Si elle intervient en début de ligne, elle sera placée juste avant la première note de cette portée. Enfin, une marque qui tomberait sur un saut de ligne sera imprimée au début de la ligne suivante.

```
\relative c'' {
  \mark "Allegro"
  c1 c
  \mark "assai" \break
  c  c
}
```



Commandes prédéfinies

`\markLengthOn`, `\markLengthOff`.
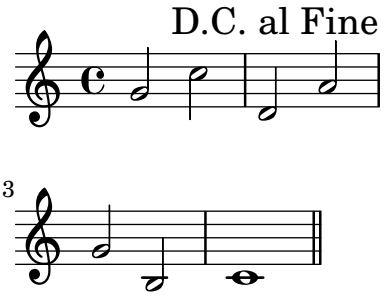
Morceaux choisis

Indication d'un repère en fin de ligne

Les indications de repère peuvent s'imprimer à la fin d'une ligne plutôt qu'en tête de la suivante. L'alignement sur la barre de mesure devra alors s'effectuer par l'extrémité droite de l'indication.

```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility =
    #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X =
```

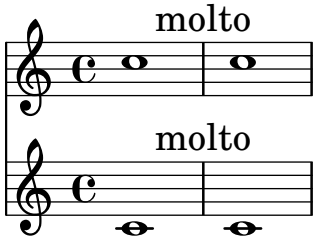
```
    #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}
```



Impression des indications sur toutes les portées d'un système

Bien que ces indications textuelles ne soient habituellement imprimées qu'au niveau de la portée supérieure d'un système, leur affectation peut être répercutée à chacune des portées.

```
\score {
  <<
    \new Staff { c''1 \mark "molto" c'' }
    \new Staff { c'1 \mark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove "Mark_engraver"
      \remove "Staff_collecting_engraver"
    }
    \context {
      \Staff
      \consists "Mark_engraver"
      \consists "Staff_collecting_engraver"
    }
  }
}
```



Voir aussi

Manuel de notation : [Indications de repère], page 117, Section A.8 [La fonte Emmentaler], page 718, Section 1.8.2 [Mise en forme du texte], page 263, [Notation musicale dans du texte formaté], page 275.

Morceaux choisis: Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section “MarkEvent” dans *Référence des propriétés internes*, Section “Mark_engraver” dans *Référence des propriétés internes*, Section “RehearsalMark” dans *Référence des propriétés internes*.

Texte indépendant

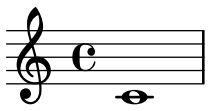
Un bloc `\markup` peut exister en lui-même, indépendamment de tout bloc `\score`, et venir en préambule par exemple – voir le chapitre Section 3.1.5 [Structure de fichier], page 513, à ce propos.

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

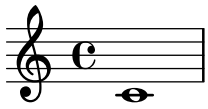
Tomorrow, and tomorrow, and tomorrow...

De cette manière, vous pouvez insérer du texte en dehors de la musique. Ceci devient particulièrement utile lorsque le fichier source contient plusieurs morceaux. Pour plus d'informations à ce propos, reportez-vous au chapitre Section 3.1.2 [Plusieurs partitions dans un même ouvrage], page 510.

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Les blocs de textes peuvent s'étendre sur plusieurs pages, ce qui permet de générer des ouvrages complets uniquement grâce à LilyPond. Cette fonctionnalité, ainsi que la syntaxe appropriée, est abordée plus en détail au chapitre [Texte avec sauts de page], page 278.

Commandes prédéfinies

`\markup`, `\markuplist`.

Morceaux choisis

Bloc de texte indépendant sur deux colonnes

L'utilisation de la commande `\markup` permet de distribuer un bloc de texte indépendant sur plusieurs colonnes.

```
\markup {
```

```
\fill-line {
  \hspace #1
  \column {
    \line { O sacrum convivium }
    \line { in quo Christus sumitur, }
    \line { recolitur memoria passionis ejus, }
    \line { mens impletur gratia, }
    \line { futurae gloriae nobis pignus datur. }
    \line { Amen. }
  }
  \hspace #2
  \column \italic {
    \line { O sacred feast }
    \line { in which Christ is received, }
    \line { the memory of His Passion is renewed, }
    \line { the mind is filled with grace, }
    \line { and a pledge of future glory is given to us. }
    \line { Amen. }
  }
  \hspace #1
 }
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.

Voir aussi

Manuel de notation : Section 1.8.2 [Mise en forme du texte], page 263, Section 3.1.2 [Plusieurs partitions dans un même ouvrage], page 510, Section 3.1.5 [Structure de fichier], page 513, [Texte avec sauts de page], page 278.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

1.8.2 Mise en forme du texte

Nous allons voir dans les lignes qui suivent la manière de mettre en forme du texte à l’aide de la syntaxe propre au mode `\markup`.

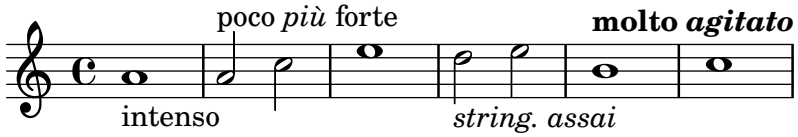
Introduction au formatage de texte

La commande `\markup` permet d’ajouter du texte et dispose de sa propre syntaxe que nous appellerons le « mode *markup* ». De tels blocs peuvent s’utiliser partout, comme indiqué dans [Vue d’ensemble des objets textuels], page 254.

La syntaxe du mode *markup* n’est pas différente de celle des autres modes de LilyPond : une expression `\markup` est bornée par des accolades `{ ... }`. Un mot unique sera considéré comme une expression minimale, et n’aura donc pas besoin d’être mis entre accolades.

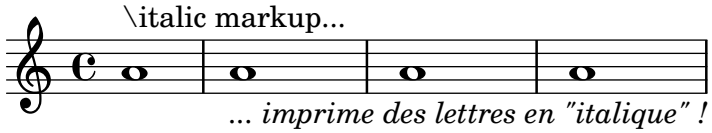
Contrairement aux indications simples, du type "entre guillemets", les blocs `\markup` peuvent contenir des expressions imbriquées ou d'autres commandes *markup*, dès lors qu'elles sont précédées du caractère \. Ces commandes n'affecteront que la première expression qui les suit.

```
\relative {
  a'1-\markup intenso
  a2^\markup { poco \italic più forte  }
  c e1
  d2_\markup { \italic "string. assai" }
  e
  b1^\markup { \bold { molto \italic  agitato } }
  c
}
```



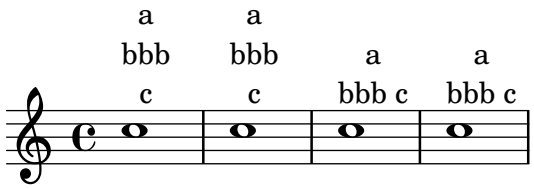
Un bloc `\markup` peut contenir du texte entre guillemets. De telles chaînes seront considérées comme des expressions textuelles minimales ; à ce titre, toute commande de type *markup* ou tout caractère spécial – tel un \ ou un # – sera imprimé littéralement et sans influer sur le formatage du texte. Il est de ce fait possible d'imprimer des guillemets informatiques " dès lors qu'ils sont précédés d'une oblique inverse.

```
\relative {
  a'1^"\italic markup..."
  a_\markup { \italic "... imprime des lettres en \"italique\" !" }
  a a
}
```



Une liste de mots, pour pouvoir être traitée en tant qu'expression distincte, doit être bornée par des " ou précédée d'une commande. La manière de définir les expressions *markup* aura une influence sur la manière dont elles seront empilées, centrées ou alignées. Dans l'exemple qui suit, la deuxième expression `\markup` est traitée tout comme la première :
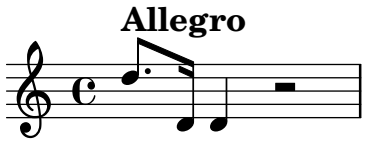
```
\relative c'' {
  c1^\markup { \center-column { a bbb c } }
  c1^\markup { \center-column { a { bbb c } } }
  c1^\markup { \center-column { a \line { bbb c } } }
  c1^\markup { \center-column { a "bbb c" } }
}
```



Vous pouvez stocker les étiquettes textuelles en tant que variables, et attacher ces identificateurs à des notes, comme ici :

```
allegro = \markup { \bold \large Allegro }

{
  d''8.^\allegro
  d'16 d'4 r2
}
```



Pour une liste des différentes commandes spécifiques au mode `\markup`, consultez l'annexe Section A.11 [Commandes pour *markup*], page 746, (en anglais).
Le fonctionnement interne de ces commandes et la façon d'en implémenter de nouveaux est abordée dans Section "Fonctions pour markups" dans *Extension de LilyPond*.

Voir aussi

Manuel de notation : Section A.11 [Commandes pour *markup*], page 746, [Vue d'ensemble des objets textuels], page 254.

Manuel d'extension des fonctionnalités : Section "Fonctions pour markups" dans *Extension de LilyPond*

Fichiers d'initialisation : `scm/markup.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.
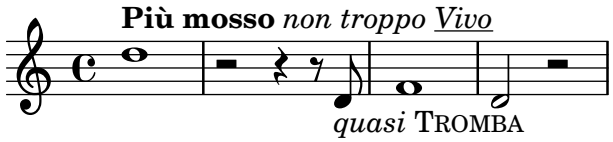
Problèmes connus et avertissements

Les messages d'erreur de syntaxe en mode *markup* sont peu explicites.

Sélection de la fonte et de la taille

Le mode *markup* autorise des changements élémentaires de la fonte :

```
\relative {
  d''1^\markup {
    \bold { Più mosso }
    \italic { non troppo \underline Vivo }
  }
  r2 r4 r8
  d,_\markup { \italic quasi \smallCaps Tromba }
  f1 d2 r
}
```

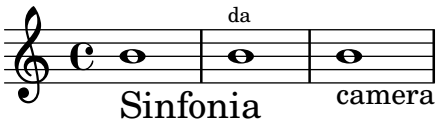


La taille des caractères se modifie, relativement à la taille globale des portées, de différentes manières.

Vous pouvez adopter l'une des tailles prédéfinies, comme ici :

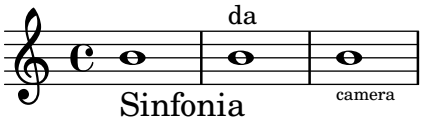
```
\relative b' {
```

```
  b1_\markup { \huge Sinfonia }
  b1^\markup { \teeny da }
  b1-\markup { \normalsize camera }
}
```



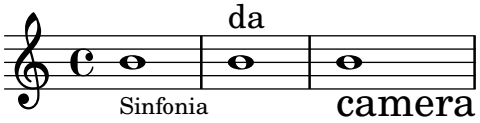
Vous pouvez la modifier relativement à sa valeur précédente :

```
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6  camera }
}
```



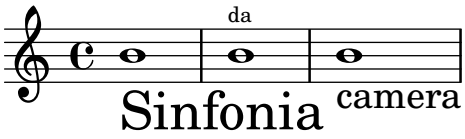
Vous pouvez l'augmenter ou la diminuer par rapport à la taille globale de portée :

```
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}
```



Vous pouvez lui attribuer une valeur arbitraire quelle que soit la taille de portée globale :

```
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}
```



Lorsque le texte comporte des espaces, mieux vaut le borner par des guillemets informatiques ; s'en suivra une meilleure adéquation entre la taille des espaces et celle des autres caractères :

```
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }
```

Sinfonia da camera

Sinfonia da camera

Vous pouvez imprimer du texte en indice ou en exposant. Celui-ci sera dans une taille plus petite, mais rien ne s'oppose à ce que vous lui affectiez un taille normale :

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement
      \sub { (part two) }  }
  }
}
```

1 st movement
1 st movement (part two)

Le mode *markup* vous permet de sélectionner d'autres familles de fontes. Par défaut, LilyPond utilise une police avec empattement, du type roman, et tout changement doit être explicite. Dans la dernière ligne de l'exemple qui suit, vous noterez qu'il n'y a aucune différence entre les premier et deuxième mots.

```
\markup {
  \column {
    \line { Act \number 1 }
    \line { \sans { Scene I. } }
    \line { \typewriter { Verona. An open place. } }
    \line { Enter \roman Valentine and Proteus. }
  }
}
```

Act 1
Scene I.
Verona. An open place.
Enter Valentine and Proteus.

Certaines familles de police spécifiques aux nombres ou aux nuances par exemple, ne disposent pas de tous les caractères, comme nous l'avons vu dans les chapitres [Personnalisation des indications de nuance], page 136, et [Indications de reprise manuelles], page 166.

Lorsqu'un changement survient au milieu d'un mot, il se peut qu'un espacement supplémentaire apparaisse. Il suffit en pareil cas de concaténer les différents éléments :

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

1st movement
***p**, con dolce espressione*

Une liste des différentes commandes permettant de changer de fonte ou d'utiliser des fontes personnalisées est disponible à l'annexe Section A.11.1 [Font], page 746.

Pour savoir comment personnaliser des fontes, reportez-vous au chapitre Section 1.8.3 [Fontes], page 279.

Commandes prédéfinies

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

Voir aussi

Manuel de notation : Section "Fonte" dans *Manuel de notation*, Section 1.8.3 [Fontes], page 279, [Indications de reprise manuelles], page 166, [Personnalisation des indications de nuance], page 136.

Fichiers d'initialisation : `scm/define-markup-commands.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Problèmes connus et avertissements
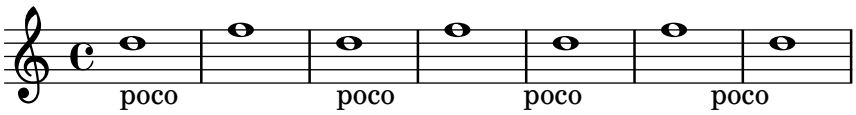
Le recours aux commandes `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` et `\huge` produiront des espacements nettement moins réguliers que si vous utilisez `\fontsize`.

Alignement du texte

Cette partie traite de la manière de positionner du texte en mode *markup*. On déplace l'intégralité d'un objet *markup* en utilisant la syntaxe décrite au chapitre Section "Déplacement d'objets" dans *Manuel d'initiation*.

Les objets de type *markup* peuvent s'aligner de différentes manières. Une indication textuelle est par défaut alignée sur son extrémité gauche. Dans l'exemple qui suit, il n'y a aucune différence entre les deux premiers *markups*. Cet exemple indique aussi différentes syntaxes permettant pour les commandes d'alignement.

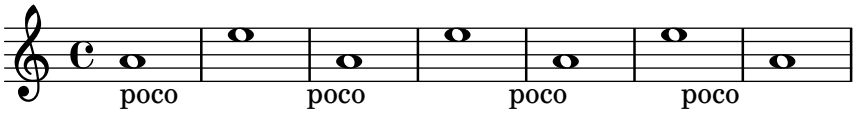
```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \left-align poco }
  f
  d-\markup { \center-align { poco } }
  f
  d-\markup \right-align { poco }
}
```



L'alignement horizontal peut être ajusté à l'aide d'une valeur numérique :

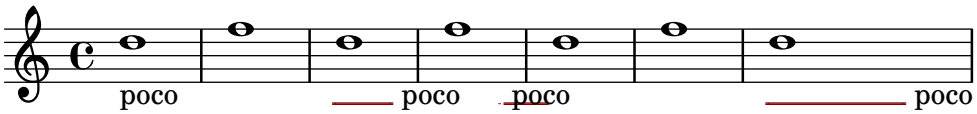
```
\relative {
  a'1-\markup { \halign #-1 poco }
  e'
  a,-\markup { \halign #0 poco }
```

```
  e'
  a,-\markup { \halign #0.5 poco }
  e'
  a,-\markup { \halign #2 poco }
}
```



Enfin, les mots et tout autre objet peuvent se déplacer en les faisant précéder d'un décalage. Un décalage négatif est possible, ce qui aura pour effet de déplacer les objets qui le suivent dans la direction opposée. Bien que le décalage soit normalement invisible, des commandes spécifiques permettent de le mettre en évidence dans l'exemple ci-dessous :

```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \with-color #darkred \box \hspace #4 poco }
  f
  d-\markup { \with-color #darkred \box \hspace #-4 poco }
  f
  d-\markup { \with-color #darkred \box \hspace #10 poco }
}
```

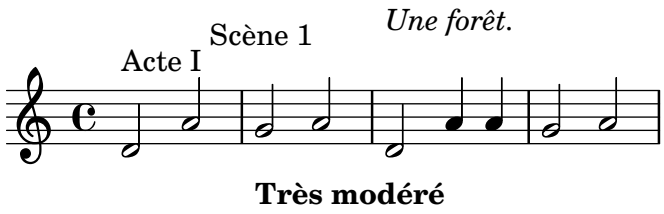


Certains objets possèdent leurs propres procédures d'alignement, qui annuleront toute spécification d'alignement que vous pourriez leur affecter. La solution consiste alors à déplacer l'intégralité de ces objets *markup*, comme indiqué par exemple au chapitre [Indications textuelles], page 259.

L'alignement vertical est quant à lui un peu plus compliqué. Comme nous l'avons vu ci-avant, les objets *markup* peuvent être déplacés dans leur intégralité. Il est néanmoins possible de déplacer certains éléments spécifiques au sein d'un bloc *markup*. En pareil cas, l'élément à déplacer doit être précédé d'un « point d'ancrage » – un autre élément du *markup* ou un objet invisible. L'exemple qui suit illustre ces deux possibilités. Vous noterez par ailleurs que le dernier *markup*, dépourvu de point d'ancrage, n'est de ce fait pas déplacé.

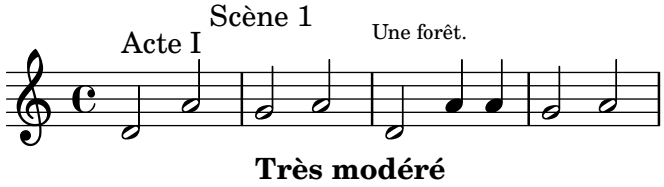
```
\relative {
  d'2^\markup {
    Acte I
    \raise #2 { Scène 1 }
  }
  a'
  g_\markup {
    \lower #4 \bold { Très modéré }
  }
  a
  d,^\markup \raise #4 \italic {
    Une forêt.
    }
```

```
  a'4 a g2 a
}
```



Certaines commandes permettent de régler l'alignement des objets textuels en mode *markup*, tant au niveau horizontal que vertical.

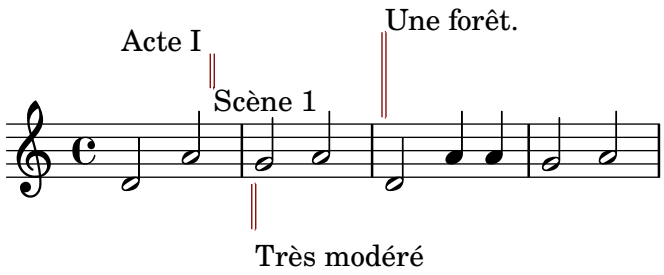
```
\relative {
  d'2^\markup {
    Acte I
    \translate #'(-1 . 2) "Scène 1"
  }
  a'
  g_\markup {
    \general-align #Y #3.2 \bold "Très modéré"
  }
  a
  d,^\markup \translate-scaled #'(-1 . 2) \teeny {
   "Une forêt."
  }
  a'4 a g2 a
}
```



Ici aussi, un décalage (positif ou négatif) constitue un moyen efficace de positionner verticalement des objets empilés dans un *markup* :

```
\relative {
  d'2^\markup {
    Acte I
    \column {
      \with-color #darkred \box \vspace #-1
      "Scène 1"
    }
  }
  a'
  g_\markup \column {
    \with-color #darkred \box \vspace #1
    "Très modéré"
  }
  a
  d,^\markup \column {
   "Une forêt."
```

```
    \with-color #darkred \box \vspace #2
  }
  a'4 a g2 a
}
```



Un objet de type *markup* peut contenir plusieurs lignes de texte. Dans l'exemple suivant, chaque élément ou expression ira se placer sur sa propre ligne, tantôt alignée à gauche, tantôt centrée.

```
\markup {
  \column {
    a
    "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a
    "b c"
    \line { d e f }
  }
}
```

a
b c
d e f

a
b c
d e f

Pareillement, une liste d'éléments ou d'expressions sera répartie sur une ligne entière, voire même centrée sur toute la page s'il n'y a qu'un seul élément. De telles expressions peuvent à leur tour contenir du texte multiligne ou une autre expression *markup*.

```
\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
\markup {
  \fill-line { 1885 }
}
```

William S. Gilbert THE MIKADO Sir Arthur Sullivan
or
THE TOWN OF TITIPU

1885

Les éléments peuvent se répartir sur une largeur spécifiée par une dérogation à la propriété `line-width`. Celle-ci est réglé par défaut à `#f`, autrement dit l'entièreté de la ligne :

```
\markup {
  \column {
    \fill-line { left center right }
    \null
    \override #'(line-width . 30)
    \fill-line { left center right }
  }
}
```

left center right

left center right

Les indications textuelles, lorsqu'elles sont relativement longues, peuvent se répartir sur plusieurs lignes en fonction de la largeur de ligne. Le texte sera alors soit aligné à gauche, soit justifié, comme le montre l'exemple suivant :

```
\markup {
  \column {
    \line  \smallCaps { La vida breve }
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de
      gitanos en el Albaicín de Granada. Al fondo una
      puerta por la que se ve el negro interior de
      una Fragua, iluminado por los rojos resplandores
      del fuego.)
    }
    \hspace #0

    \line \bold { Acto II }
    \override #'(line-width . 50)
    \justify \italic {
      (Calle de Granada. Fachada de la casa de Carmela
      y su hermano Manuel con grandes ventanas abiertas
      a través de las que se ve el patio
      donde se celebra una alegre fiesta)
    }
  }
}
```

La vida breve

Acto I

(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)

Acto II

(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre fiesta)

Une liste des différentes commandes permettant d'aligner du texte en mode *markup* est disponible à l'annexe Section A.11.2 [Align], page 756.

Voir aussi

Manuel d'initiation : Section "Déplacement d'objets" dans *Manuel d'initiation*.

Manuel de notation : Section "Alignement" dans *Manuel de notation*, [Indications textuelles], page 259.

Fichiers d'initialisation : `scm/define-markup-commands.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

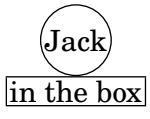
Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Éléments graphiques dans du texte formaté

Vous pouvez, grâce au mode *markup*, ajouter divers objets graphiques à votre partition.

Certaines commandes de *markup* permettent d'ornementer des éléments textuels avec des graphismes, à l'instar de l'exemple suivant :

```
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
    }
    \null
    \rounded-box \bold Prelude
  }
}
```

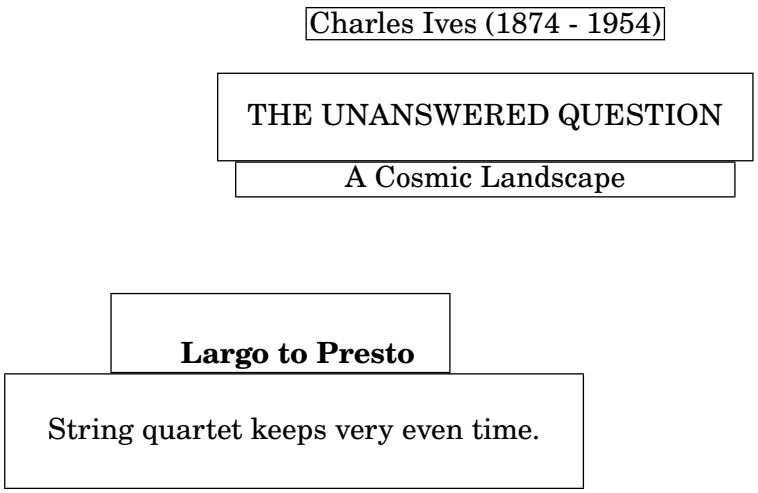


Erik Satie [1866 - 1925]

Prelude

Certaines directives peuvent nécessiter d'accroître l'espacement autour du texte – voir l'annexe Section A.11.2 [Align], page 756, pour une liste des différentes commandes particulières au mode *markup* ainsi que leur description.

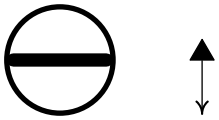
```
\markup \fill-line {
  \center-column {
    \box "Charles Ives (1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
      \bold "Largo to Presto"
  }
  \box \pad-around #3 "String quartet keeps very even time."
}
```



Vous pouvez imprimer certains graphismes ou symboles sans qu'il n'y ait de texte. Ces objets peuvent même se combiner, à l'instar de n'importe quelle expression *markup*.

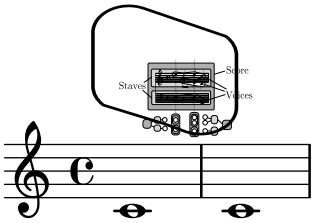
```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Des fonctionnalités graphiques avancées vous permettent même d'inclure dans une partition un fichier image converti au format PostScript encapsulé (extension `eps`), ou bien de tracer un graphique directement dans le fichier source à partir d'instructions PostScript pures. Nous vous conseillons, en pareil cas, de toujours spécifier les dimensions du dessin, comme dans ce qui suit :

```
c'1^\markup {
  \combine
    \epsfile #X #10 "./context-example.eps"
    \with-dimensions #'(0 . 6) #'(0 . 10)
    \postscript "
      -2 3 translate
      2.7 2 scale
      newpath
      2 -1 moveto
      4 -2 4 1 1 arct
      4 2 3 3 1 arct
      0 4 0 3 1 arct
      0 0 1 -1 1 arct
      closepath
      stroke"
  }
c'
```



L'annexe Section A.11.3 [Graphic], page 771, répertorie les différentes commandes en matière de graphisme.

Voir aussi

Manuel de notation : Section "Alignement" dans *Manuel de notation*, Section 1.7 [Annotations éditoriales], page 232, Section 5.4.4 [Dimensions], page 669, Section "Graphique" dans *Manuel de notation*.

Fichiers d'initialisation : `scm/define-markup-commands.scm`, `scm/stencil.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

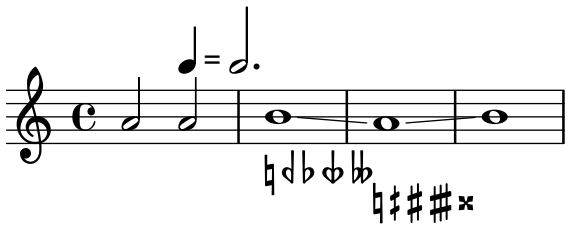
Notation musicale dans du texte formaté

Divers éléments de notation peuvent orner une partition, au moyen d'un objet *markup*.

Notes et altérations sont données à l'aide d'instructions *markup* :
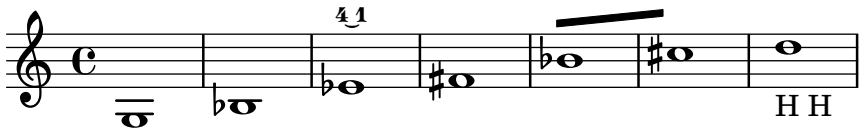
```
a'2 a'^\markup {
  \note {4} #1
  =
  \note-by-number #1 #1 #1.5
```

```
}
b'1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a'1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b'
```



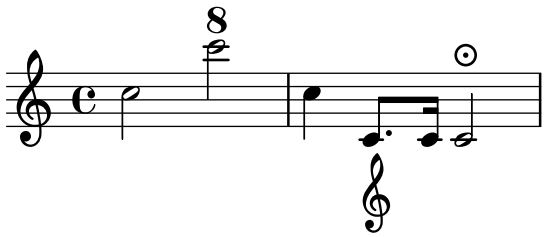
Le mode *markup* permet d'accéder à d'autres objets de notation :

```
\relative {
  g1 bes
  ees\finger \markup \tied-lyric "4~1"
  fis_\markup { \dynamic rf }
  bes^\markup {
    \beam #8 #0.1 #0.5
  }
  cis
  d-\markup {
    \markalphabet #8
    \markletter #8
  }
}
```



En règle générale, tout symbole musical peut être inclus dans un *markup*, comme le montre l'exemple qui suit. Ces différents symboles sont répertoriés dans l'annexe Section A.8 [La fonte Emmentaler], page 718.

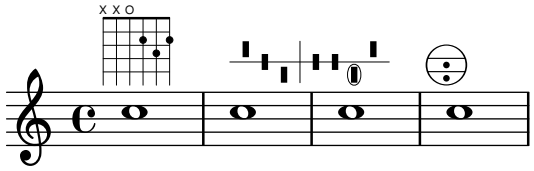
```
\relative {
  c''2
  c'^\markup { \musicglyph "eight" }
  c,4
  c,8._\markup { \musicglyph "clefs.G_change" }
  c16
  c2^\markup { \musicglyph "timesig.neomensural94" }
}
```



La rubrique [Fontes musicales], page 286, contient d'autres informations sur l'impression de glyphes non alphabétiques, tels que des crochets ou accolades.

Le mode *markup* supporte aussi les diagrammes spécifiques à certains instruments :
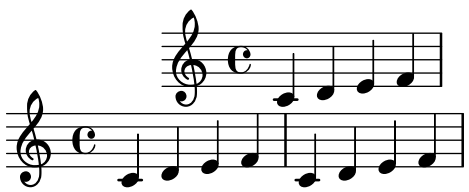
```
\relative {
  c''1^\markup {
    \fret-diagram-terse "x;x;o;2;3;2;"
  }
  c^\markup {
    \harp-pedal "^-v|--ov^"
  }
  c
  c^\markup {
    \combine
      \musicglyph "accordion.discant"
      \combine
        \raise #0.5 \musicglyph "accordion.dot"
        \raise #1.5 \musicglyph "accordion.dot"
  }
}
```



La documentation sur ces diagrammes se trouve à l'annexe Section A.11.5 [Instrument Specific Markup], page 785.

Rien ne s'oppose à ce qu'une étiquette ne comporte une partition.

```
\relative {
  c'4 d^\markup {
    \score {
      \relative { c'4 d e f }
      \layout { }
    }
  }
  e f |
  c d e f
}
```



Les différentes commandes *markup* relatives à la notation musicale sont répertoriées à l'annexe Section A.11.4 [Music], page 779.

Voir aussi

Manuel de notation : Section A.8 [La fonte Emmentaler], page 718, Section “Musique” dans *Manuel de notation*, [Fontes musicales], page 286.

Fichiers d'initialisation : `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L'espacement vertical d'un `\score` contenu dans un objet *markup* se contrôle par la propriété `baseline-skip`. Tout réglage au sein d'un bloc `\paper` est ignoré.

Texte avec sauts de page

Alors que `\markup` s'utilise pour traiter un bloc de texte insécable, `\markuplist` permet, employé en tête de partition, d'obtenir un bloc de lignes réparties différemment et, le cas échéant, sur plusieurs pages.

```
\markuplist {
  \justified-lines {
    Un long texte constitué de lignes justifiées.
    ...
  }
  \wordwrap-lines {
    Un autre grand paragraphe.
    ...
  }
  ...
}
```

Un long texte constitué de lignes justifiées. ...

Un autre grand paragraphe. ...

...

Cette syntaxe prend en charge une liste de *markups* ; il peut s'agir

- d'une suite de commandes générant à leur tour des lignes de texte,
- d'une liste de lignes de texte,
- d'une liste d'étiquettes.

Les différentes commandes permettant de générer des listes de lignes se trouve dans l'annexe Section A.12 [Commandes pour liste de *markups*], page 801, (en anglais).

Commandes prédéfinies

`\markuplist`.

Voir aussi

Manuel de notation : Section A.12 [Commandes pour liste de *markups*], page 801.

Manuel d'extension : Section “Définition d'une nouvelle commande de liste de markups” dans *Extension de LilyPond*.

Fichiers d'initialisation : `scm/define-markup-commands.scm`.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

1.8.3 Fontes

La gestion des fontes dans LilyPond est assurée par plusieurs bibliothèques : *FontConfig* (`https://fontconfig.or`) se charge de répertorier les différentes fontes disponibles ; quant à *Pango* (`https://pango.org`), elle se charge plus particulièrement de leur rendu.

Ce chapitre indique comment accéder aux fontes et polices de caractère dans LilyPond. Vous y apprendrez aussi comment changer de fonte en cours de partition.

Localisation des fontes

En sus de celles déjà installées sur le système, d'autres fontes peuvent s'ajouter au répertoire de FonConfig, ce qui les rendra donc disponibles pour les partitions LilyPond, à l'aide des commandes suivantes :

```
#(ly:font-config-add-font "chemin/au/fichier-fonte")
#(ly:font-config-add-directory "chemin/au/dossier/")
```

Le chemin fourni à ces deux commandes peut être absolu ou relatif, ce qui permet de compiler une partition sur n'importe quel système dans la mesure où les fichiers de fontes concernés sont transmis avec les fichiers sources de la partition.

L'instruction `#(ly:font-config-display-fonts)` permettra de vérifier que la fonte requise est bien détectée par FontConfig. Elle affiche en console la liste exhaustive des fontes du système ainsi que le nom exact qu'il faudra transmettre à LilyPond ; ce nom peut être différent du nom du fichier.

Les fontes se sélectionnent selon les méthodes indiquées dans Voir [Attribution d'une fonte en particulier], page 282, et Voir [Choix des fontes par défaut], page 284.

Familles de fontes

Vous disposez de trois familles de fontes[1] textuelles : *roman* pour la police sérif ou avec empattement, une police sans empattement (*sans* sérif) et une police à chasse fixe (monospace ou *typewriter*). En fonction du moteur de rendu utilisé, ces familles seront mappées selon des alias différents.

Pour le moteur `svg` :

Famille générique	**Famille de fonte SVG**
roman	serif
sans	sans-serif
typewriter	monospace

« serif », « sans-serif » et « monospace » sont des « generic-family » au titre des spécifications SVG et CSS.

[1] Dans sa forme la plus simle, une *famille de fonte* contcontient habituellement les style de police romain, italique, gras et gras italique.

Pour les autres moteurs :

Famille générique	**Fonte par défaut (alias)**	**Listes de définition des alias**
roman	LilyPond Serif	C059, Century SchoolBook URW, Century Schoolbook L, TeX Gyre Schola, DejaVu Serif, . . . , serif
sans	LilyPond Sans Serif	Nimbus Sans, Nimbus Sans L, TeX Gyre Heros, DejaVu Sans, . . . , sans-serif
typewriter	LilyPond Monospace	Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, TeX Gyre Cursor, DejaVu Sans Mono, . . . , monospace

Lorsqu'un caractère est absent de la première fonte listée, il sera remplacé par celui de la fonte suivante..
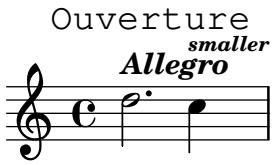
Il est à noter que les fontes URW distribuées avec LilyPond (« C059 », « Nimbus Mono PS » et « Nimbus Sans ») disposent d'une particularité : par défaut, et en complément des ligatures standard telles que « fl » ou « ffi », elles substituent la chaîne « Nr. » par le caractère « Numero Sign » (U+2116) dès lors que le script « latn » est sélectionné. On peut toutefois s'en préserver temporairement par l'insertion d'un caractère de largeur nulle et non jointant (*zero-width non-joiner* ZWNJ, U+200C) entre les caractères « N » et « r ». Les lignes ci-dessous auront pour conséquence que LilyPond insérera toujours un caractère *ZWNJ*.

```
\paper {
  #(add-text-replacements!
      `(("Nr." . ,(format #f "N~ar." (ly:wide-char->utf-8 #x200C)))))
}
```

« LilyPond Serif », « LilyPond Sans Serif » et « LilyPond Monospace » sont des alias de fonte définis dans le fichier de configuration de FontConfig spécifique à LilyPond `00-lilypond-fonts.conf` qui se trouve normalement dans le répertoire `/usr/local/share/lilypond/2.23.2/fonts`.

Chaque famille dispose en principe de différents styles et niveaux de graisse. L'exemple qui suit illustre la manière de changer la famille, le style, la graisse ou la taille. Notez bien que l'argument fourni à `font-size` correspond à la correction à apporter à la taille par défaut.

```
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d''2.^\markup "Allegro"
\override Voice.TextScript.font-size = #-3
c''4^smaller
```



Cette syntaxe s'applique aussi en mode *markup*, bien que celui-ci dispose d'une syntaxe allégée comme nous l'avons vu dans [Sélection de la fonte et de la taille], page 265 :

```
\markup {
  \column {
```

```
    \line {
      \override #'((font-shape . italic) (font-size . 4))
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter) {
        \override #'(font-series . bold) re
        di
      }
      \override #'(font-family . sans) Creta
    }
  }
}
```

Idomeneo,
re di Creta

Fonctionnalités des fontes

Le recours aux fontes OpenType permet d'utiliser certaines fonctionnalités de ces fontes.[2] Toutefois, les fontes OpenType ne disposent pas toutes de l'intégralité de ces fonctionnalités. Dans le cas où la fonctionnalité demandée n'est pas disponible dans la fonte choisie, cette fonctionnalité est tout bonnement ignorée. Les exemples ci-dessous utilisent « TeX Gyre Schola », autrement dit le style romian de la famille.

```
\markup { \override #'(font-name . "TeX Gyre Schola")
          Style normal : Hello HELLO }
\markup { \override #'(font-name . "TeX Gyre Schola")
          \caps { Petites capitales : Hello } }
\markup { \override #'(font-name . "TeX Gyre Schola")
          \override #'(font-features . ("smcp"))
          { Vraies petites capitales : Hello } }

\markup { \override #'(font-name . "TeX Gyre Schola")
          Style numérique normal : 0123456789 }
\markup { \override #'(font-name . "TeX Gyre Schola")
          \override #'(font-features . ("onum"))
          { Style numérique ancien : 0123456789 } }

\markup { \override #'(font-name . "TeX Gyre Schola")
          \override #'(font-features . ("salt 0"))
          { Alternative stylistique 0 : $\epsilon\phi\pi\rho\theta$ } }
\markup { \override #'(font-name . "TeX Gyre Schola")
          \override #'(font-features . ("salt 1"))
          { Alternative stylistique 1 : $\epsilon\phi\pi\rho\theta$ } }

\markup { \override #'(font-name . "TeX Gyre Schola")
          \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Fonctionnalités multiples : Hello 0123456789 $\epsilon\phi\pi\rho\theta$ } }
```

Style normal : Hello HELLO

[2] La sélection de scripts ou de langages des fontes OpenType n'est à ce jour par prise en charge.

Petites capitales : Hello

Vraies petites capitales : Hello

Style numérique normal : 0123456789

Style numérique ancien : 0123456789

Alternative stylistique 0 : εφπρθ

Alternative stylistique 1 : ϵϕϖϱϑ

Fonctionnalités multiples : Hello 0123456789 ϵϕϖϱϑ

Une liste exhaustive des fonctionnalités des fontes OpenType est disponible à l'adresse `https://www.microsoft.com/typography/otspec/featurelist.htm`, et les différents types de fonctionnalité des fontes OpenType sont recensés dans le message `http://lists.gnu.org/archive/html/lilypond-devel/2017-08/msg00004.html`.

En plus de pouvoir jongler entre les différentes fontes prédéfinies, LilyPond vous permet d'en utiliser d'autres, ce qui fait l'objet des deux prochaines parties : [Attribution d'une fonte en particulier], page 282, et [Choix des fontes par défaut], page 284.

Voir aussi

Manuel de notation : Section A.8 [La fonte Emmentaler], page 718, [Notation musicale dans du texte formaté], page 275, Section 5.4.9 [Rotation des objets], page 683, [Sélection de la fonte et de la taille], page 265, Section "Fonte" dans *Manuel de notation*.

Attribution d'une fonte en particulier

La plupart des fontes installées sur le système et reconnues par FontConfig peuvent s'utiliser dans une partition, à l'exception des polices bitmap – qui ne sont pas supportées de par leur conception – et les OpenType Variation Fonts (non prises en charge à ce jour).

LilyPond appelle la fonction `pango_font_description_from_string` de la bibliothèque Pango pour accéder aux fontes ; elle utilise la forme syntaxique suivante pour le nom des fontes.[3]

`[`*`family-list`*`] [`*`style-options`*`]`

où *family-list* est une liste, dont le séparateur est la virgule, des familles éventuellement terminée par une virgule, et *style-options* une liste, dont le séparateur est l'espace, de mots dont chacun désigne un style, une variant, une graisse, une chasse, ou une orientation.

Les termes suivants sont considérés comme des styles : `Normal` (par défaut), `Roman`, `Oblique`, `Italic`.

Les termes suivants sont considérés comme des variantes : `Small-Caps`. Il n'y a pas de variante par défaut.

Les termes suivants sont considérés comme des graisses : `Thin`, `Ultra-Light`, `Extra-Light`, `Light`, `Semi-Light`, `Demi-Light`, `Book`, `Regular` (the default), `Medium`, `Semi-Bold`, `Demi-Bold`, `Bold`, `Ultra-Bold`, `Extra-Bold`, `Heavy`, `Black`, `Ultra-Black`, `Extra-Black`.

[3] La donnée est issue des références de la version 1.46.1 de Pango ; la syntaxe prise en charge par LilyPond est en fait un sous-ensemble de ce que Pango procure.

Les termes suivants sont considérés comme des valeurs de chasse : `Ultra-Condensed`, `Extra-Condensed`, `Condensed`, `Semi-Condensed`, `Semi-Expanded`, `Expanded`, `Extra-Expanded`, `Ultra-Expanded`. Il n'y a pas d'étirement par défaut.

Les termes suivants sont considérés comme des valeurs d'orientation : `Not-Rotated`, `South`, `Upside-Down`, `North`, `Rotated-Left`, `East`, `Rotated-Right`, `West`. Il n'y a pas d'orientation par défaut.

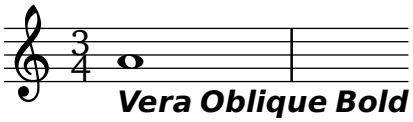
Partant du principe que la syntaxe est correcte, déterminer le nom d'une police n'échoue jamais. Si aucune famille de fonte n'est reconnue (ou aucune famille n'a été donnée), FontConfig renvoie une fonte par défaut dépendant du système. Si aucune option de style n'est reconnue (ou aucune option de style n'a été spécifiée), seront utilisées les valeurs par défaut.

Dans l'exemple qui suit, la fonte de la métrique est déterminée à « Bitstream Charter ». Dans la mesure où aucun style n'est spécifié, FontConfig utilisera les valeurs par défaut comme vu ci-dessus. Pour la chaîne de *markup* sont demandées les familles « Bitstream Vera Sans » et « sans-serif », ce qui indique à FontConfig de rechercher en premier « Bitstream Vera Sans » et, si elle n'est pas disponible, de se reporter sur une famille « sans-serif » générique comme indiqué dans [Familles de fontes], page 279. Le style du *markup* est défini à « Oblique Bold » ; FontConfig tentera donc de trouver une fonte qui soit à la fois penchée et grasse. Si elle échoue, elle tentera de trouver soit une fonte penchée, soit une fonte grasse. Si elle échoue à nouveau, elle essaiera une fonte répondant aux options de style par défaut.

Notez bien que FontConfig vérifie aussi la présence du glyphe requis dans la fonte demandée. Dans le cas où il y serait absent sera essayée une autre fonte, toujours selon le même algotithme, jusqu'à ce qu'elle trouve le glyphe en question. Dans l'éventualité où FontConfig ne trouve nulle part le glyphe approprié sera affiché un symbole de glyphe absent – habituellement un rectangle ou un simple espace blanc.

```
\override Staff.TimeSignature.font-name = "Bitstream Charter"
\override Staff.TimeSignature.font-size = #2
\time 3/4

a'1_\markup {
  \override #'(font-name .
               "Bitstream Vera Sans,sans-serif, Oblique Bold")
    { Vera Oblique Bold }
}
```



Note : Dès lors que l'une des options stylistiques mentionnées ci-dessus fait partie du nom de la (famille) fonte, il est **impératif** de faire suivre ce nom d'une virgule même si aucun style n'est défini. L'exemple typique est « Times New Roman » : spécifier `"Times New Roman"` fera rechercher à FontConfig une police « Times New » de style roman, et c'est seulement si est libellé `"Times New Roman,"` que cette police sera réellement accédée.

Lancer `lilypond` en ligne de commande avec l'option suivante, affiche la liste de toutes les polices disponibles sur votre machine :

```
lilypond -dshow-available-fonts
```

Voir aussi

Manuel de notation : [Choix des fontes par défaut], page 284, [Localisation des fontes], page 279, [Familles de fontes], page 279.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Choix des fontes par défaut

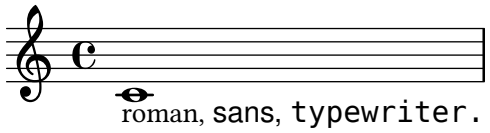
On peut tout à fait modifier le jeu de polices par défaut de LilyPond grâce à un appel de la fonction `make-pango-font-tree`. Il vous faudra alors spécifier les différentes familles, en respectant l'ordre *roman*, *sans empattement* et *monospace*, ainsi qu'un facteur d'échelle. Tout comme indiqué dans [Attribution d'une fonte en particulier], page 282, il peut s'agir d'une liste de polices séparées par une virgule, à ceci près qu'il n'est pas possible de spécifier de style.[4]

Note : `make-pango-font-tree` réinitialise les fontes musicales à leur valeur par défaut, à savoir `emmentaler`.

Dans l'exemple suivant, les fontes seront mises à l'échelle de la taille globale des portées.

```
\paper  {
  #(define fonts
    (make-pango-font-tree "Linux Libertine O"
                          "Nimbus Sans, Nimbus Sans L"
                          "DejaVu Sans Mono"
                          (/ staff-height pt 20)))
}

\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```



La syntaxe suivante permet de substituer une fonte particulière tout en laissant les autres à leur valeur par défaut. L'exemple ci-dessous produit les mêmes effets que celui utilisant `make-pango-font-tree`. De même qu'avec `make-pango-font-tree`, il est possible de fournir une liste de fontes séparées par une virgule pour les catégories `roman`, `sans` et `typewriter`. Dès lors que la taille de portée reste à sa valeur par défaut de 20 pt, l'instruction `#:factor` n'est pas nécessaire.

```
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Linux Libertine O"
     #:sans "Nimbus Sans, Nimbus Sans L"
     #:typewriter "DejaVu Sans Mono"
     ; inutile si taille de portée par défaut
     #:factor (/ staff-height pt 20)
```

[4] Ce qui a pour conséquence que, contrairement au cas d'une fonte particulière, aucune virgule terminale n'est nécessaire.

```
    ))
}
```

Il est aussi posible de substituer les fontes musicales. L'exemple ci-dessous produit les mêmes effets que ceux qui précèdent dans la mesure où les fontes musicales sont fixées à leur valeur par défaut. Pour de plus amples informations, voir Section 3.4.4 [Changement des fontes musicales], page 554.

```
\paper {
  #(define fonts
    (set-global-fonts
     #:music "emmentaler"            ; défaut
     #:brace "emmentaler"            ; défaut
     #:roman "Linux Libertine O"
     #:sans "Nimbus Sans, Nimbus Sans L"
     #:typewriter "DejaVu Sans Mono"
     ; inutile si taille de portée par défaut
     #:factor (/ staff-height pt 20)
    ))
}
```

En tout état de cause, tout appel à `set-global-fonts` réinitialise aussi bien les fontes musicales que les fontes textuelles.[5] Dès lors que l'une de ces catégories n'est pas mentionnée sera utilisée la fonte par défaut y afférente.
Par ailleurs, chaque appel à `set-global-fonts` affecte les fontes du bloc `\book` qui le suit ; chaque bloc `\book` consécutif peut donc disposer de son propre jeu de fontes grâce à un simple appel à `set-global-fonts`, comme ici :

```
\paper {
  #(define fonts
    (set-global-fonts
     ...
    ))
}
\book {
  ...
}

\paper {
  #(define fonts
    (set-global-fonts
     ...
    ))
}
\book {
  ...
}
```

[5] Pour être plus précis, « emmentaler » (avec un « e » minuscule en argument à `#:music` et `#:brace`) constitue un jeu de fontes – et non une famille de fonte au sens de FontConfig – auquel LilyPond a accès et gère directement. Au lieu de plusieurs styles, elle est toutefois fournie dans différentes tailles – voir [Fontes musicales], page 286, et Section 3.4.4 [Changement des fontes musicales], page 554. Les noms de fonte, pour FontConfig, sont « Emmentaler-*taille* », où *taille* est un nombre entre 11, 13, 14, 16, 18, 20, 23 et 26.

Pour ce qui est des accolades, le nom de la fonte pour FontConfig est « Emmentaler-Brace ».

Voir aussi

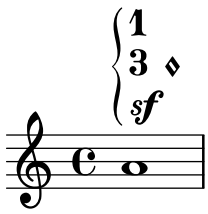
Manuel de notation : [Attribution d'une fonte en particulier], page 282, Section "Fonte" dans *Manuel de notation*, Section 3.4.4 [Changement des fontes musicales], page 554, [Familles de fontes], page 279, [Localisation des fontes], page 279, [Sélection de la fonte et de la taille], page 265.

Fontes musicales

LilyPond neither uses FontConfig nor Pango for accessing music notation fonts but handles them by itself. As a consequence, the interface is different. This section describes how to insert music symbols into markup strings.

Music notation fonts are a collection of specific glyphs that can be accessed with several *encodings*. The following syntax allows LilyPond's various Emmentaler glyphs[6] to be used directly in markup mode:

```
a'1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup "brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup "noteheads.s0petrucci"
  }
}
```



However, all these glyphs except the braces of various sizes are available using the simpler syntax described in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

When accessing braces with the 'fetaBraces' encoding, the size of the brace is specified by the numerical part of the glyph name, in arbitrary units. Any integer from 0 to 575 inclusive may be specified, with 0 giving the smallest brace. The optimum value must be determined by trial and error. These glyphs are all left braces; right braces may be obtained by rotation, voir ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

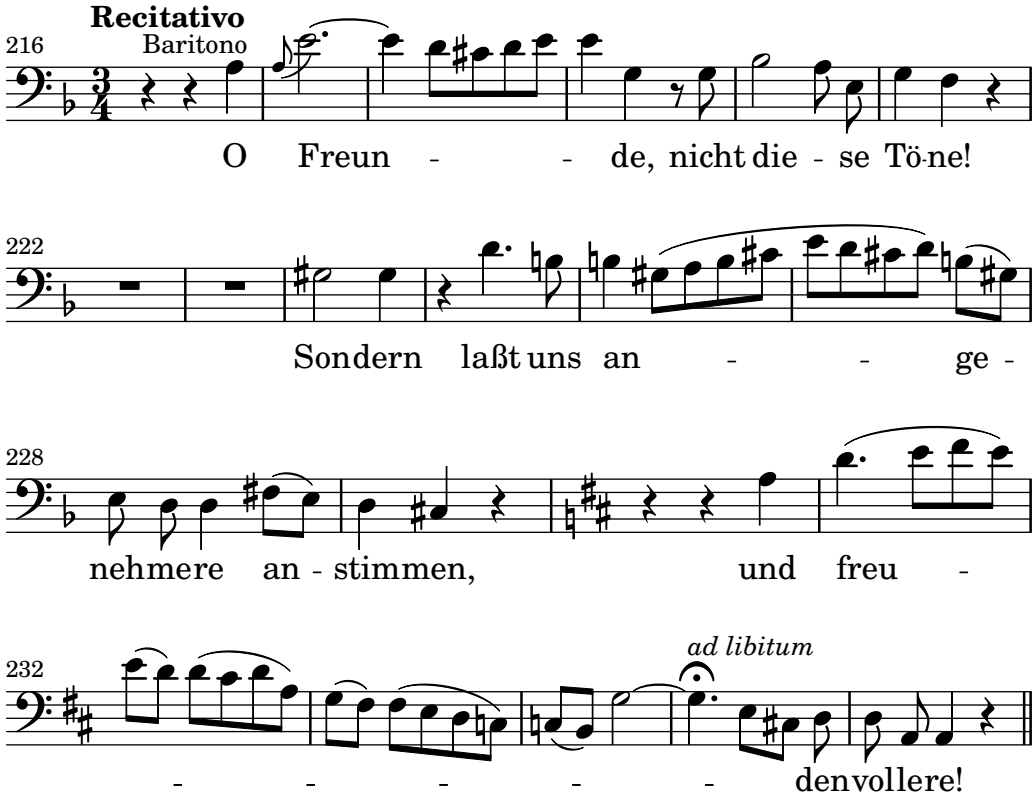
[6] LilyPond's Emmentaler fonts contain three glyph sets: *Feta* (for modern notation), *Parmesan* (for ancient notation), and braces. Both Feta and Parmesan are accessed with the 'fetaMusic' encoding.

2 Notation spécialisée

Ce chapitre explique comment créer la notation musicale spécifique à certains intruments ou certaines époques.

Bien que la musique orchestrale ou d'ensemble de fasse pas l'objet d'une partie distincte, un certain nombre de notions couramment utilisées dans ces types de partitions est abordé dans [Références en matière d'opéra et musique de scène], page 332.

2.1 Musique vocale



Ce chapitre traite de la musique vocale : comment la saisir et comment s'assurer que les paroles s'alignent avec les notes de la mélodie correspondante.

2.1.1 Vue d'ensemble de la musique vocale

En complément de généralités, ce sous-chapitre aborde quelques styles particuliers en terme de musique vocale.

Références en matière de musique vocale

Graver de la musique vocale soulève plusieurs problèmes ; ils sont abordés soit dans ce chapitre, soit dans d'autres parties de la documentation de LilyPond.

- La plupart du temps, les paroles ne sont constituées que de texte simple. Cette forme de notation est abordée dans Section "Écriture de chants simples" dans *Manuel d'initiation*.
- La musique vocale nécessite souvent de recourir au mode `markup`, aussi bien pour des paroles que pour d'autres éléments textuels comme le nom des personnages. Cette syntaxe est expliquée dans [Introduction au formatage de texte], page 263.
- L'impression d'un *ambitus* – ou tessiture – que l'on trouve en tête de certaines partitions, est abordée dans [Ambitus], page 37.

- Les indications de nuance viennent, par défaut, se placer sous la portée. Il en va différemment pour la musique vocale, de telle sorte qu'elles ne soient pas mélangées avec les paroles. Ceci fait l'objet de la rubrique [Mise en forme d'une partition chorale], page 330.

Voir aussi

Glossaire musicologique : Section "ambitus" dans *Glossaire*.

Manuel d'initiation : Section "Écriture de chants simples" dans *Manuel d'initiation*.

Manuel de notation : [Ambitus], page 37, [Introduction au formatage de texte], page 263, [Mise en forme d'une partition chorale], page 330.

Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

Saisie des paroles

Il existe un mode de saisie spécialement adapté aux paroles. On l'introduit avec le mot-clé `\lyricmode`, ou en utilisant `\addlyrics` ou `\lyricsto`. Ce mode permet de saisir des paroles ainsi que leur ponctuation, de telle sorte que le caractère `a` ne sera plus interprété comme une note, un *la* pour les latinistes, mais comme une syllabe. Les syllabes sont saisies comme des notes, mais les hauteurs sont alors remplacées par du texte. Exemple avec une comptine anglaise :

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

Il y a deux manières principales de préciser la place exacte des syllabes : soit en spécifiant explicitement la durée de chaque syllabe – comme dans l'exemple ci-dessus – soit en alignant automatiquement les paroles sur les notes d'une mélodie ou d'une voix en utilisant `\addlyrics` ou `\lyricsto`. La première méthode est abordée plus en détail à la rubrique [Durée explicite des syllabes], page 294, la deuxième à la rubrique [Durée automatique des syllabes], page 291.

Dans les paroles, un mot ou une syllabe commence par une lettre de l'alphabet, et se termine par une espace. Toute syllabe doit donc être séparée d'une autre par une espace, tout autre caractère – chiffre ou ponctuation – étant considéré comme partie intégrante de cette même syllabe. L'exemple suivant comporte une faute de frappe évidente :
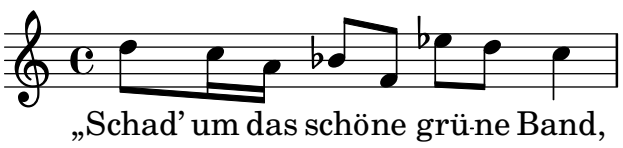
```
\lyricmode { lah lah lah}
```

la dernière syllabe contient une `}` ; il y a de fait un défaut de parité avec l'accolade ouvrante, et la compilation échouera fort probablement. Prenez dès à présent l'habitude de toujours encadrer d'espaces une accolade :

```
\lyricmode { lah lah lah }
```
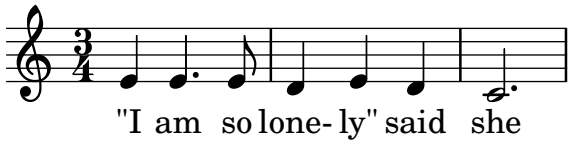
Pour utiliser des lettres accentuées ou des caractères spéciaux – cœurs ou guillemets inversés par exemple – il suffit de les insérer dans le fichier et de veiller à le sauvegarder avec le codage UTF-8. Voir à ce sujet Section 3.3.3 [Caractères spéciaux], page 549, pour plus de détails.

```
\relative { d''8 c16 a bes8 f ees' d c4 }
\addlyrics { „Schad’ um das schö -- ne grü -- ne Band, }
```



Pour utiliser des guillemets informatiques standard, faites-les précéder d'une barre oblique inverse et encadrez d'une paire de guillemets la syllabe ainsi composée :

```
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone- "ly\"" said she }
```



Expliquer exactement comment LilyPond repère le début d'un mot en mode paroles (*Lyrics*) est quelque peu compliqué. En mode *Lyrics*, un mot peut commencer par : tout caractère alphabétique, `_`, `?`, `!`, `:`, `'`, un des codes de contrôle `^A` à `^F` et `^Q` à `^W`, `^Y`, `^^`, tout caractère ASCII de code strictement supérieur à 127, ou enfin l'un des symboles `` ` ``, `'`, `"` ou `^`, s'il est précédé d'une barre oblique inverse.

LilyPond permet de contrôler très finement le rendu des paroles grâce au mode `\markup`, utilisable y compris au sein du mode `\lyricmode`. Des explications complètes sont disponibles au chapitre Section 1.8.2 [Mise en forme du texte], page 263.
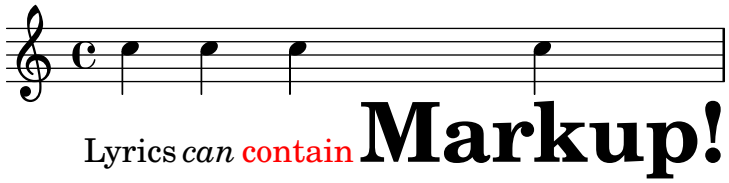
Morceaux choisis

Mise en forme individuelle de syllabes

Le mode *markup* permet d'individualiser la mise en forme de certaines syllabes.

```
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>
```



Voir aussi

Manuel d'initiation : Section "Chansons" dans *Manuel d'initiation*

Manuel de notation : Section 3.3.3 [Caractères spéciaux], page 549, [Durée automatique des syllabes], page 291, [Durée explicite des syllabes], page 294, Section 1.8.3 [Fontes], page 279, Section 1.8.2 [Mise en forme du texte], page 263, Section 5.4.1 [Modes de saisie], page 666.

Référence des propriétés internes : Section "LyricText" dans *Référence des propriétés internes*.

Alignement des paroles sur la mélodie

Les paroles sont interprétées à partir à partir du mode `\lyricmode` et imprimées dans un contexte `Lyrics` – voir Section 5.1.1 [Tout savoir sur les contextes], page 628.

```
\new Lyrics \lyricmode { ... }
```

Deux variantes à `\lyricmode` permettent de plus d'associer un contexte pour synchroniser les syllabes à la musique. La plus commode consiste à ajouter un `\addlyrics` directement après le contenu musical du contexte `Voice` qui devrait se synchroniser avec le contexte `Lyrics` alors implicitement créé. L'instruction `\lyricsto` est plus versatile en ceci qu'elle requiert de spécifier à la fois le contexte `Voice` associé et de créer explicitement un contexte `Lyrics` pour contenir les paroles. Pour de plus amples détails, voir [Durée automatique des syllabes], page 291.

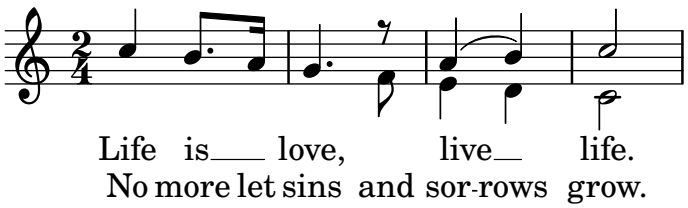
Vous disposez de deux méthodes pour aligner des paroles sur une mélodie :

- Les paroles peuvent s'aligner automatiquement, la durée des syllabes étant déterminée à partir d'un contexte de voix ou, dans certaines circonstances, une mélodie associée, grâce aux commandes `\addlyrics` et `\lyricsto` ou en définissant la propriété `associatedVoice`. Ceci est détaillé à la rubrique [Durée automatique des syllabes], page 291.

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e4 d c2
    }
  >>

% takes durations and alignment from notes in "one"
  \new Lyrics \lyricsto "one" {
    Life is __ _ love, live __ life.
  }

% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



La première ligne de paroles est saisie selon la méthode la plus simple.

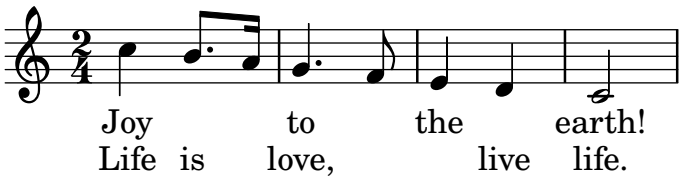
Vous pouvez constater, dans la deuxième ligne, que les paroles s'alignent selon les durées d'une voix différente. Ceci est particulièrement utile lorsque le texte s'agence différemment selon les couplets et que les durées sont accessibles grâce à des contextes `Voice` particuliers. Pour de plus amples détails, rendez-vous à la rubrique Section 2.1.3 [Couplets], page 319.

- Les paroles s'aligneront indépendamment de la valeur des notes dès lors que vous utiliserez le mode `\lyricmode` et affecterez explicitement leur durée à chaque syllabe.

```
<<
  \new Voice = "one" \relative {
    \time 2/4
    c''4 b8. a16 g4. f8 e4 d c2
  }
```

```
% uses previous explicit duration of 2;
  \new Lyrics \lyricmode {
    Joy to the earth!
  }

% explicit durations, set to a different rhythm
  \new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
  }
>>
```



La première ligne de paroles ne s'aligne pas vraiment sur les notes parce qu'aucune durée n'a été spécifiée. En fait, LilyPond adopte la dernière durée mentionnée, un 2, et l'applique à chaque mot.

La deuxième ligne illustre la manière d'aligner des paroles sans tenir compte de la durée des notes. Cette façon de procéder permet de traiter un alignement différent selon les couplets lorsqu'il n'y a pas moyen de déduire les durées à partir d'un contexte musical ; la rubrique [Durée explicite des syllabes], page 294, aborde ceci plus en détails. Cette technique permet aussi d'ajouter des dialogues, comme indiqué à la rubrique [Dialogue et musique], page 339.

Voir aussi

Manuel d'initiation : Section "Alignement des paroles sur une mélodie" dans *Manuel d'initiation*.

Référence des propriétés internes : Section "Lyrics" dans *Référence des propriétés internes*.

Durée automatique des syllabes

Les paroles peuvent être automatiquement alignées sous une mélodie, de trois manières différentes :

- en utilisant la commande `\lyricsto` pour spécifier le contexte de voix qui contient la mélodie,
- en introduisant les paroles par la commande `\addlyrics`, placée juste après le contexte `Voice` qui contient la mélodie,
- en définissant la propriété `associatedVoice` pour synchroniser les paroles avec un autre contexte de voix, ce à n'importe quel moment.

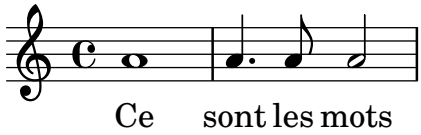
Ces trois méthodes permettent d'ajouter les traits d'union séparant les syllabes d'un même mot ainsi que d'indiquer la tenue de la dernière syllabe. Ceci fait l'objet de la rubrique [Traits d'union et de prolongation], page 299.

Le contexte `Voice` contenant la mélodie sur laquelle les paroles vont s'aligner doit rester actif, au risque de voir la suite du texte disparaître. Ceci peut se produire lorsqu'il y a des moments où l'on ne chante pas. La rubrique Section 5.1.3 [Conservation d'un contexte], page 634, vous indiquera comment maintenir un contexte actif.

Utilisation de \lyricsto

Vous pouvez aligner automatiquement des paroles sous une mélodie en spécifiant à l'aide de la commande \lyricsto le contexte de voix qui contient cette mélodie :

```
<<
  \new Voice = "melodie" \relative {
    a'1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melodie" {
    Ce sont les mots
  }
>>
```



Cette commande adapte les paroles aux notes de la voix (contexte Voice dans le jargon LilyPond) *melodie*. Ce contexte Voice doit exister avant l'affectation des paroles par \lyricsto. La commande \lyricsto introduit automatiquement le mode \lyricmode. Les paroles viendront par défaut se placer en dessous des notes. Pour un autre positionnement, voir [Positionnement vertical des paroles], page 302.

Utilisation de \addlyrics

La commande \addlyrics n'est en fait qu'une manière plus aisée d'écrire de la musique vocale dans une structure LilyPond plus complexe.
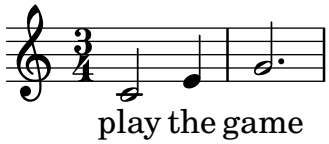
```
{ MUSIQUE }
\addlyrics { PAROLES }
```

revient au même que

```
\new Voice = "blah" { MUSIQUE }
\new Lyrics \lyricsto "blah" { PAROLES }
```

En voici un exemple :
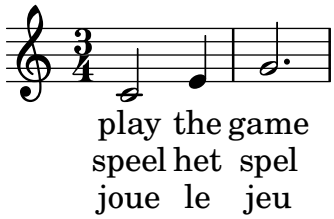
```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}
```



On peut ajouter davantage de couplets en multipliant le nombre de sections \addlyrics.

```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
```

```
}
```



Cependant, la commande `\addlyrics` ne peut pas gérer les constructions polyphoniques et ne permet pas d'associer des paroles à un contexte `TabVoice`. Dans ces cas là, mieux vaut employer `\lyricsto`.

Utilisation de associatedVoice
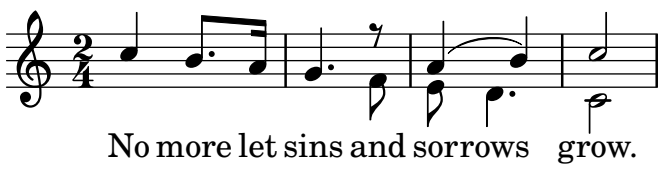
La propriété `associatedVoice` permet de basculer de mélodie pour la synchronisation des paroles. Elle s'emploie de la manière suivante :

```
\set associatedVoice = "lala"
```

La valeur que vous attribuez à cette propriété (ici `"lala"`) doit désigner un contexte `Voice` nommé, sans quoi les mélismes ne seront pas imprimés correctement.

Voici un exemple de cette manière de procéder :

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e8 d4. c2
    }
  >>
% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



Voir aussi

Manuel de notation : Section 5.1.3 [Conservation d'un contexte], page 634, [Traits d'union et de prolongation], page 299.

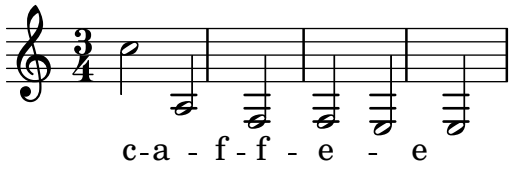
Durée explicite des syllabes

On peut aussi se passer de `\addlyrics`, `\lyricsto` et `associatedVoice` pour saisir des paroles. Dans ce cas, les syllabes sont entrées comme des notes – du texte remplaçant les hauteurs – ce qui veut dire que vous devez définir leur durée explicitement.
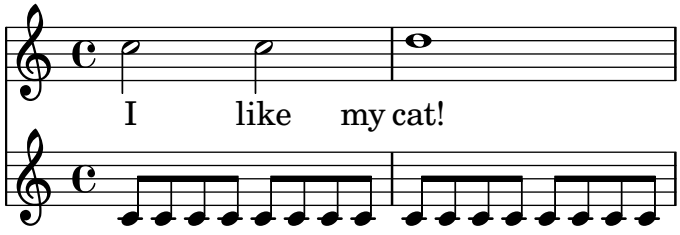
Les traits d'union seront imprimés entre les syllabes, à l'inverse des mélismes puisqu'il n'y a pas de voix associée.

Voici deux illustrations de cette technique :

```
<<
  \new Voice = "melody" {
    \time 3/4
    c''2 a f f e e
  }
  \new Lyrics \lyricmode {
    c4. -- a -- f -- f -- e2. -- e
  }
>>
```
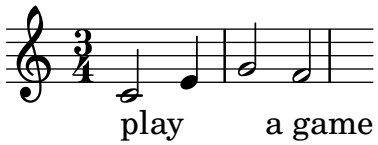



```
<<
  \new Staff {
    \relative {
      c''2 c2
      d1
    }
  }
  \new Lyrics {
    \lyricmode {
      I2 like4. my8 cat!1
    }
  }
  \new Staff {
    \relative {
      c'8 c c c c c c c
      c8 c c c c c c c
    }
  }
>>
```



Cette manière de procéder est tout à fait adaptée lorsqu'un fond musical accompagne des dialogues – voir [Dialogue et musique], page 339.

Les syllabes seront alignées selon la dérogation apportée à la propriété `self-alignment-X` :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    c'2 e4 g2 f
  }
  \new Lyrics \lyricmode {
    \override LyricText.self-alignment-X = #LEFT
    play1 a4 game4
  }
>>
```



Voir aussi

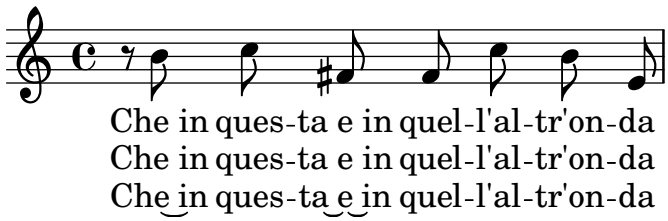
Manuel de notation : [Dialogue et musique], page 339.

Référence des propriétés internes : Section “Lyrics” dans *Référence des propriétés internes*, Section “Voice” dans *Référence des propriétés internes*.

Plusieurs syllabes sur une note

Pour attribuer plus d’une syllabe à une même note, vous pouvez soit les mettre entre guillemets, soit utiliser le caractère souligné (`_`) pour obtenir une espace, ou bien encore utiliser un tilde (`~`) pour obtenir une liaison entre les syllabes.

```
{
  \relative {
    \autoBeamOff
    r8 b' c fis, fis c' b e,
  }
  \addlyrics
  {
    % Ensure hyphens are visible
    \override LyricHyphen.minimum-distance = #1.0
    Che_in ques -- ta_e_in quel -- l'al -- tr'on -- da
  }
  \addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
  \addlyrics { Che~in ques -- ta~e~in quel -- l'al -- tr'on -- da }
}
```



Voir aussi

Référence des propriétés internes : Section “LyricCombineMusic” dans *Référence des propriétés internes*.

Plusieurs notes pour une même syllabe

Parfois, tout particulièrement en musique médiévale ou baroque, plusieurs notes correspondent à une même syllabe. Ces vocalises sont appelées Section “mélismes” dans *Glossaire*. La syllabe à vocaliser est traditionnellement alignée par la gauche sur la première note du mélisme.
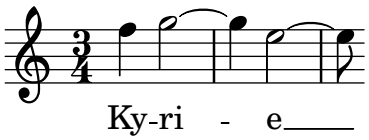
Lorsqu'un mélisme tombe sur une syllabe autre que la dernière d'un mot, un trait d'union étiré, indiqué par un double tiret `--` dans le fichier source, reliera cette syllabe à la suivante.

Lorsqu'un mélisme tombe sur la dernière syllabe d'un mot ou que ce mot n'en comporte qu'une, l'usage est d'indiquer la « tenue » jusqu'à la dernière note de la vocalise. Ceci s'obtient en ajoutant un double caractère souligné `__` après cette syllabe.

Vous disposez de cinq méthodes pour indiquer la présence d'un mélisme :

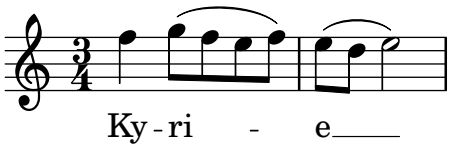
- Une liaison de prolongation constitue de fait un mélisme :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g2 ~ |
    4 e2 ~ |
    8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```



- LilyPond considère une liaison d'articulation comme un mélisme – il s'étendra de la première à la dernière note couverte par cette liaison. Il s'agit là de la façon traditionnelle de saisir des paroles :
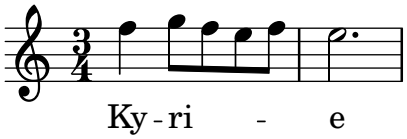
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```



  Notez bien qu'une liaison de phrasé – indiquée par `\(...\)` – n'a aucune incidence sur la gestion des mélismes.

- LilyPond considère des notes regroupées par une ligature manuelle comme un mélisme, si tant est que la procédure de ligature automatique a été désactivée – voir [Définition des règles de ligature automatique], page 90.
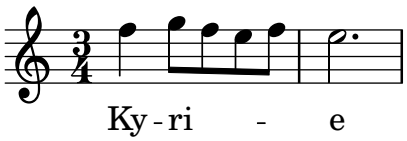
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \autoBeamOff
    f''4 g8[ f e f]
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



Ceci ne peut, vous en conviendrez, prendre en compte des durées plus longues que la croche.

- LilyPond considère un groupe de notes non liées, mais encadrées par `\melisma` et `\melismaEnd`, comme constituant un mélisme :
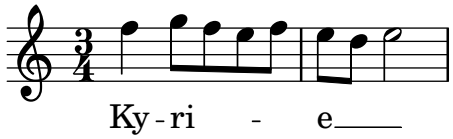
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8
    \melisma
    f e f
    \melismaEnd
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



- Vous pouvez indiquer un mélisme directement dans les paroles, à l'aide d'un caractère souligné simple _ pour chaque note faisant partie de la vocalise :
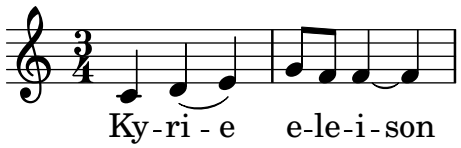
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 f e f
    e8 d e2
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ _ _ e __ _ _
```

```
  }
>>
```



Vous pouvez totalement désactiver l'interprétation des liaisons de prolongation ou d'articulation et des ligatures apparaissant dans une mélodie comme fait générateur d'un mélisme. Il suffit en ce cas de définir `melismaBusyProperties` :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] f4 ~ 4
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e e -- le -- i -- son
  }
>>
```
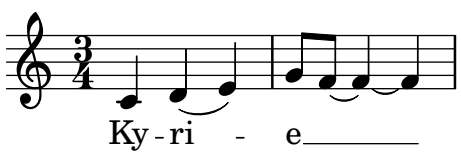


Certains réglages de `melismaBusyProperties` permettent de prendre en compte ou non les liaisons de tenue, les liaisons d'articulation et les ligatures dans la détection automatique des mélismes – voir `melismaBusyProperties` à la rubrique Section "Tunable context properties" dans *Référence des propriétés internes*.

Dans le cas où les indications de mélisme doivent être totalement ignorées, il vous faudra alors activer `ignoreMelismata` – voir [Rythme différent selon le couplet], page 321.

Lorsque, dans un passage où la propriété `melismaBusyProperties` est active, survient un mélisme, vous pouvez l'indiquer dans les paroles par un simple caractère souligné pour chaque note à inclure :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] ~ 4 ~ f
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ e __ _ _ _
  }
>>
```



Commandes prédéfinies

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

Voir aussi

Glossaire musicologique : Section “melisma” dans *Glossaire*.

Manuel d’initiation : Section “Alignement des paroles sur une mélodie” dans *Manuel d’initiation*.

Manuel de notation : [Alignement des paroles sur la mélodie], page 289, [Définition des règles de ligature automatique], page 90, [Durée automatique des syllabes], page 291, [Rythme différent selon le couplet], page 321.

Référence des propriétés internes : Section “Tunable context properties” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Certains mélismes ne sont pas détectés automatiquement ; vous devrez alors prolonger vous-même les syllabes concernées à l’aide d’un double caractère souligné.

Traits d’union et de prolongation

Un mélisme sur la dernière syllabe d’un mot est indiqué par une longue ligne horizontale basse s’étirant jusqu’à la syllabe suivante. Une telle ligne, que nous appellerons prolongateur ou extenseur, s’obtient en saisissant ‘ `__` ’ (notez les espaces entourant le double caractère souligné).

> **Note :** Dans une partition, les mélismes, ou vocalises, sont matérialisés par une ligne de prolongation. On l’indique par un double caractère souligné. Lorsqu’ils sont assez courts, ces mélismes peuvent s’indiquer par un souligné unique, ce qui aura pour effet de sauter une note à chaque fois et de ne pas imprimer de ligne.

Un trait d’union séparant deux syllabes d’un même mot s’obtient en saisissant ‘ `--` ’ (notez les espaces entourant le tiret double). Ce trait d’union sera centré entre les deux syllabes et sa longueur sera proportionnelle à l’espace les séparant.

Dans les partitions très serrées, les traits d’union peuvent ne pas être imprimés. Cet inconvénient peut être contrôlé par les propriétés `minimum-distance` pour gérer l’espace minimum entre deux syllabes, et `minimum-length`, seuil en deçà duquel il n’y a pas de trait d’union, toutes deux attachées à l’objet `LyricHyphen`.

Un trait d’union ne sera pas, par défaut, imprimé après un saut de ligne lorsque le nouveau système débute sur une nouvelle syllabe. Basculer la propriété `after-line-breaking` à `#t` permet de répéter le trait d’union en pareille situation.

Voir aussi

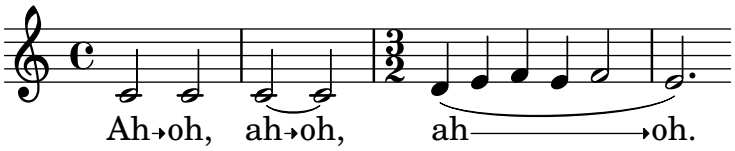
Référence des propriétés internes : Section “LyricExtender” dans *Référence des propriétés internes*, Section “LyricHyphen” dans *Référence des propriétés internes*

Changement graduel de voyelle

Un changement graduel de voyelle (ou une consonne maintenue) peut s’indiquer en ajoutant une flêche entre les syllabes, à l’aide de la commande `\vowelTransition` – voir Gould p. 452–453. La flêche indique la longueur de la transition et sera par défaut toujours affichée – de l’espace sera ajouté en cas d’espacement contraint. Une liaison de tenue ou de vocalise indique une absence

d'articulation si ce n'est un changement de voyelle. La taille minimale de la flêche se règle à l'aide de la propriété `minimum-length` de l'objet `VowelTransition`.

```
{
  c'2 c'
  \set melismaBusyProperties = #'()
  c'2 ~ c'
  \time 3/2
  d'4( e' f' e' f'2
  e'2.)
}
\addlyrics
{
  Ah \vowelTransition oh,
  ah \vowelTransition oh,
  ah \vowelTransition _ _ _ _
  oh.
}
```



Voir aussi

Glossaire musicologique : Section "vowel transition" dans *Glossaire*.
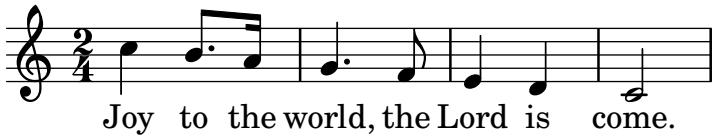
Référence des propriétés internes : Section "VowelTransition" dans *Référence des propriétés internes*.

2.1.2 Situations particulières en matière de paroles

Travail avec des paroles et variables

Vous pouvez créer des variables pour contenir les paroles, dès lors que vous faites appel au mode approprié :

```
musicOne = \relative {
  c''4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
    \new Voice = "one" {
      \time 2/4
      \musicOne
    }
    \new Lyrics \lyricsto "one" {
      \verseOne
    }
  >>
}
```
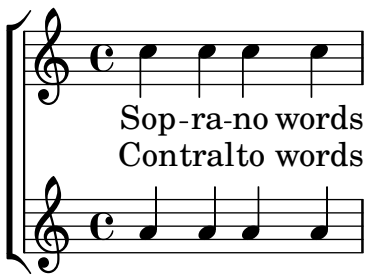


La fonction `\lyricmode` permet de définir une variable pour les paroles. Point n'est besoin de spécifier les durées si vous utilisez `\addlyrics` ou `\lyricsto` lorsque vous y faites référence.

Pour une organisation différente ou plus complexe, mieux vaut commencer par créer et alimenter les variables contenant mélodies et paroles, puis définir la hiérarchie des portées et des lignes de paroles, et enfin combiner correctement mélodies et paroles à l'aide de la commande `\context`. Vous serez ainsi assuré que la voix à laquelle il est fait référence par `\lyricsto` aura bien été préalablement définie, comme dans l'exemple suivant :

```
sopranoMusic = \relative { c''4 c c c }
contraltoMusic = \relative { a'4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  >>
}
```



Voir aussi

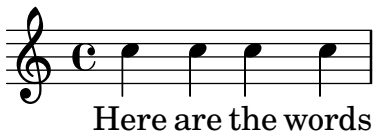
Manuel de notation : [Positionnement vertical des paroles], page 302.

Référence des propriétés internes : Section “LyricCombineMusic” dans *Référence des propriétés internes*, Section “Lyrics” dans *Référence des propriétés internes*.

Positionnement vertical des paroles

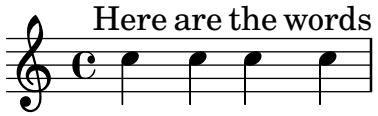
Selon le type de musique, les paroles apparaîtront au-dessus ou au-dessous d’une portée ou bien entre deux portées. Positionner des paroles en dessous de la portée à laquelle elles se rattachent est de loin la chose la plus simple : il suffit de mentionner le contexte de paroles après le contexte de portée :

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Positionner les paroles au-dessus de la portée se fait de deux manières différentes, le plus simple étant d’utiliser la même syntaxe que ci-dessus, à ceci près que la ligne de paroles sera positionnée de manière explicite :

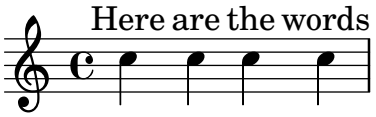
```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Autre façon de procéder, cette fois-ci en deux étapes. Nous commençons par déclarer un contexte `Lyrics` que nous laissons vide, puis les contextes `Staff` et `Voice`. Dans un deuxième

temps, nous ajoutons l'instruction `\context` et la commande `\lyricsto` pour affecter les paroles au contexte de voix en question. Voici comment cela se présente :

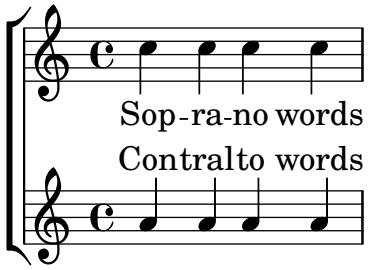
```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Lorsque deux voix sont isolées chacune sur une portée, vous pouvez placer les paroles entre les deux portées en utilisant l'une des méthodes que nous venons de voir. En voici un exemple, basé sur la deuxième méthode :

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative { a'4 a a a }
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        Sop -- ra -- no words
      }
    }
    \context Lyrics = "contraltos" {
```

```
      \lyricsto "contraltos" {
        Con -- tral -- to words
      }
    }
  >>
}
```



Vous pouvez générer d'autres combinaisons de paroles et portées à partir de ces exemples, ou en examinant ce qui figure à la rubrique Section "Modèles pour ensemble vocal" dans *Manuel d'initiation* du manuel d'initiation.

Morceaux choisis

Espacement des paroles selon les pratiques de la version 2.12

Le moteur d'espacement vertical des paroles a évolué avec la version 2.14. Celles-ci peuvent donc se retrouver positionnées différemment.

Le moteur adoptera les usages de la version 2.12 une fois que vous aurez réglé certaines propriétés des contextes `Lyric` et `Staff`.

```
global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

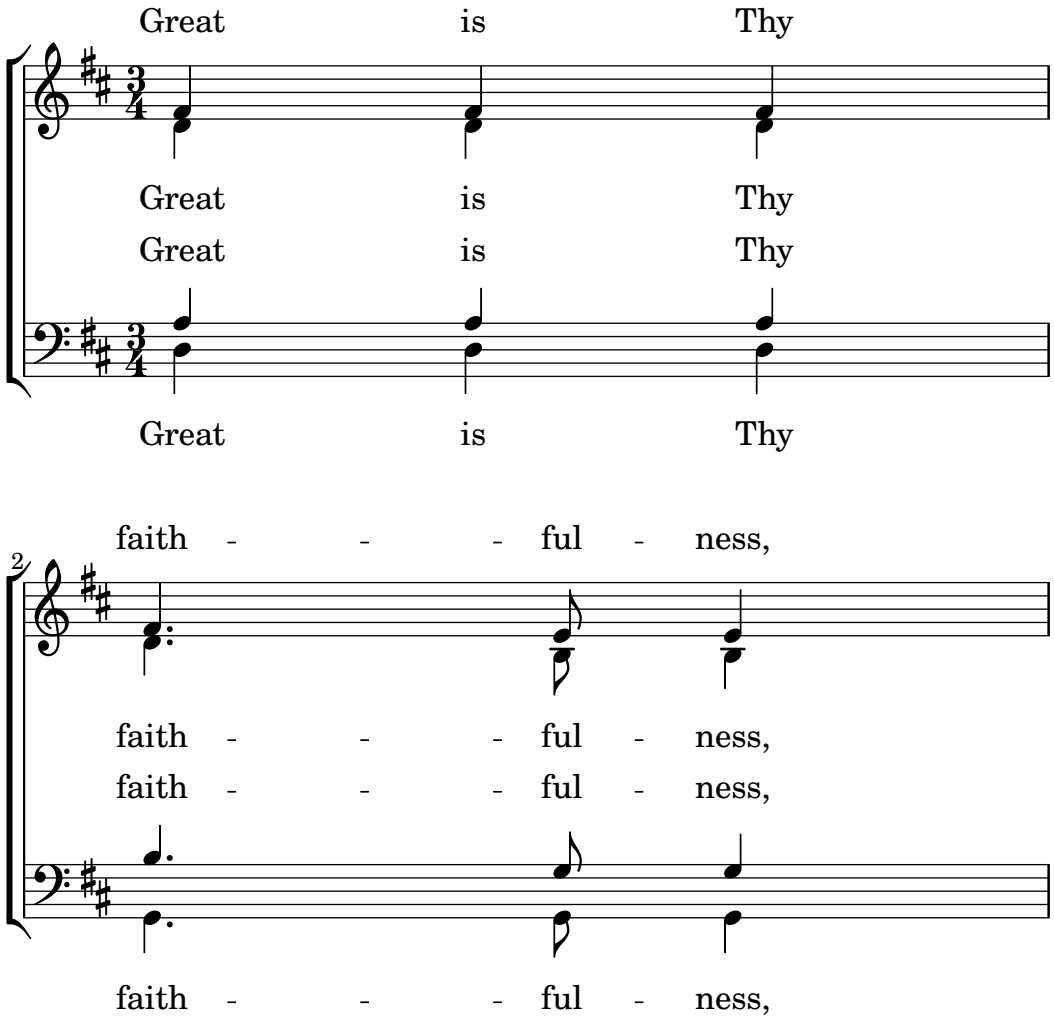
tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}
```

```
words = \lyricmode {
  Great is Thy faith -- ful -- ness,
}

\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo  \global \bassMusic
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup.staff-affinity = ##f
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
    \context {
      \Staff
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
```

```
  }
}
```



Voir aussi

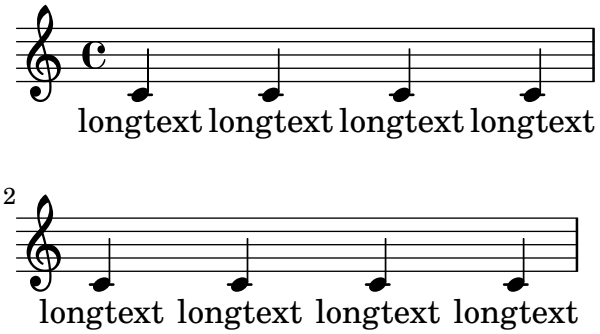
Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*.

Manuel de notation : Section 5.1.2 [Création et référencement d'un contexte], page 630, Section 5.1.7 [Ordonnancement des contextes], page 646.

Positionnement horizontal des syllabes

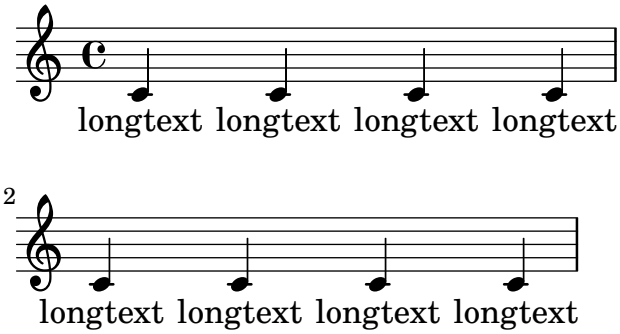
La propriété `minimum-distance` de l'objet `LyricSpace` permet d'accroître l'espacement des paroles.

```
\relative c' {
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```



Pour que ce réglage s'applique à toute la partition, définissez-le dans le bloc `\layout`.

```
\score {
  \relative {
  c' c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace.minimum-distance = #1.0
    }
  }
}
```



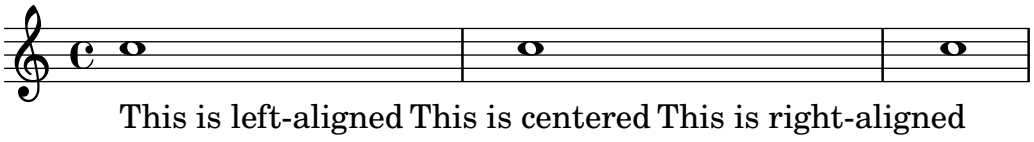
Morceaux choisis

Alignement des syllabes

L'alignement horizontal des paroles peut se gérer à l'aide de la propriété `self-alignment-X` de l'objet `LyricText`. Les valeurs `-1` ou `LEFT` produiront un alignement par la gauche, les valeurs `0` ou `CENTER` un alignement centré, et les valeurs `1` ou `RIGHT` un alignement par la droite.

```
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
```

```
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}
```



Problèmes connus et avertissements

L'assurance que tous les scripts textuels et les paroles resteront bien à l'intérieur des marges requiert des ressources non négligeables. Afin de réduire le temps de traitement, vous pouvez désactiver cette fonctionnalité en ajoutant

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Pour s'assurer que les paroles ne seront pas traversées par des barres de mesure, il faut ajouter

```
\layout {
  \context {
    \Lyrics
      \consists "Bar_engraver"
      \consists "Separating_line_group_engraver"
      \hide BarLine
  }
}
```

Paroles et reprises
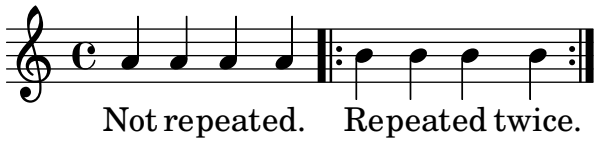
La répétition de *fragments musicaux* est abordée de manière détaillée dans un Section "chapitre spécifique" dans *Manuel de notation*. Nous nous intéresserons ici aux moyens d'ajouter des paroles à des reprises.

Reprises simples

Les paroles attachées à un fragment musical répété devraient adopter rigoureusement la même construction que la musique, si tant est qu'elles ne diffèrent pas d'une fois sur l'autre.
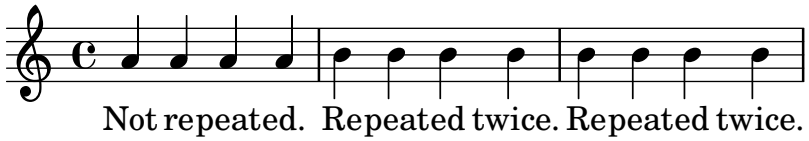
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- ed twice. }
      }
    }
  >>
```

```
}
```



Les mots seront alors correctement répétés si la reprise est développée.

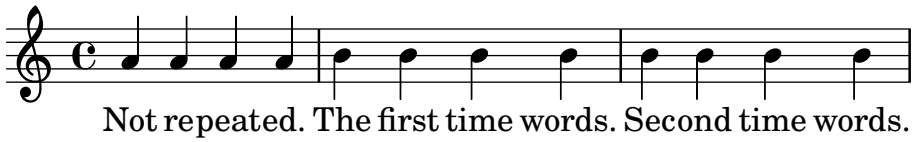
```
\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
          \relative {
            a'4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Not re -- peat -- ed.
          \repeat volta 2 { Re -- peat -- ed twice. }
        }
      }
    >>
  }
}
```



Lorsque la reprise est développée et que les paroles diffèrent, saisissez le texte normalement :
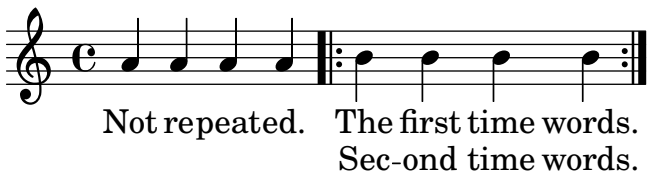
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        The first time words.
        Sec -- ond time words.
      }
    }
```

```
  >>
}
```



Lorsque les paroles diffèrent pour une reprise non développée – utilisation de `volta` au lieu de `unfold` – les paroles en question doivent être saisies dans des contextes `Lyrics` séparés ; ils seront combinés dans une section parallèle :

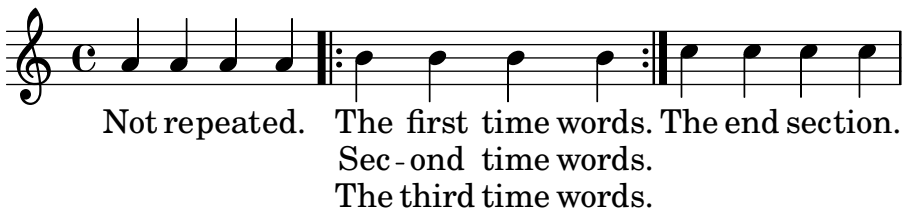
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "melody"
          Sec -- ond time words.
        }
      >>
    }
  >>
}
```



Et ce quel que soit le nombre de « couplets » :

```
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
        \relative {
          a'4 a a a
          \repeat volta 3 { b4 b b b }
          c4 c c c
        }
      }
    }
```

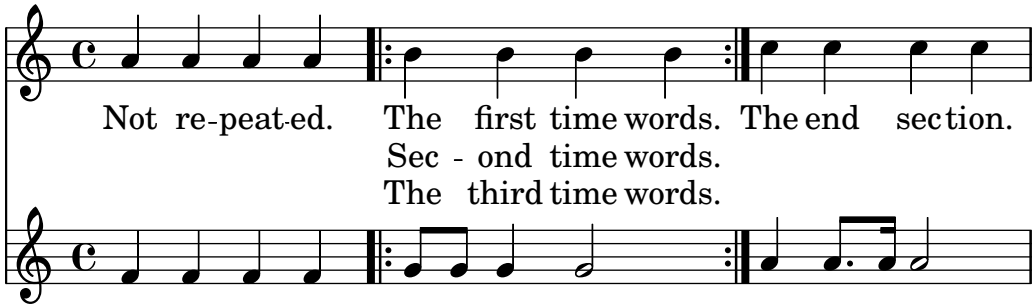
```
    \new Lyrics \lyricsto "singleVoice" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          Sec -- ond time words.
        }
        \new Lyrics {
          \set associatedVoice = "singleVoice"
          The third time words.
        }
      >>
      The end sec -- tion.
    }
  >>
}
```



Cependant, lorsque la partition comporte plusieurs portées, cas typique d'un `ChoirStaff`, les paroles des deuxième et troisième couplets seront repoussées sous la dernière portée. L'instruction `alignBelowContext` permet alors de les repositionner correctement :

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 3 { b4 b b b }
          c4 c c c
        }
      }
    }
    \new Lyrics = "firstVerse" \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }
        \new Lyrics = "secondVerse"
        \with { alignBelowContext = "firstVerse" } {
          \set associatedVoice = "melody"
          Sec -- ond time words.
        }
        \new Lyrics = "thirdVerse"
        \with { alignBelowContext = "secondVerse" } {
          \set associatedVoice = "melody"
          The third time words.
```

```
        }
      >>
      The end sec -- tion.
    }
    \new Voice = "harmony" {
      \relative {
        f'4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2
      }
    }
  >>
}
```
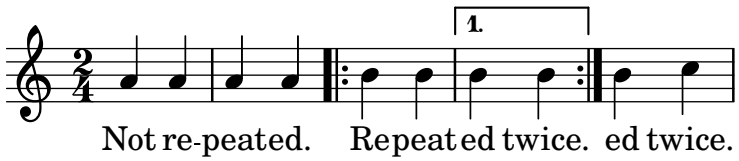


Reprises avec alternative

Les paroles d'un fragment répété, lorsqu'elles sont identiques et qu'aucune alternative ne débute par un silence, peuvent adopter la même construction que la musique. Ceci permet par ailleurs une expansion correcte à la fois de la musique et des paroles lors de l'utilisation de `\unfoldRepeats`.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b }
          \alternative {
            \volta 1 { b b }
            \volta 2 { b c }
          }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- }
        \alternative {
          \volta 1 { ed twice. }
          \volta 2 { ed twice. }
        }
      }
    }
```

```
  >>
}
```

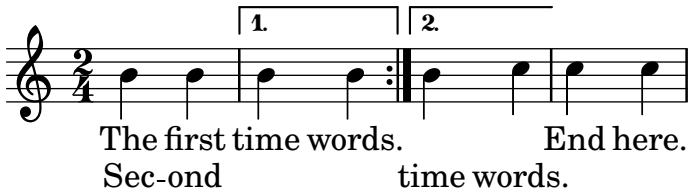


Cette identité de structure n'est toutefois pas possible lorsque les paroles sont différentes ou que l'un des blocs `\alternative` débute par un silence. Des instructions `\skip` devront venir s'insérer dans les paroles pour « sauter » les notes des alternatives qui ne les concernent pas.

N'utilisez pas de simple caractère souligné pour sauter une note. N'oubliez pas qu'un caractère souligné indique un mélisme ; la syllabe précédente sera donc alignée à gauche.

> **Note :** La commande `\skip` doit comporter une durée quelle qu'elle soit – elle sera toujours ignorée lorsque les paroles sont associées à une mélodie à l'aide de `\addlyrics` ou `\lyricsto`. Chaque `\skip` correspond à une seule note quelle qu'en soit la durée.

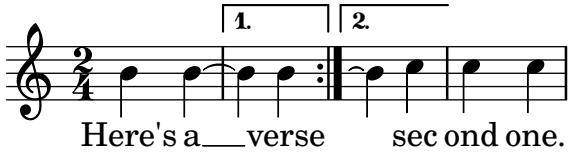
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b }
          \alternative {
            \volta 1 { b b }
            \volta 2 { b c }
          }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        The first time words.
        \repeat unfold 2 { \skip 1 }
        End here.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Sec -- ond
        \repeat unfold 2 { \skip 1 }
        time words.
      }
    }
  >>
}
```



Lorsqu'une note se prolonge sur les alternatives, la tenue est indiquée normalement pour la première alternative, et à l'aide de l'instruction `\repeatTie` pour les suivantes. Cette liaison « de répétition » pose problème en matière d'alignement des paroles puisque la longueur de l'alternative est accrue en raison de la liaison.

D'autre part, une liaison de prolongation crée un mélisme qui sera effectif pour la première alternative, mais pas pour les autres. La solution pour « recaler » les paroles consiste à désactiver temporairement la détection automatique de mélismes et insérer des « blancs ».

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \set melismaBusyProperties = #'()
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
          }
          \unset melismaBusyProperties
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        \repeat volta 2 { Here's a __ }
        \alternative {
          \volta 1 { \skip 1 verse }
          \volta 2 { \skip 1 sec }
        }
        ond one.
      }
    }
  >>
}
```

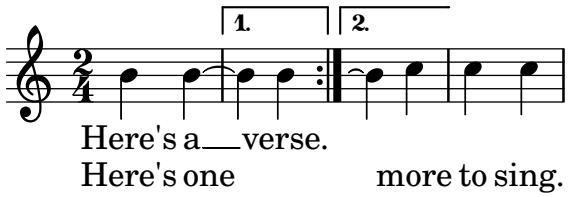


Notez bien que l'utilisation conjointe de `\unfoldRepeats` et de `\repeatTie` entraîne l'impression d'une double liaison, sauf à supprimer les `\repeatTie`.

Lorsque les paroles sont différentes sur le fragment répété, la construction avec `\repeat` est inefficace ; vous devrez alors insérer des blancs :

```
\score {
```
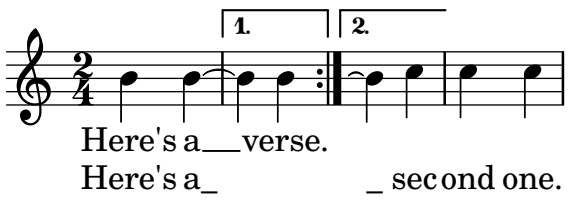
```
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
          }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's one
        \repeat unfold 2 { \skip 1 }
        more to sing.
      }
    }
  >>
}
```



Les indications de mélisme et traits d'union en début d'alternative doivent être insérées manuellement :

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
          }
          c4 c
        }
```

```
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's "a_"
        \skip 1
        "_" sec -- ond one.
      }
    }
  >>
}
```



Voir aussi
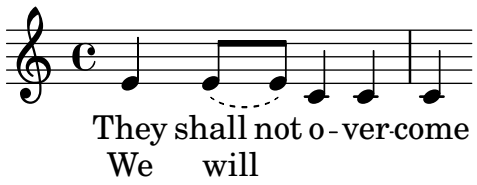
Manuel de notation : Section 5.1.3 [Conservation d'un contexte], page 634, Section 1.4 [Répétitions et reprises], page 156.

Paroles alternatives

Il arrive parfois, dans un fragment répété, qu'une note soit divisée pour répondre au texte. Vous pouvez indiquer cette adaptation rythmique en désactivant temporairement la détection automatique des mélismes tout en spécifiant ces mélismes au niveau des paroles :
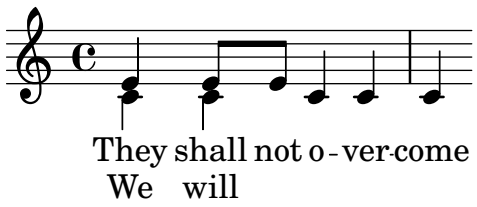
```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        \set melismaBusyProperties = #'()
        \slurDown
        \slurDashed
        e4 e8 ( e ) c4 c |
        \unset melismaBusyProperties
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _
    }
  >>
```

```
}
```



En donnant un nom à chaque voix et en leur attribuant spécifiquement des paroles, vous pourrez traiter le cas où notes et rythme diffèrent d'une fois sur l'autre :

```
\score {
  <<
    \new Voice = "melody" {
      \relative {
        <<
          {
            \voiceOne
            e'4 e8 e
          }
          \new Voice = "splitpart" {
            \voiceTwo
            c4 c
          }
        >>
        \oneVoice
        c4 c |
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```



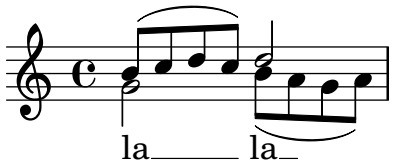
Il n'est pas rare, en musique chorale, qu'une voix se divise pendant plusieurs mesures. Bien qu'une construction du type `<< {...} \\ {...} >>`, où deux expressions musicales (ou plus) séparées par des doubles obliques inversées peuvent sembler être le moyen adéquat de définir cette division, **toutes** les expressions qu'elle contient seront assignées à de **nouveaux contextes de voix**, ce qui aura pour effet qu'aucune parole ne leur sera affectée – les paroles sont attachées au contexte de voix initial. Il vaut mieux construire ce passage comme une polyphonie temporaire – voir [Polyphonie sur une portée], page 179.

Polyphonie et paroles communes

Lorsque deux voix au rythme différent partagent les mêmes paroles, l'alignement des syllabes sur l'une des voix peut gêner la lecture de l'autre voix. Par exemple, la deuxième extension de syllabe ci-dessous est trop courte puisque les paroles ne sont alignées que sur la voix du haut :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
words = \lyricmode { la __ la __ }

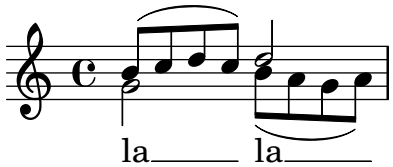
\new Staff <<
  \new Voice = "sopranoVoice" { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new Lyrics \lyricsto "sopranoVoice" \words
>>
```



Le résultat attendu sera obtenu grâce à l'alignement des paroles sur un contexte `NullVoice` supplémentaire, celui-ci contenant une combinaison judicieuse des deux voix. Les notes du contexte `NullVoice`, bien que n'apparaissant pas sur la version imprimable, peuvent servir à aligner correctement les syllabes :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```

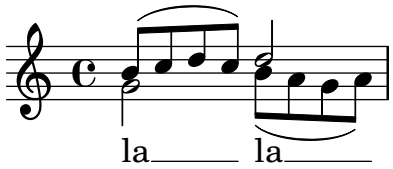


Cette façon de procéder permet par ailleurs d'utiliser la fonction `\partCombine` qui, normalement, ne peut s'utiliser avec des paroles :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice \partCombine \soprano \alto
  \new NullVoice = "aligner" \aligner
```

```
  \new Lyrics \lyricsto "aligner" \words
>>
```



Problèmes connus et avertissements

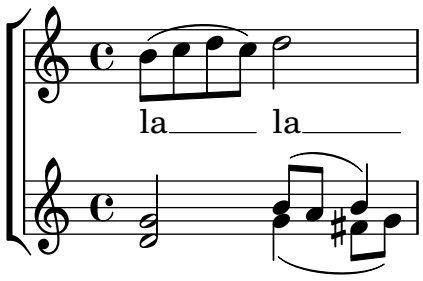
La commande `\addlyrics` ne peut traiter que des paroles attachées à un contexte `Voice` ; elle ne peut donc s'utiliser avec un `NullVoice`.

La fonction `\partCombine` est abordée en détails dans [Regroupement automatique de parties], page 189.

Pour finir, cette méthode est aussi utilisable lorsque les voix sont sur des portées différentes, et ne se limite pas à deux voix :

```
soprano = \relative { b'8( c d c) d2 }
altoOne = \relative { g'2 b8( a b4) }
altoTwo = \relative { d'2 g4( fis8 g) }
aligner = \relative { b'8( c d c) d( d d d) }
words = \lyricmode { la __ la __ }

\new ChoirStaff \with { \accepts NullVoice } <<
  \new Staff \soprano
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
  \new Staff \partCombine \altoOne \altoTwo
>>
```

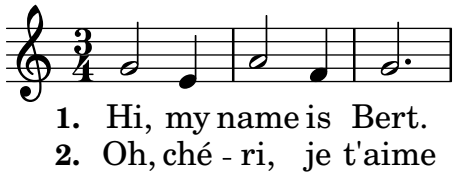


2.1.3 Couplets

Numérotation des couplets

On peut ajouter un numéro aux couplets en définissant la variable `stanza` :

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = "1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = "2. "
  Oh, ché -- ri, je t'aime
}
```



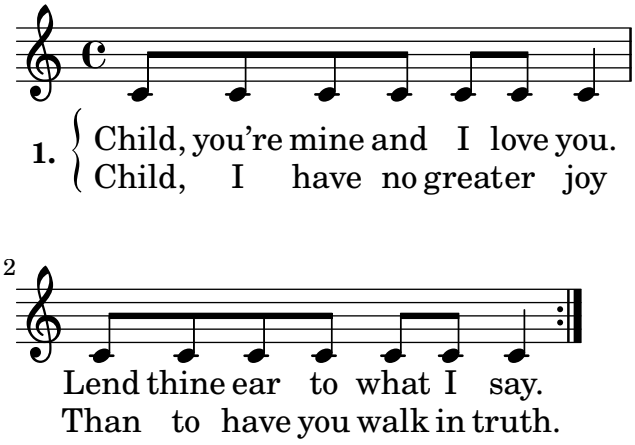
Ces numéros se placeront juste avant le début de la première syllabe. Deux lignes de couplet peuvent aussi être regroupées, par exemple dans le cas d'une reprise avec des paroles différentes.

```
leftbrace = \markup {
  \override #'(font-encoding . fetaBraces)
  \lookup "brace80"
}

stanzaOneOne = \lyricmode {
  \set stanza = \markup {
    \column { \vspace #.33 "1. "}
    \leftbrace
  }
  Child, you're mine and I love you.
  Lend thine ear to what I say.
}

stanzaOneThree = \lyricmode {
  Child, I have no great -- er joy
  Than to have you walk in truth.
}

\new Voice {
  \repeat volta 2 {
    c'8 c' c' c' c' c' c'4
    c'8 c' c' c' c' c' c'4
  }
}
\addlyrics { \stanzaOneOne }
\addlyrics { \stanzaOneThree }
```



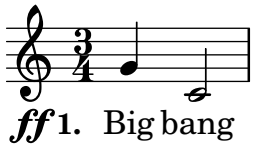
Indication de nuance dans les couplets

Lorsque des couplets ont des nuances différentes, vous pouvez ajouter une nuance en regard de chacun d'eux. L'objet `StanzaNumber` contient tout ce qui se place avant les paroles du cou-

plet. Pour des raisons techniques, vous devrez définir la variable `stanza` en dehors du mode `\lyricmode`.

```
text = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
}

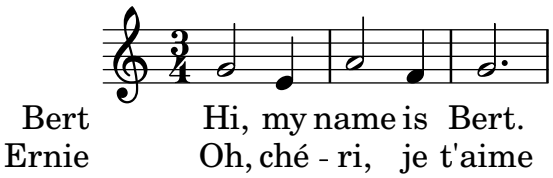
<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \text
>>
```



Indication du personnage et couplets

On peut également ajouter le nom de chaque rôle ; ils s'imprimeront au début de chaque ligne comme les noms d'instrument. Il faut pour cela définir `vocalName`, et `shortVocalName` pour une version abrégée.

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = "Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = "Ernie "
  Oh, ché -- ri, je t'aime
}
```



Rythme différent selon le couplet

Il arrive assez souvent que le paroles de différents couplets, bien qu'attachées à une même mélodie, ne s'articulent pas de la même manière. La commande `\lyricsto` est cependant capable de gérer de telles situations.

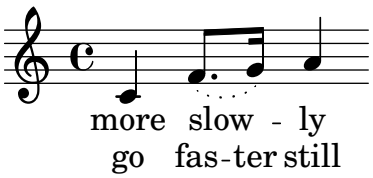
Mélismes dans certaines strophes seulement

Il peut survenir que les paroles comportent un mélisme pour l'un des couplets, mais plusieurs syllabes pour d'autres. Une solution consiste à ignorer temporairement les mélismes dans le

couplet ayant le plus de syllabes. Il suffit pour cela de définir la propriété `ignoreMelismata` à l'intérieur du contexte `Lyrics`.

Petit détail qui a son importance : l'activation de `ignoreMelismata` doit **précéder** la syllabe à partir de laquelle elle s'appliquera :

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
    \unset ignoreMelismata
    still
  }
>>
```



Problèmes connus et avertissements
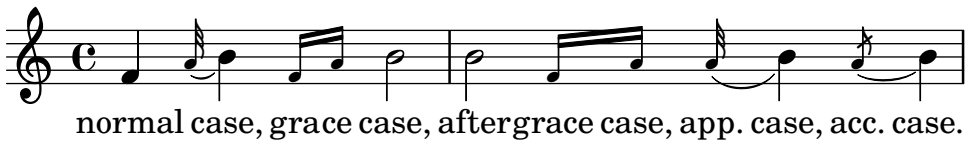
Contrairement aux autres utilisations de l'instruction `\set`, il n'est pas possible de la faire précéder d'un `\once` dans le cas de `\set ignoreMelismata`. Les paroles affectées par la propriété `ignoreMelismata` **doivent** être encadrées respectivement d'un `\set` et d'un `\unset`.

Syllabe sur note de passage

L'utilisation de la commande `\lyricsto` ne permet pas, par défaut, d'assigner une syllabe à des notes d'ornement – introduites par la commande `\grace`. Vous pouvez cependant y parvenir grâce à la propriété `includeGraceNotes` :

```
<<
  \new Voice = melody \relative {
    f'4 \appoggiatura a32 b4
    \grace { f16 a16 } b2
    \afterGrace b2 { f16[ a16] }
    \appoggiatura a32 b4
    \acciaccatura a8 b4
  }
  \new Lyrics
  \lyricsto melody {
    normal
    \set includeGraceNotes = ##t
```
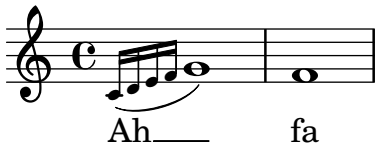
```
    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
    acc. case.
  }
>>
```



Problèmes connus et avertissements

Tout comme pour la propriété `associatedVoice`, la propriété `includeGraceNotes` doit être activée au moins une syllabe avant celle qui viendra s'attacher à la note d'ornement. Dans le cas où cette note se trouve être la première de la pièce, vous devrez recourir à une clause `\with` ou introduire une section `\context` dans le bloc `\layout` :

```
<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

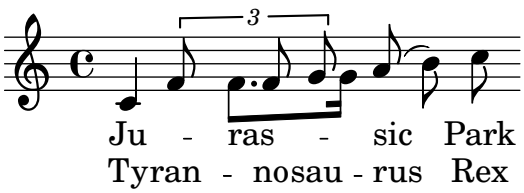


Basculer vers une mélodie alternative

On peut créer des variations plus complexes à partir d'une mélodie à plusieurs voix. Les paroles peuvent suivre l'une ou l'autre des lignes mélodiques, et même basculer de l'une à l'autre si l'on modifie la propriété `associatedVoice`. Dans cet exemple,

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \tuplet 3/2 {
          % show associations clearly.
          \override NoteColumn.force-hshift = #-3
          f8 f g
        }
```

```
      }
      {
        \voiceTwo
        f8.[ g16]
        \oneVoice
      } >>
    a8( b) c
  }
  \new Lyrics \lyricsto "lahlah" {
    Ju -- ras -- sic Park
  }
  \new Lyrics \lyricsto "lahlah" {
    % Tricky: need to set associatedVoice
    % one syllable too soon!
    \set associatedVoice = alternative % applies to "ran"
    Ty --
    ran --
    no --
    \set associatedVoice = lahlah % applies to "rus"
    sau -- rus Rex
  } >>
```



le texte du premier couplet s'aligne de manière habituelle sur la mélodie nommée « lahlah ». Mais le second couplet, tout d'abord rattaché au contexte `lahlah`, bascule sur la mélodie `alternative` pour les syllabes « ran » à « sau » grâce aux lignes

```
\new Lyrics \lyricsto "lahlah" {
  \set associatedVoice = alternative % s'applique à "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = lahlah % s'applique à "rus"
  sau -- rus Rex
}
```

où `alternative` désigne le nom du contexte `Voice` qui contient le triolet.

Notez bien où apparaît la commande `\set associatedVoice` – une syllabe en avance, ce qui est tout à fait correct.

> **Note :** La commande `\set associatedVoice` **doit** intervenir une syllabe *avant* celle qui sera suivie par la bascule. Autrement dit, une modification de la voix associée sera effective une syllabe plus tard que prévu. Il ne s'agit en aucun cas d'une bogue, la raison en est purement technique.

Paroles en fin de partition

Il peut parfois s'avérer opportun d'aligner un seul couplet sur la mélodie et de présenter tous les autres en bloc à la fin du morceau. Ces couplets additionnels peuvent être inclus dans une

section `\markup` en dehors du bloc `\score` principal. Vous pourrez noter qu'il existe différentes méthodes pour indiquer les sauts de ligne en mode *markup*. On peut saisir du texte en continu avec `string-lines` et y insérer des `\n` ou couper automatiquement selon la saisie, ou bien utiliser `\wordwrap-string`. Dans le cas de formatages différent au sein du bloc, il vaut mieux utiliser une combinaison de `\line` et `\column`.
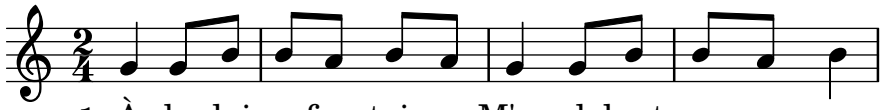
```
melody = \relative {
  \time 2/4
  g'4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

text = \lyricmode {
\set stanza = "1." À la clai- re fon- tai- ne,
M'en al- lant pro- me- ner...
}

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
>>
  \layout { }
}
\markup {
  \column {
    \string-lines
      "Couplet 2. \n Sous les feuilles d'un chêne \n Je me suis fait sécher..."
  }
}
\markup{
  \column {
    \string-lines
      "Couplet 3.
       Chante, rossignol, chante,
       Toi qui as le cœur gai..."
  }
}
\markup {
  \column {
    \line \italic { Couplet 4. }
    \line { J'ai perdu mon ami}
    \line { Sans l'avoir mérité... }
  }
}
\markup {
  \wordwrap-string "
    Couplet 5.

    Je voudrais que la rose

    Fût encore au rosier..."
}
```



1. À la clai- re fon- tai- ne, M'en al- lant pro- me- ner...

Couplet 2.
Sous les feuilles d'un chêne
Je me suis fait sécher...

Couplet 3.
Chante, rossignol, chante,
Toi qui as le cœur gai...

Couplet 4.
J'ai perdu mon ami
Sans l'avoir mérité...

Couplet 5.
Je voudrais que la rose
Fût encore au rosier...

Paroles sur plusieurs colonnes en fin de partition

Si les couplets sont vraiment nombreux, il est possible de les imprimer sur plusieurs colonnes. L'exemple suivant vous montre comment procéder pour que le numéro du couplet soit en retrait à gauche, comme c'est traditionnellement le cas.
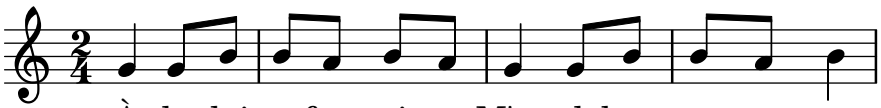
```
melody = \relative {
  \time 2/4
  g'4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

text = \lyricmode {
  \set stanza = "1." À la clai- re fon- tai- ne,
  M'en al- lant pro- me- ner...
}

\score{ <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
   >>
  \layout { }
}

\markup {
  \fill-line {
    % décalage par rapport à la marge de gauche
    % peut être supprimé si l'espace sur la page est réduit
    \hspace #0.1
    \column {
      \line { \bold "2."
        \column {
          "Sous les feuilles d'un chêne"
```

```
          "Je me suis fait sécher..."
        }
      }
      % ajout d'espace vertical entre les couplets
      \combine \null \vspace #0.1
      \line { \bold "3."
        \column {
          "Chante, rossignol, chante,"
          "Toi qui as le cœur gai..."
        }
      }
    }
    % ajout d'espace horizontal entre les colonnes
    \hspace #0.1
    \column {
      \line { \bold "4."
        \column {
          "J'ai perdu mon ami"
          "Sans l'avoir mérité..."
        }
      }
      % ajout d'espace vertical entre les couplets
      \combine \null \vspace #0.1
      \line { \bold "5."
        \column {
          "Je voudrais que la rose"
          "Fût encore au rosier..."
        }
      }
    }
    % décalage par rapport à la marge de droite
    % peut être supprimé si l'espace sur la page est réduit
    \hspace #0.1
  }
}
```



1. À la clai- re fon- tai- ne, M'en al- lant pro- me- ner...

2. Sous les feuilles d'un chêne
Je me suis fait sécher...

3. Chante, rossignol, chante,
Toi qui as le cœur gai...

4. J'ai perdu mon ami
Sans l'avoir mérité...

5. Je voudrais que la rose
Fût encore au rosier...

Voir aussi

Référence des propriétés internes : Section “LyricText” dans *Référence des propriétés internes*, Section “StanzaNumber” dans *Référence des propriétés internes*.

2.1.4 Chansons

Références en matière de chanson

Une chanson se présente la plupart du temps sous la forme de trois portées : une pour la mélodie surmontant un système pianistique pour l'accompagnement ; les paroles du premier couplet s'accolent sous la mélodie. S'il n'y a que deux ou trois couplets, et que cela n'est pas gênant au niveau de l'aspect général, tous peuvent prendre place entre la mélodie et l'accompagnement. Dans le cas contraire, le premier couplet sera imprimé sous la mélodie et les suivants après la partition, sous forme de blocs de texte indépendants.

Tous les éléments qui permettent d'imprimer des chansons sont examinés à différents endroits de la documentation de LilyPond :

- L'agencement des portées est abordé au chapitre Section 1.6.1 [Gravure des portées], page 197.
- Les spécificités du piano sont abordées au chapitre Section 2.2 [Instruments utilisant des portées multiples], page 353.
- L'affectation de paroles à une ligne mélodique est abordée au chapitre Section 2.1.1 [Vue d'ensemble de la musique vocale], page 287.
- Le positionnement des paroles fait l'objet d'une Section "rubrique dédiée" dans *Manuel de notation*.
- La gestion des couplets est abordée dans un Section "chapitre spécifique" dans *Manuel de notation*.
- L'harmonisation d'une chanson est souvent indiquée par des noms d'accord en surplomb de la mélodie. Ceci est abordé au chapitre Section 2.7.2 [Gravure des accords], page 450.
- L'impression de diagrammes d'accord, lorsque l'accompagnement est fait à la guitare, est expliqué à la rubrique « Tablatures sous forme d'étiquette », au chapitre Section 2.4.1 [Vue d'ensemble des cordes frettées], page 369.

Voir aussi

Manuel d'initiation : Section "Chansons" dans *Manuel d'initiation*.

Manuel de notation : Section 2.1.3 [Couplets], page 319, Section 2.7.2 [Gravure des accords], page 450, Section 1.6.1 [Gravure des portées], page 197, Section 2.2 [Instruments utilisant des portées multiples], page 353, [Positionnement vertical des paroles], page 302, Section 2.1.1 [Vue d'ensemble de la musique vocale], page 287.

Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

Feuille de chant

Une simple feuille de chant s'obtient en combinant une partie vocale et son harmonisation. La syntaxe appropriée est expliquée en détails au chapitre Section 2.7 [Notation des accords], page 444.

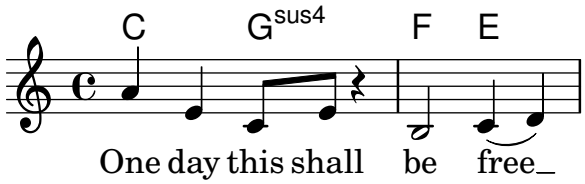
Morceaux choisis

Chanson simple

Assembler des noms d'accords, une mélodie et des paroles permet d'obtenir la partition d'une chanson :

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
```

```
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Manuel de notation : Section 2.7 [Notation des accords], page 444.

2.1.5 Chorale

Nous allons voir, dans les paragraphes qui suivent, les particularités de la musique chorale, qu'il s'agisse de motet, d'oratorio ou de simple partie de chœur.

Références en matière de chorale

Une partition pour chœur comporte habituellement de deux à quatre portées regroupées dans un `ChoirStaff`. L'accompagnement, s'il y en a un, se présente sous la forme d'un système pianistique – un `PianoStaff` – en dessous du chœur ; il s'agira d'une simple réduction dans le cas d'une œuvre *a capella*. Les notes de chaque pupitre font l'objet d'un contexte `Voice` distinct. Ces contextes `Voice` peuvent se voir groupés ou non sur une même portée.

Les paroles sont traitées dans des contextes `Lyrics` qui viendront se placer tantôt sous la portée, tantôt au-dessus et au-dessous de la portée si elle contient deux voix.

Certaines composantes d'une partition pour chœur sont examinées à différents endroits de la documentation de LilyPond :

- La création pas à pas d'une partition pour chœur se trouve dans le manuel d'initiation, à la rubrique Section "Partition pour chœur à quatre voix mixtes" dans *Manuel d'initiation*. LilyPond dispose aussi d'un canevas automatisé qui simplifie grandement la saisie d'une partition pour chœur SATB disponible à la rubrique Section "Gabarits préprogrammés" dans *Manuel d'initiation*.
- Plusieurs exemples et canevas sont regroupés dans le manuel d'initiation, à la rubrique Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*.
- Des informations détaillées sur les contextes `ChoirStaff` et `PianoStaff` sont disponibles au chapitre [Regroupement de portées], page 199.
- Les formes de notation particulière, telle que celle utilisée en *Sacred Harp* et assimilées, sont abordées au chapitre [Têtes de note à forme variable], page 43.
- Lorsque plusieurs pupitres sont regroupés sur la même portée, les hampes, liaisons, etc. de la voix supérieure sont orientées vers le haut, et inversement pour la voix inférieure. L'utilisation de `\voiceOne` et `\voiceTwo` est expliquée au chapitre [Polyphonie sur une portée], page 179.
- La division temporaire d'un pupitre, ce qui correspond à un passage polyphonique temporaire, est expliquée à la section [Polyphonie sur une portée], page 179.

Commandes prédéfinies

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

Voir aussi

Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*, Section "Partition pour chœur à quatre voix mixtes" dans *Manuel d'initiation*.

Manuel de notation : Section 5.1.7 [Ordonnancement des contextes], page 646, [Polyphonie sur une portée], page 179, [Regroupement de portées], page 199, [Têtes de note à forme variable], page 43.

Morceaux choisis. . . : Section "Musique vocale" dans *Morceaux choisis*.

Référence des propriétés internes. . . : Section "ChoirStaff" dans *Référence des propriétés internes*, Section "Lyrics" dans *Référence des propriétés internes*, Section "PianoStaff" dans *Référence des propriétés internes*.

Mise en forme d'une partition chorale

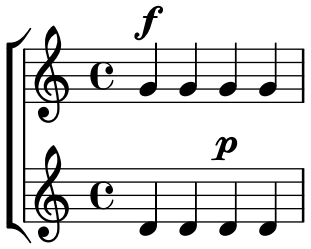
Une partition pour chœur sur quatre portées, avec ou sans accompagnement, présente traditionnellement deux systèmes par page. Selon la taille du papier, vous pourrez être amené à effectuer quelques ajustements aux réglages par défaut, notamment en raison des points suivants :

- La taille des portées a des répercussions sur l'ensemble des éléments de la partition. Voir Section 4.2.2 [Définition de la taille de portée], page 585.
- La distance séparant les systèmes, les portées et les paroles peut s'ajuster de manière séparée, comme expliqué au chapitre Section 4.4 [Espacement vertical], page 595.
- La mise en évidence des différentes dimensions permet d'appréhender avec finesse le réglage des variables d'espacement vertical et, pourquoi pas, de faire tenir la partition sur moins de pages, comme l'explique la rubrique Section 4.6 [Réduction du nombre de pages de la partition], page 624.
- Lorsque le nombre de systèmes varie d'une page à l'autre, il est judicieux de l'indiquer visuellement, en suivant les instructions de la rubrique [Séparation des systèmes], page 204.
- Pour de plus amples détails quant aux propriétés liées au formatage, consultez le chapitre Section 4.1 [Mise en forme de la page], page 571.

Les indications de nuance se placent traditionnellement sous la portée, ce qui n'est pas le cas en matière de musique vocale dans le but d'éviter toute collision avec les paroles. La commande prédéfinie `\dynamicUp` attachée à un contexte `Voice` permet de positionner les nuances au-dessus de la portée. Dans le cas où il y en aurait plusieurs, cette commande devra apparaître dans chacun des contextes `Voice` qui le requiert. Vous pouvez aussi opter pour la forme développée, comme dans l'exemple ci-dessous, pour que cela s'applique à toutes les portées de la partition – changez `\Score` en `\ChoirStaff` s'il y a d'autres parties que celles du chœur.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative { g'4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
        \relative { d'4 d d\p d }
      }
    }
  >>
  \layout {
    \context {
```

```
      \Score
      \override DynamicText.direction = #UP
      \override DynamicLineSpanner.direction = #UP
    }
  }
}
```



Commandes prédéfinies

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Voir aussi

Manuel de notation : Section 4.2.2 [Définition de la taille de portée], page 585, Section 4.4 [Espacement vertical], page 595, Section 4.6.1 [Mise en évidence de l'espacement], page 624, Section 4.1 [Mise en forme de la page], page 571, Section 4.2 [Mise en forme de la partition], page 583, Section 4.6.2 [Modification de l'espacement], page 625, Section 4.6 [Réduction du nombre de pages de la partition], page 624, Section 4.3 [Sauts], page 588, [Séparation des systèmes], page 204.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

Morceaux choisis

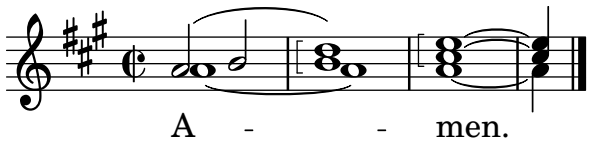
Utilisation d'un `arpeggioBracket` *pour rendre les divisions plus évidentes*

Un crochet d'arpège (`arpeggioBracket`) permet de mettre en évidence les divisions d'un pupitre en l'absence de hampe, comme on le voit régulièrement dans les partitions pour chœur.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
          <b d>1\arpeggio)
          <cs e>\arpeggio ~
          <cs e>4
        }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
```

```
        a1 ~
        a
        a ~
        a4 \bar "|."
      }
    >>
  }
  \layout { ragged-right = ##t }
}
```



Voir aussi

Manuel de notation : Section 1.3.3 [Signes d'interprétation sous forme de ligne], page 146.

2.1.6 Opéras et musiques de scène

Tout ce qui permet d'exécuter un opéra ou une œuvre scénique accompagnée de musique se présente généralement sous l'une ou plusieurs des formes suivantes :

- Un *Conducteur* destiné au chef d'orchestre. Il comprend l'intégralité des parties d'orchestre et des chanteurs, ainsi que des citations du livret pour les passages déclamés.
- Un *matériel d'orchestre*, autrement dit une partition pour chacun des pupitres de l'orchestre ou de l'ensemble instrumental.
- Une *partition de chœur* regroupant toutes les parties vocales avec accompagnement au piano. Cet accompagnement est souvent une réduction d'orchestre où les différents instruments sont indiqués. Les partitions de chœur comprennent parfois des indications de mise en scène ainsi que des extraits du livret.
- Une *partition de choriste* qui ne comprend que les parties vocales – donc sans accompagnement. Elle peut être augmentée du livret.
- Un *livret* contenant l'intégralité des dialogues et le texte des passages chantés. On y trouve aussi très souvent les indications de mise en scène. Bien que LilyPond soit capable de « typographier » un livret, n'oubliez pas qu'il n'y a dans ce cas pas de musique, et que d'autres outils pourraient être mieux appropriés.

La plupart de ce qui est nécessaire à la mise en forme d'un opéra ou d'une musique de scène est disséminé dans la somme documentaire de LilyPond. Nous commencerons par rappeler ces différents éléments, avant que d'en examiner certaines particularités adaptées aux styles opératique et scénique.

Références en matière d'opéra et musique de scène

En plus des formations scéniques ou vocales, la plupart des notions qui suivent peuvent s'appliquer aux orchestres et autres musiques d'ensemble.

- Un conducteur contient un certain nombre de portées et de nombreuses paroles. Les manières d'agencer les portées sont indiquées à la rubrique [Regroupement de portées], page 199, et les façons de les combiner à la rubrique [Imbrication de regroupements de portées], page 202.
- Les portées vides sont la plupart du temps éliminées d'un conducteur ou d'une partition de chœur. La réalisation d'une telle partition – les anglophones la disent « à la française » – est expliquée à la rubrique [Masquage de portées], page 212.

- La génération d'un matériel d'orchestre fait l'objet de la rubrique Section 1.6.3 [Écriture de parties séparées], page 217. D'autres parties du chapitre consacré à la notation spécialisée vous seront utiles selon l'orchestration de la pièce. Tous les instruments ne sont pas accordés pareil ; vous trouverez des informations à ce sujet à la rubrique [Instruments transpositeurs], page 28.
- Lorsque le nombre de systèmes varie d'une page à l'autre, il peut être judicieux de les mettre en évidence, en suivant les indications de la rubrique [Séparation des systèmes], page 204.
- Les différentes propriétés impliquées dans la mise en page sont répertoriées au chapitre Section 4.1 [Mise en forme de la page], page 571.
- L'insertion de dialogues et d'indications de mise en scène peuvent se réaliser à l'aide de *markups*, en suivant les directives fournies aux chapitres Section 3.2.4 [Notes de bas de page], page 529, et Section 1.8 [Texte], page 253. Les indications de mise en scène peuvent s'intercaler entre deux blocs `\score` selon les préceptes de la rubrique [Texte indépendant], page 262.

Voir aussi

Glossaire musicologique : Section "Partition à la française" dans *Glossaire*, Section "Frenched staves" dans *Glossaire*, Section "instrument transpositeur" dans *Glossaire*.

Manuel de notation : Section 1.8.1 [Ajout de texte], page 254, Section 1.6.3 [Écriture de parties séparées], page 217, [Imbrication de regroupements de portées], page 202, [Instruments transpositeurs], page 28, [Masquage de portées], page 212, Section 4.1 [Mise en forme de la page], page 571, Section 3.2.4 [Notes de bas de page], page 529, [Regroupement de portées], page 199, [Séparation des systèmes], page 204, [Transposition], page 11.

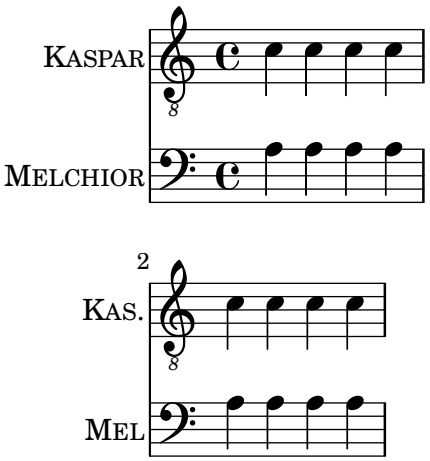
Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

Indication du rôle

Lorsqu'un rôle est distribué sur une portée spécifique, vous pouvez l'indiquer en regard de cette portée :
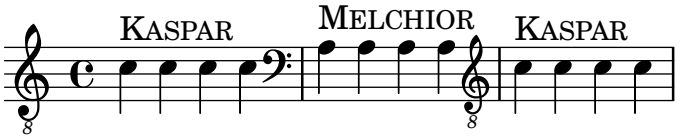
```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative {
        \clef "G_8"
        c'4 c c c
        \break
        c4 c c c
      }
    }
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Melchior
      \set Staff.shortVocalName = \markup \smallCaps Mel
      \clef "bass"
      \relative {
        a4 a a a
        a4 a a a
      }
    }
  >>
```

```
}
```



Lorsque la même portée sert à plusieurs personnages, leur nom est généralement imprimé en surplomb de la portée, à chaque changement de rôle. L'utilisation d'un *markup* – dans une fonte réservée à cet effet – vous permettra de générer ces indications :

```
\relative c' {
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
  \clef "bass"
  a4^\markup \fontsize #1 \smallCaps Melchior
  a a a
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
}
```
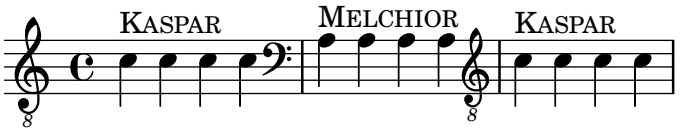


Dans le cas où les changements de personnage se multiplient, il peut s'avérer pratique d'affecter à des variables la définition de chacun des rôles afin de simplifier la gestion des différentes interventions de l'un ou de l'autre.

```
kaspar = {
  \clef "G_8"
  \set Staff.shortVocalName = "Kas."
  \set Staff.midiInstrument = "voice oohs"
  <>^\markup \smallCaps "Kaspar"
}

melchior = {
  \clef "bass"
  \set Staff.shortVocalName = "Mel."
  \set Staff.midiInstrument = "choir aahs"
  <>^\markup \smallCaps "Melchior"
}
```

```
\relative c' {
  \kaspar
  c4 c c c
  \melchior
  a4 a a a
  \kaspar
  c4 c c c
}
```



Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section A.11 [Commandes pour *markup*], page 746, Section 1.8 [Texte], page 253.

Citation-repère

Les citations d'instruments insérées dans les parties vocales, les partitions de chœur ou les partitions d'un pupitre permettent d'indiquer ce qui se passe ailleurs juste avant une entrée. On les retrouve souvent dans la réduction pour piano, ce qui fournit au chef de chœur de précieuses indications sur qui joue quoi, lorsqu'il ne dispose pas d'un conducteur en bonne et due forme.

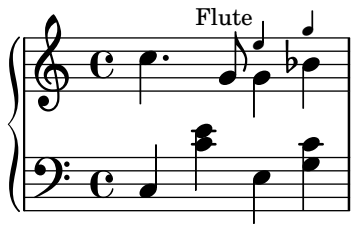
Les mécanismes de base permettant d'insérer des citations sont expliqués en détail aux rubriques [Citation d'autres voix], page 221, et [Mise en forme d'une citation], page 224. Dans le cas où les citations concernent différents instruments, faire mention de celui qui intervient devient une nécessité ; voici une illustration de la manière de procéder en pareil cas :

```
flute = \relative {
  s4 s4 e'' g
}
\addQuote "flute" { \flute }

pianoRH = \relative {
  c''4. g8
  % position name of cue-ing instrument just before the cue notes,
  % and above the staff
  <>^\markup { \right-align { \tiny "Flute" } }
  \cueDuring "flute" #UP { g4 bes4 }
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
      \pianoRH
    }
    \new Staff {
      \clef "bass"
```

```
      \pianoLH
    }
  >>
}
```

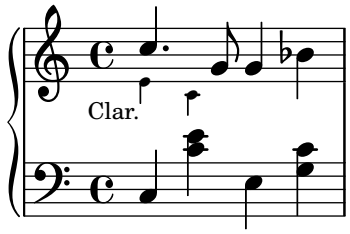


La citation peut concerner un instrument transpositeur, auquel cas il faudra mentionner sa tonalité dans sa définition, afin que ses hauteurs soient automatiquement converties dans la réplique. Ceci est illustré par l'exemple ci-dessous, dans lequel il est fait appel à une clarinette en si bémol. Dans la mesure où les notes citées se trouvent vers le bas de la portée, nous affectons un `DOWN` à la commande `\cueDuring`, de telle sorte que les hampes aillent vers le bas et que le nom de l'instrument cité soit en dessous de la portée.

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % position name of cue-ing instrument below the staff
  <>_\markup { \right-align { \tiny "Clar." } }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Il est clair, au vu de ces deux exemples, que multiplier le nombre de citations dans une partition vocale demande un travail fastidieux et que relire la partie de piano deviendra vite un cauchemar. Vous pouvez néanmoins, comme l'illustre l'exemple suivant, définir une fonction musicale dans le but de vous épargner de la saisie tout en améliorant la lisibilité des notes du piano.

Morceaux choisis

Indication de l'instrument cité dans l'accompagnement d'une partition pour chœur

Lorsque le nombre d'instruments cités dans la réduction pour piano se multiplie, vous pourriez avoir intérêt à créer votre propre fonction pour gérer ces repères. La fonction musicale `\cueWhile` prend quatre arguments : la musique d'où provient la citation, telle que définie par `\addQuote`, le nom qui sera mentionné en regard de cette citation, son positionnement – `UP` ou `DOWN` selon qu'il sera attribué à `\voiceOne` et placé au-dessus ou `\voiceTwo` et placé en dessous – et enfin la musique du piano qui interviendra en parallèle. Le nom de l'instrument en question viendra s'aligner sur la gauche de la citation. Bien que vous puissiez effectuer plusieurs citations, elle ne peuvent être simultanées.

```
cueWhile =
#(define-music-function
   (instrument name dir music)
   (string? string? ly:dir? ly:music?)
   #{
     \cueDuring $instrument #dir {
       \once \override TextScript.self-alignment-X = #RIGHT
       \once \override TextScript.direction = $dir
       <>-\markup { \tiny #name }
       $music
     }
   #})

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }
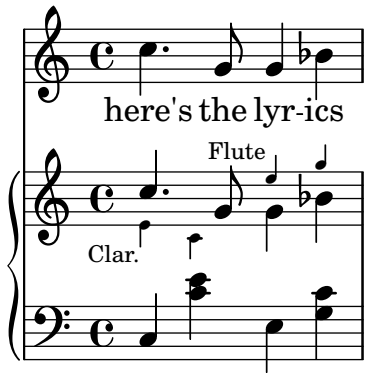
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
```

```
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Voir aussi

Glossaire musicologique : Section “cue-notes” dans *Glossaire*.

Manuel de notation : Section 5.5.1 [Alignement des objets], page 684, [Citation d’autres voix], page 221, Section 5.4.2 [Direction et positionnement], page 667, [Mise en forme d’une citation], page 224, Section 5.6 [Utilisation de fonctions musicales], page 698.

Morceaux choisis : Section “Musique vocale” dans *Morceaux choisis*.

Référence des propriétés internes : Section “CueVoice” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

`\cueDuring` crée automatiquement un contexte `CueVoice` qui accueillera toutes les notes répliquées. Il est par conséquent impossible de faire se superposer des citations à l'aide de la technique simplifiée telle que nous venons de le voir. La superposition de fragments cités requiert que les contextes `CueVoice` soient explicitement déclarés, ainsi que l'utilisation de la commande `\quoteDuring` pour extraire et insérer les notes répliquées.

Musique parlée
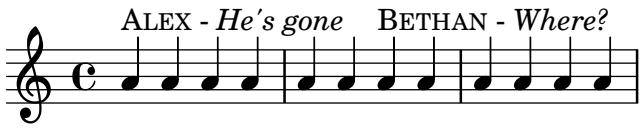
Le *parlato* – ou *Sprechgesang* pour les germanistes – est du texte scandé en rythme, mais sans hauteurs définies ; il est indiqué par des notes en croix, à l'instar des percussions – voir [Têtes de note spécifiques], page 40.

Dialogue et musique

Les dialogues que l'on ajoute à la musique sont traditionnellement imprimés en italique au-dessus des portées, au moment même où ils surviennent.

Une courte intervention peut se formuler à l'aide d'un simple *markup* :

```
\relative {
  a'4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
  a4 a a^\markup { \smallCaps { Bethan - } \italic Where? } a
  a4 a a a
}
```



Une intervention un peu plus longue peut nécessiter d'étirer la musique de telle sorte que le texte ait suffisamment de place. LilyPond ne disposant d'aucun mécanisme permettant d'automatiser l'étirement, vous devrez probablement ajuster vous même la mise en forme.

Dans le cas d'une phrase entière ou de passages relativement denses, le recours à un contexte `Lyrics` peut donner de meilleurs résultats. Le contexte `Lyrics` en question ne doit être rattaché à aucune voix musicale ; chaque fragment de dialogue devra donc comporter des durées explicites. Lorsque les dialogues comportent des pauses, le dernier mot devra être séparé du reste et les durées individualisées pour obtenir un espacement harmonieux de la musique.

Des dialogues qui s'étendent sur plus d'une ligne vous obligeront à insérer des `\break` et ajuster leur placement pour éviter qu'ils ne débordent dans la marge droite. Le dernier mot de la dernière mesure d'une ligne doit être saisi sur une ligne à part.
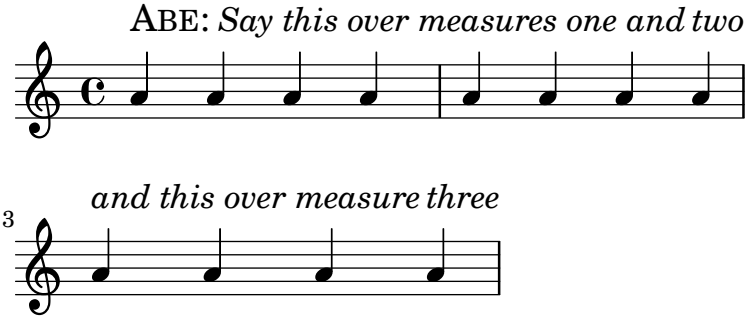
Voici une illustration de tout ce que nous venons de voir :

```
music = \relative {
  \repeat unfold 3 { a'4 a a a }
}

dialogue = \lyricmode {
  \markup {
    \fontsize #1 \upright \smallCaps Abe:
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
```

```
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText.font-shape = #'italic
      \override LyricText.self-alignment-X = #LEFT
    }
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```



Voir aussi

Manuel de notation : [Durée explicite des syllabes], page 294, Section 1.8 [Texte], page 253.

Référence des propriétés internes : Section “LyricText” dans *Référence des propriétés internes*.

2.1.7 Chants liturgiques

Selon les chapelles, la mise en forme des cantiques, psaumes et hymnes répond à des canons bien établis. Bien que différents de par leur présentation, nous verrons dans ce qui suit que les problèmes qui surviennent en matière de typographie se ressemblent, quelle que soit l'obédience.

Références en matière de chant liturgique

La présentation du plain chant et du grégorien selon différents styles est abordée au chapitre Section 2.9 [Notations anciennes], page 468.

Voir aussi

Manuel de notation : Section 2.9 [Notations anciennes], page 468.

Morceaux choisis : Section “Musique vocale” dans *Morceaux choisis*.

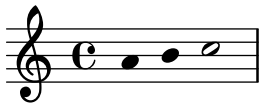
Cantiques et hymnes

La mise en forme contemporaine de cantiques utilise à la fois la notation moderne et un certain nombre d'éléments propres aux notations anciennes. Nous allons examiner quelques uns de ces éléments et la méthode consacrée pour les mettre en œuvre.

Les cantiques utilisent souvent des noires dépourvues de hampe pour indiquer les hauteurs ; le rythme de la mélodie est donné par le rythme et l'accentuation des paroles elles-mêmes.
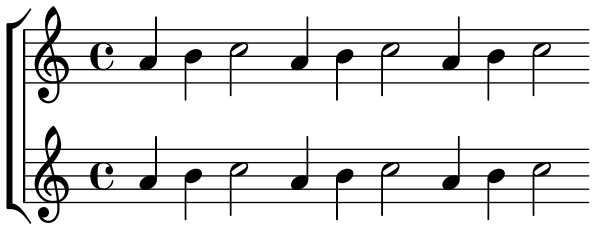
```
stemOff = { \hide Staff.Stem }
```

```
\relative c' {
  \stemOff
  a'4 b c2 |
}
```



Les barres de mesure sont absentes dans la plupart des cas ; celles que vous rencontrerez seront raccourcies ou en pointillé, dans le but d'indiquer une « respiration ». Le fait de supprimer le graveur de barres de mesure produira des portées sans barre :

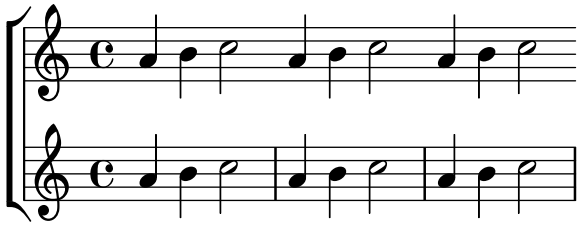
```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```



L'absence de barre de mesure peut ne concerner que certaines portées :
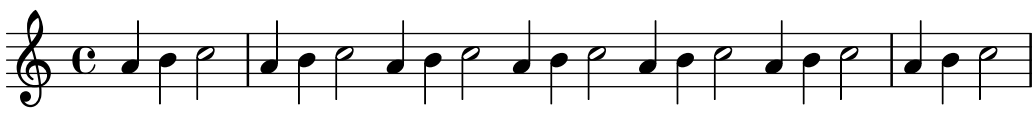
```
\score {
  \new ChoirStaff <<
    \new Staff
    \with { \remove "Bar_engraver" } {
      \relative {
```

```
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
}
```



L'absence de barre de mesure sur un fragment seulement s'obtient en traitant ce fragment comme une cadence. S'il est relativement long, pensez à y insérer des barres fantômes – un simple `\bar ""` – pour indiquer à LilyPond où serait susceptible de se produire un saut de ligne.
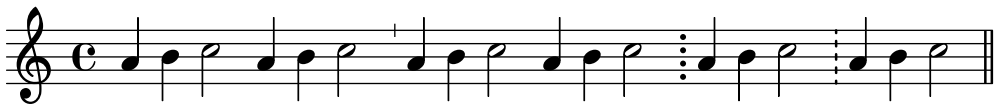
```
\relative a' {
  a4 b c2 |
  \cadenzaOn
  a4 b c2
  a4 b c2
  \bar ""
  a4 b c2
  a4 b c2
  \cadenzaOff
  a4 b c2 |
  a4 b c2 |
}
```



Dans la mélodie d'un cantique, les silences ou pauses s'indiquent à l'aide de barres de mesure spécifiques :

```
\relative a' {
  a4
  \cadenzaOn
  b c2
  a4 b c2
  \bar "'"
  a4 b c2
```

```
  a4 b c2
  \bar ";"
  a4 b c2
  \bar "!"
  a4 b c2
  \bar "||"
}
```
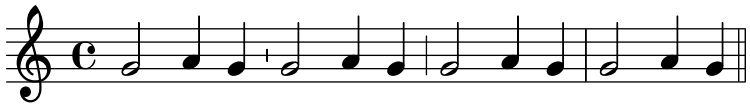


Vous pouvez accessoirement, bien qu'il s'agisse de notation moderne, emprunter au grégorien des indications de pause et silence. Il vous suffit pour cela d'adapter la commande `\breathe` selon vos besoins :

```
divisioMinima = {
  \once \override BreathingSign.stencil =
    #ly:breathing-sign::divisio-minima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
divisioMaior = {
  \once \override BreathingSign.stencil =
    #ly:breathing-sign::divisio-maior
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
divisioMaxima = {
  \once \override BreathingSign.stencil =
    #ly:breathing-sign::divisio-maxima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
finalis = {
  \once \override BreathingSign.stencil =
    #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
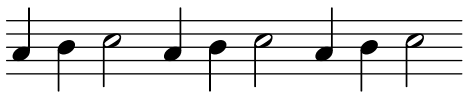
\score {
  \relative {
    g'2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
    g2 a4 g
    \finalis
  }
  \layout {
```

```
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```



De nombreux cantiques sont dépourvus de métrique, voire même de clef.

```
\score {
  \new Staff {
    \relative {
      a'4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
    }
  }
}
```



L'une des traditions anglicanes est de chanter les psaumes sur la base d'un fragment de sept mesures – forme *single* ou simple – ou de deux fragments toujours de sept mesures – forme *double*. Chaque fragment est divisé en deux parties correspondant aux deux moitiés de chaque verset et généralement séparées par une double barre. Il n'est fait usage que de rondes et de blanches, et la première mesure de chaque moitié contient un simple accord de rondes. Il s'agit donc des notes correspondant au « récitatif ». Ces cantiques sont traditionnellement centrés sur la page.

```
SopranoMusic = \relative {
  g'1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative {
  e'1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

TenorMusic = \relative {
```

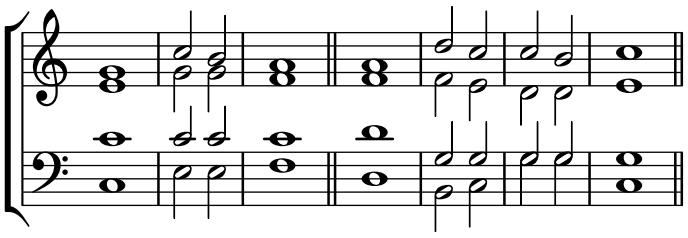
```
  c'1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
}

BassMusic =  \relative {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score {  % centered
      <<
        \new ChoirStaff <<
          \new Staff <<
            \global
            \clef "treble"
            \new Voice = "Soprano" <<
              \voiceOne
              \SopranoMusic
            >>
            \new Voice = "Alto" <<
              \voiceTwo
              \AltoMusic
            >>
          >>
          \new Staff <<
            \clef "bass"
            \global
            \new Voice = "Tenor" <<
              \voiceOne
              \TenorMusic
            >>
            \new Voice = "Bass" <<
              \voiceTwo
              \BassMusic
            >>
          >>
        >>
      >>
      \layout {
        \context {
          \Score
          \override SpacingSpanner.base-shortest-duration =
            #(ly:make-moment 1/2)
        }
        \context {
```

```
          \Staff
          \remove "Time_signature_engraver"
        }
      }
    }  % End score
  }
}  % End markup
```



D'autres approches d'une telle mise en forme font l'objet du premier des exemples qui suivent.

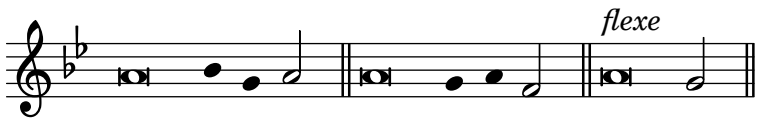
Morceaux choisis

Notation pour psalmodie

Ce style de notation permet d'indiquer la mélodie d'une psalmodie lorsque les strophes sont de longueur inégale.

```
stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \bar "||"
  }
}
```



Cantiques et autres textes liturgiques peuvent être mis en forme avec une grande liberté et parfois emprunter des éléments de notation ancienne. Le texte apparaît souvent sous la mélodie, les mots alors alignés sur les notes. En pareil cas, les notes sont espacées selon les syllabes et non leur durée.

Exemples de notation ancienne – transcription moderne de musique grégorienne
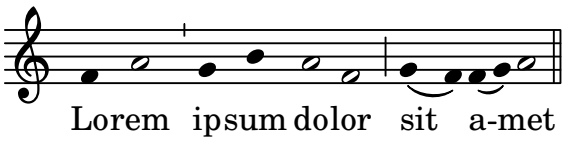
Voici comment vous pourriez transcrire du grégorien. Pour mémoire, il n'y a en grégorien ni découpage en mesure, ni hampe ; seules sont utilisées des têtes de note blanches ou noires, ainsi que des signes spécifiques permettant d'indiquer des silences de différentes durées.

```
\include "gregorian.ly"
```

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



Voir aussi

Manuel d'initiation : Section “Visibilité et couleur des objets” dans *Manuel d'initiation*, Section “Modèles pour ensemble vocal” dans *Manuel d'initiation*.

Manuel de notation : [Barres de mesure], page 102, Section 5.1.4 [Modification des greffons de contexte], page 636, [Musique sans métrique], page 78, Section 2.9 [Notations anciennes], page 468, Section 2.9.4 [Typographie du chant grégorien], page 479, Section 5.4.7 [Visibilité des objets], page 675.

Psalmodie

Les versets d'un psaume anglican sont habituellement centrées sous la mélodie.

Dans le cas d'un chant simple, les sept mesures qui le composent sont répétées pour chaque verset. Dans le cas d'un chant double, les quatorze mesures se répètent par couple de versets. Des marques insérées dans le texte indiquent comment il s'articule par rapport à la mélodie. Chaque verset est séparé en deux, et la rupture est indiquée par un caractère deux points (:) correspondant à la double barre de la mélodie. Le texte précédant les deux points se chante sur les trois premières mesures, celui qui suit sur les quatre dernières mesures.

De simples barres verticales – remplacées par des virgules inversées dans certains psautiers – représentent les barres de mesures portées sur la mélodie. En mode *markup*, ces barres s'obtiennent en saisissant le même caractère | qui sert pour les contrôles de mesure.

```
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing | unto the | Lord : let }
        \line { us heartily rejoice in the | strength of | our }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Vous pourriez tout à fait utiliser d'autres symboles disponibles au travers des glyphes de la fonte `fetaMusic` – voir le chapitre Section 1.8.3 [Fontes], page 279, pour plus de détails.

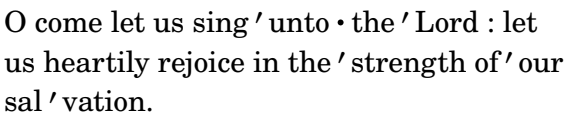
```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O come let us sing \tick unto the \tick Lord : let }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing ' unto the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Lorsqu'une mesure ne comporte qu'une ronde, le texte correspondant à cette mesure est chanté sur cette même note, selon le rythme naturel de la phrase. Lorsque la mesure comporte

deux notes, celles-ci correspondent en général à une ou deux syllabes ; dans le cas contraire, le changement de note est indiqué par un point.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us sing \tick unto \dot the \tick Lord : let
        }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```



Certains psautiers font apparaître, pour indiquer une césure, une astérisque au lieu d'une virgule, ainsi que des caractères gras pour les syllabes accentuées ou allongées.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { Today if ye will hear his voice * }
        \line {
          \concat { \bold hard en }
          | not your | hearts : as in the pro-
        }
        \line { vocation * and as in the \bold day of tempt- | }
        \line { -ation | in the | wilderness. }
      }
    }
  }
}
```

Today if ye will hear his voice *
harden | not your | hearts : as in the pro-
vocation * and as in the **day** of tempt- |
-ation | in the | wilderness.

D'autres psautiers indiquent une syllabe accentuée en la surchargeant d'un accent.

```
tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us \concat {
            si \combine \tick ng
          }
          | unto the | Lord : let
        }
        \line {
          us heartily \concat {
            rejo \combine \tick ice
          }
          in the | strength of | our
        }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing′ | unto the | Lord : let
us heartily rejoice′ in the | strength of | our
sal- | -vation.

L'utilisation du mode *markup* pour centrer le texte et agencer les lignes est abordée en détails au chapitre Section 1.8.2 [Mise en forme du texte], page 263.

La plupart de ces éléments sont regroupés dans l'un des versets du modèle présenté à la rubrique Section "Psalmodie" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*, Section "Psalmodie" dans *Manuel d'initiation*.

Manuel de notation : Section 1.8.3 [Fontes], page 279, Section 1.8.2 [Mise en forme du texte], page 263.

Mesure incomplète et musique liturgique

Il arrive fréquemment que les chants liturgiques comportent des mesures incomplètes aussi bien en début qu'en fin de ligne, de telle sorte qu'à une portée corresponde une ligne de texte. Ceci requiert donc l'utilisation de la commande `\partial` en début de partition et d'une commande `\bar "|"` ou `\bar "||"` à la fin de chaque ligne.

Modèle pour cantique

Le code ci-dessous illustre la manière d'agencer un cantique liturgique dans lequel chaque ligne débute et se termine par une mesure incomplète. Vous noterez par ailleurs l'affichage des paroles indépendamment de la musique.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic =  \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

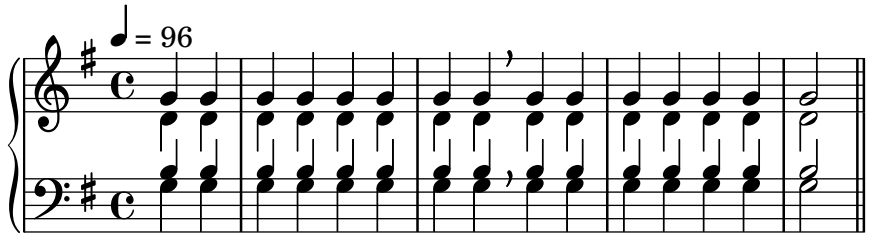
global = {
 \key g \major
}

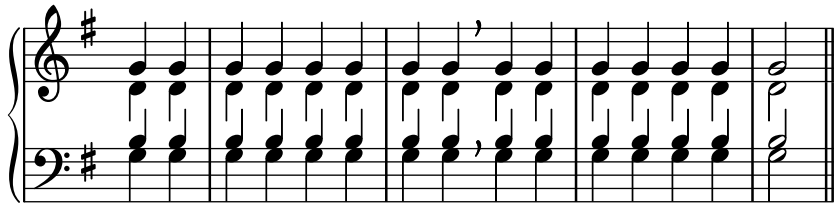
\score {  % Start score
  <<
    \new PianoStaff <<  % Start pianostaff
      \new Staff <<  % Start Staff = RH
        \global
        \clef "treble"
        \new Voice = "Soprano" <<  % Start Voice = "Soprano"
          \Timeline
          \voiceOne
          \SopranoMusic
        >>  % End Voice = "Soprano"
        \new Voice = "Alto" <<  % Start Voice = "Alto"
          \Timeline
          \voiceTwo
          \AltoMusic
```

```
        >>  % End Voice = "Alto"
      >>  % End Staff = RH
      \new Staff <<  % Start Staff = LH
        \global
        \clef "bass"
        \new Voice = "Tenor" <<  % Start Voice = "Tenor"
          \Timeline
          \voiceOne
          \TenorMusic
        >>  % End Voice = "Tenor"
        \new Voice = "Bass" <<  % Start Voice = "Bass"
          \Timeline
          \voiceTwo
          \BassMusic
        >>  % End Voice = "Bass"
      >>  % End Staff = LH
    >>  % End pianostaff
  >>
}  % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
    ""
  }
}

\paper {  % Start paper block
  indent = 0     % don't indent first system
  line-width = 130   % shorten line length to suit music
}  % End paper block
```





This is line one of the first verse
This is line two of the same
And here's line three of the first verse
And the last line of the same

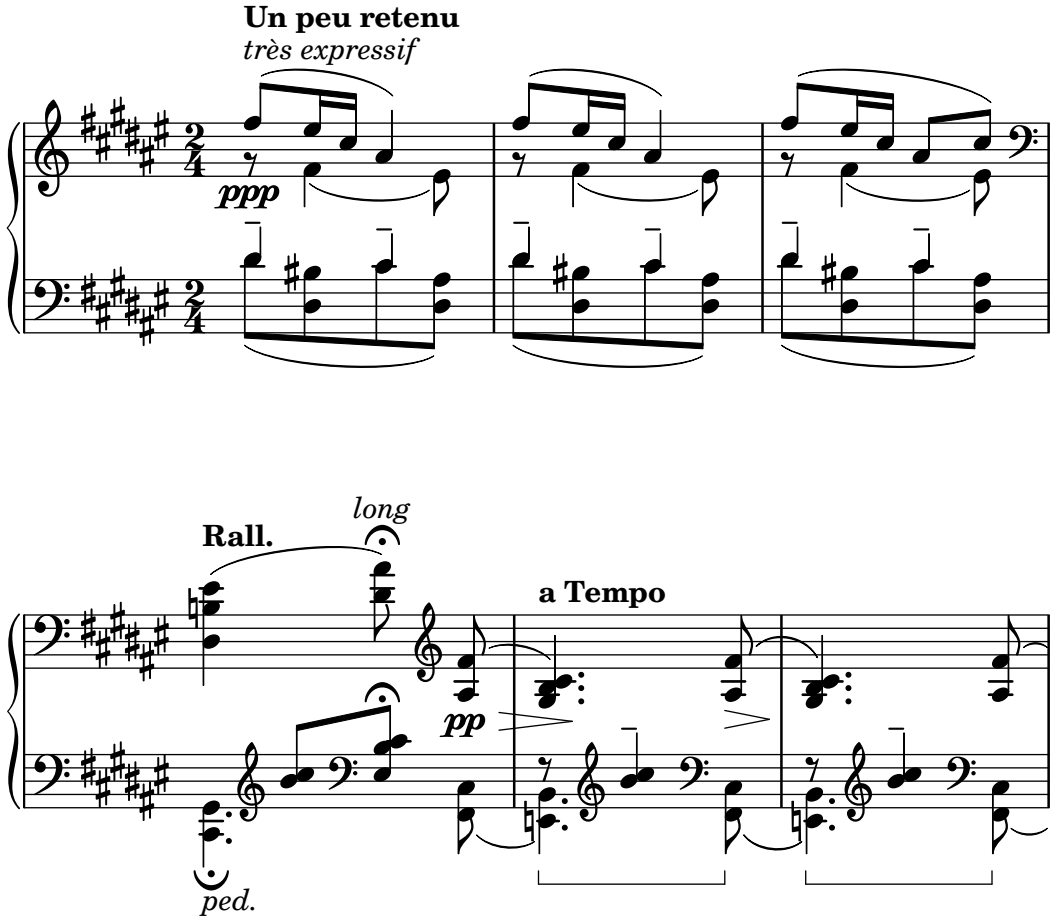
2.1.8 Musique vocale ancienne

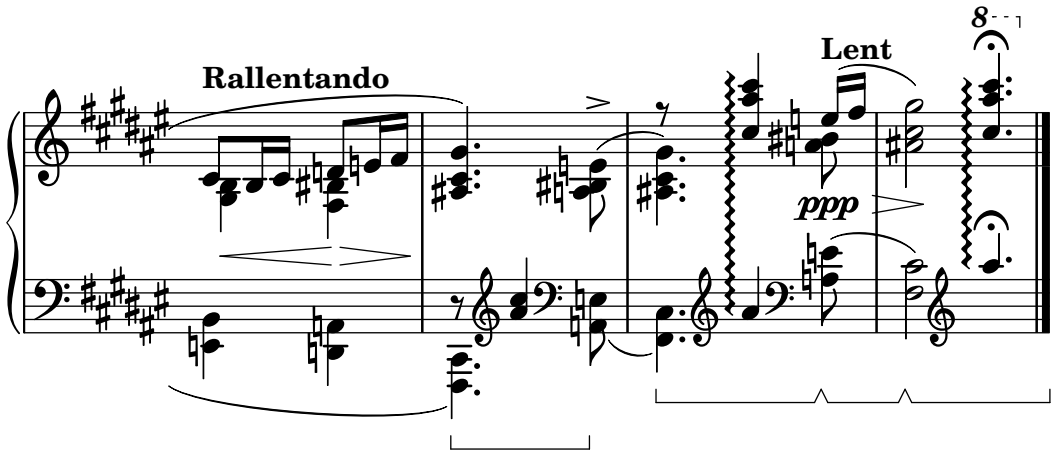
LilyPond prend en charge la musique vocale ancienne. Elle est abordée en détails au chapitre Section 2.9 [Notations anciennes], page 468.

Voir aussi

Manuel de notation : Section 2.9 [Notations anciennes], page 468.

2.2 Instruments utilisant des portées multiples





Ce chapitre traite des différents aspects de la notation que l'on rencontre particulièrement avec les instruments qui ont recours à plusieurs portées, tels que ceux disposant de claviers, la harpe ou le vibraphone. Pour les besoins du discours, et pour simplifier, nous parlerons ici de « clavier » bien que le recours à des portées multiples concerne aussi des instruments qui en sont dépourvus.

2.2.1 Vue d'ensemble des claviers

Nous allons examiner ici les problèmes qui peuvent survenir en matière de notation pour la plupart des instrument à cordes multiples.

Généralités sur les instruments à clavier

La notation pour instrument à clavier est en règle générale présentée sous la forme d'un système pour piano, autrement dit deux portées normales ou plus réunies par une accolade. Cette notation sert également à la harpe ou à d'autres instruments à clés. L'organiste, quant à lui, lira une partition composée de deux portées au sein d'un système pianistique auquel vient s'adjoindre une portée normale pour le pédalier.

Les portées sont largement autonomes, mais il arrive que des voix passent de l'une à l'autre. Le contexte `PianoStaff` est précisément conçu pour gérer la notation spécifique au piano et autres instruments à clavier, notamment ces croisements.

Certaines particularités de la notation pour claviers sont abordées dans d'autres chapitres :

- Les claviers ont régulièrement recours à plusieurs voix dont le nombre peut varier. Voir à ce sujet [Résolution des collisions], page 183.
- On peut écrire la musique pour claviers de façon parallèle, comme l'explique [Saisie de musique en parallèle], page 194.
- Les nuances peuvent se gérer dans un contexte `Dynamics` qu'il suffira d'insérer entre les deux contextes `Staff` pour qu'elles apparaissent sur leur propre ligne centrée entre les deux portées. Voir à ce sujet [Nuances], page 129.
- Les indications de doigté sont abordées dans [Doigtés], page 236.
- Les indications en matière de pédalier d'orgue sont traitées comme des articulations. Voir à ce sujet Section A.14 [Liste des signes d'articulation], page 805.
- Pour générer des repères verticaux, voir [Quadrillage temporel], page 249.
- En plus des *Laissez vibrer*, les liaisons en matière de clavier peuvent intervenir sur des accords arpégés ou des trémolos. Reportez-vous au chapitre [Liaisons de prolongation], page 57.
- Le traitement des arpèges couvrant plusieurs voix ou portées est abordé au chapitre [Arpèges], page 150.
- Une description des indications de trémolo est disponible au chapitre [Répétitions en trémolo], page 170.

- Certaines retouches particulières au monde des claviers sont abordées au chapitre Section "Exemple concret" dans *Manuel d'initiation*.
- Des notes fantômes permettent d'introduire des liaisons de tenue qui passent d'une voix à l'autre, comme le montre Section "Autres utilisations des retouches" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Autres utilisations des retouches" dans *Manuel d'initiation*, Section "Exemple concret" dans *Manuel d'initiation*.

Manuel de notation : [Arpèges], page 150, [Doigtés], page 236, [Liaisons de prolongation], page 57, Section A.14 [Liste des signes d'articulation], page 805, [Noms d'instrument], page 217, [Quadrillage temporel], page 249, [Regroupement de portées], page 199, [Répétitions en trémolo], page 170, [Résolution des collisions], page 183, [Saisie de musique en parallèle], page 194.

Morceaux choisis : Section "Claviers" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PianoStaff" dans *Référence des propriétés internes*.

Changement de portée manuel
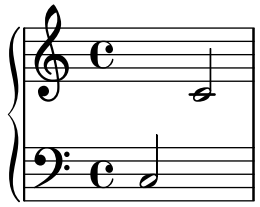
Il est possible de passer d'une portée à l'autre de façon manuelle, au moyen de la commande

```
\change Staff = nomDeLaPortee
```

La valeur *nomDeLaPortee* est le nom de la portée sur laquelle va se déplacer la voix courante. Pour des raisons pratiques, on nomme la portée supérieure `"haut"` et la portée inférieure `"bas"`, donc *nomDeLaPortee* désigne habituellement `"haut"`, `"bas"`, `"MD"` ou `"MG"`.

Dans tous les cas, le contexte de portée ainsi utilisé doit exister au préalable. Le cas échéant, vous devrez « garder actives » certaines portées – voir Section 5.1.3 [Conservation d'un contexte], page 634, à ce propos, ou bien explicitement instanciées – en recourant par exemple à un accord vide `<>` (voir [Notes en accords], page 173).
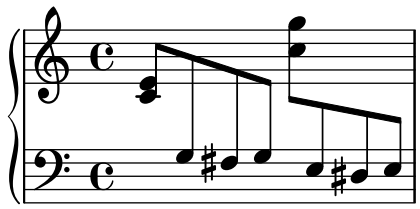
```
\new PianoStaff <<
  \new Staff = "up" {
    % enforce creation of all contexts at this point of time
    <>
    \change Staff = "down" c2
    \change Staff = "up" c'2
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```



Pour ligaturer automatiquement des notes entre deux portées, procédez ainsi :

```
\new PianoStaff <<
```

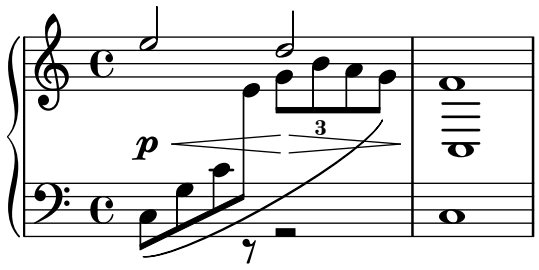
```
  \new Staff = "up" {
    <e' c'>8
    \change Staff = "down"
    g8 fis g
    \change Staff = "up"
    <g'' c''>8
    \change Staff = "down"
    e8 dis e
    \change Staff = "up"
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```



Si les ligatures demandaient à être retouchées, commencez par modifier la direction des hampes. L'emplacement des ligatures sera alors calculé à partir du centre de la portée la plus proche – voir Section "Correction des collisions d'objets" dans *Manuel d'initiation* pour un exemple de retouche sur des ligatures.

Une voix qui change de portée peut entraîner des collisions :

```
\new PianoStaff <<
  \new Staff = "up" {
    \voiceOne
    % Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner.staff-padding = #4
    e''2\p\< d''\>
    c1\!
  }
  \new Staff = "down" <<
    {
      \clef bass
      s4. e,8\rest g,2\rest
      c1
    } \\ {
      c8\( g c'
      \change Staff = "up"
      e' g' b'-3 a' g'\)
      f'1
    }
  >>
>>
```



Hampes et liaisons viennent en surimpression sur la ligne des nuances parce que la résolution automatique des collisions est inactivée pour ce qui relie les notes de différentes portées ainsi que pour les hampes ou extenseurs affectés à des notes incluses dans un changement de portée. Lorsque des collisions surviennent en pareil cas, il vous faudra les résoudre, en suivant les directives du chapitre Section “Correction des collisions d’objets” dans *Manuel d’initiation*.

Voir aussi

Manuel d’initiation : Section “Correction des collisions d’objets” dans *Manuel d’initiation*.

Manuel de notation : [Barres de ligature automatiques], page 87, Section 5.1.3 [Conservation d’un contexte], page 634, [Hampes], page 245.

Morceaux choisis : Section “Claviers” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Beam” dans *Référence des propriétés internes*, Section “ContextChange” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Dans la mesure où l’évitement de collision des ligatures ne fonctionne pas lorsqu’une ligature automatique prend fin juste avant un changement de portée, utilisez alors une ligature manuelle.
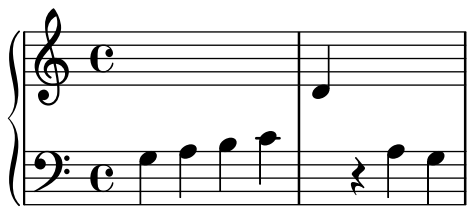
Changement de portée automatique

Les voix peuvent passer automatiquement d’une portée à l’autre, au moyen de la syntaxe suivante :

```
\autoChange ...musique...
```

Deux portées seront alors créées au sein du contexte `PianoStaff`, nommées respectivement `"up"` et `"down"`. La portée du bas sera par défaut en clef de fa. La commande `\autoChange` bascule les notes d’une portée à l’autre en fonction de leur hauteur (le do du milieu servant de pivot), et place les silences en fonction des notes qui les suivront. Ainsi :

```
\new PianoStaff {
  \autoChange {
    g4 a b c'
    d'4 r a g
  }
}
```
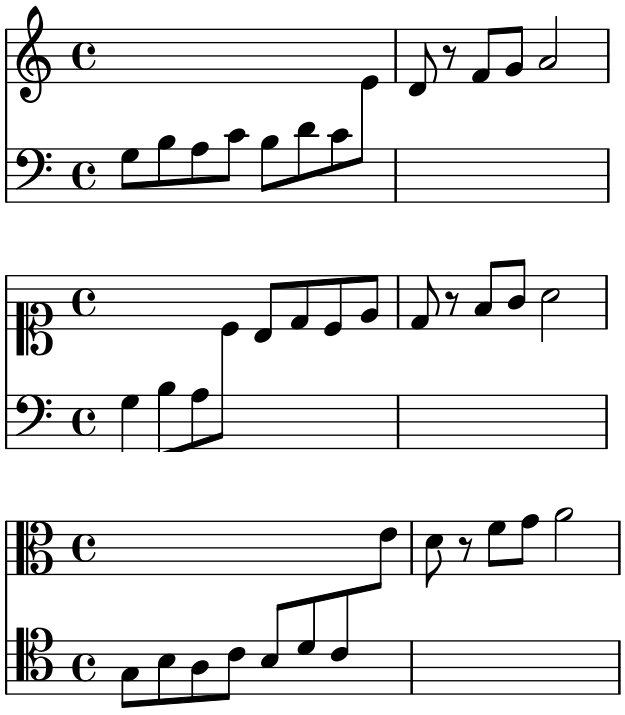


Il est tout à fait possible de déterminer une autre hauteur charnière. Dès lors qu’aucune portée n’a été spécifiquement instanciée, d’autres clefs peuvent être utilisées.

```
music = {
  g8 b a c' b8 d' c'8 e'
```

```
  d'8 r f' g' a'2
}

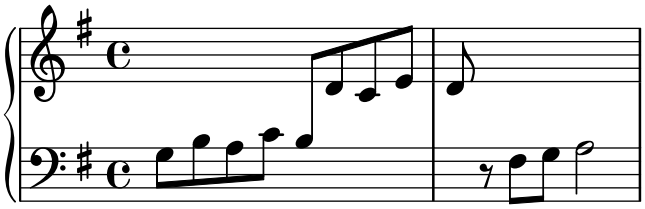
\autoChange d' \music
\autoChange b \with { \clef soprano } \music
\autoChange d' \with { \clef alto } \with { \clef tenor } \music
```



Une section en mode `\relative` se situant en dehors de la commande `\autoChange` n'aura pas d'effet sur les hauteurs de l'expression *musique*. Il est donc préférable d'introduire la directive `\relative` **après** `\autoChange`.

Lorsque des contrôles particuliers doivent s'appliquer aux portées, mieux vaut les nommer explicitement – attention : sous peine d'effet indésirable quant au résultat, la portée supérieure doit s'appeler `"up"` et l'inférieure `"down"` ; *ceci est à notre connaissance le seul cas où ces noms de variable sont figés*. Cette procédure sert, entre autres, à indiquer l'armure sur la portée inférieure :

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melodieUn" {
      \key g \major
      \autoChange \relative {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



Voir aussi

Manuel de notation : [Changement de portée manuel], page 355.

Morceaux choisis : Section “Claviers” dans *Morceaux choisis*.

Référence des propriétés internes : Section “AutoChangeMusic” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les changements de portée automatiques n'interviennent pas toujours à l'endroit le plus opportun. Pour un résultat de meilleure qualité, il vaut mieux indiquer soi-même ces changements.

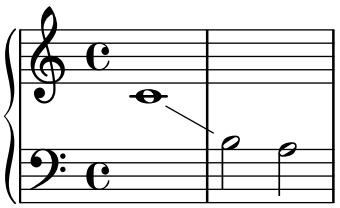
Un accord ne peut se répartir sur plusieurs portées ; sa portée d'affectation sera déterminée par la première hauteur mentionnée dans la construction de cet accord.

`\autoChange` ne peut intervenir à l'intérieur d'une commande `\tuplet`.

Lignes de changement de portée

Lorsqu'une voix change de portée, il est possible d'imprimer automatiquement une ligne reliant les notes, en faisant appel à la commande `\showStaffSwitch` :

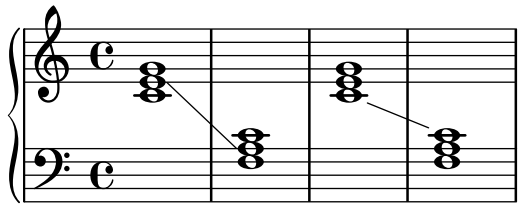
```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c'1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



Dans le cas d'accords, cette ligne connectera la **dernière** hauteur de chacun d'entre eux selon leur ordre d'apparition dans le fichier source ; ceci permet d'ajuster rapidement les positions de départ et d'arrivée de cette ligne.

```
\new PianoStaff <<
  \new Staff = "one" {
    <c' e' g'>1
    \showStaffSwitch
    \change Staff = "two"
    <a c' f>1
    \hideStaffSwitch
```

```
    \change Staff = "one"
    <e' g' c'>1
    \showStaffSwitch
    \change Staff = "two"
    <f a c'>1
  }
  \new Staff = "two" {
    \clef bass
    s1*4
  }
>>
```



Commandes prédéfinies

`\showStaffSwitch`, `\hideStaffSwitch`.

Voir aussi

Morceaux choisis : Section “Claviers” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Note_head_line_engraver” dans *Référence des propriétés internes*, Section “VoiceFollower” dans *Référence des propriétés internes*.

Morceaux choisis

Hampes interportées
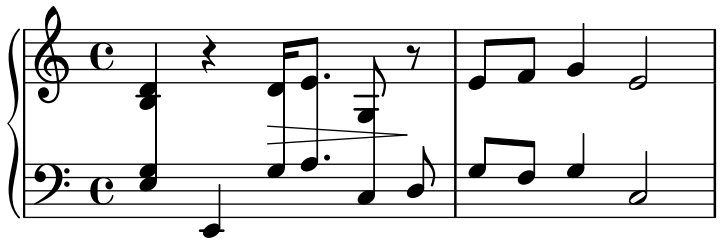
L’exemple ci-dessous illustre l’utilisation du `Span_stem_engraver` et de la commande `\crossStaff` afin de connecter des hampes entre les portées.

Nul n’est besoin de spécifier la taille des hampes ; le graveur calcule automatiquement la distance relative des têtes de note avec les portées.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
    \new Staff {
      <b d'>4 r d'16\> e'8. g8 r\!
      e'8 f' g'4 e'2
    }
    \new Staff {
      \clef bass
      \voiceOne
      \autoBeamOff
      \crossStaff { <e g>4 e, g16 a8. c8} d
```

```
      \autoBeamOn
      g8 f g4 c2
    }
  >>
}
```



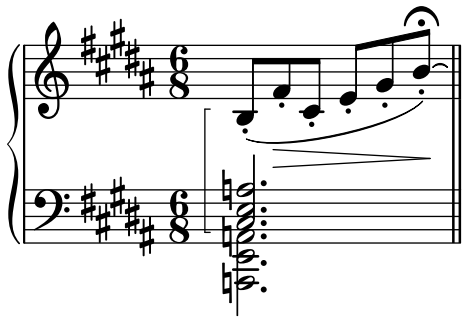
Il n'est pas possible, à l'heure actuelle et en raison de la manière dont il a été implémenté, de spécifier ce graveur en le mettant entre guillements ; il faut le mentionner en faisant précéder son nom d'un symbole *hash* (un `#`).

Indication d'un accord à cheval sur deux portées par un crochet d'arpège

Un crochet d'arpège peut indiquer que des notes réparties sur deux portées différentes doivent être jouées par la même main. Le contexte `PianoStaff` doit accepter ces arpèges « distribués », et les indications d'arpège du contexte `PianoStaff` adopter une allure de crochet.

(Debussy, Les collines d'Anacapri, mesure 65)

```
\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio.stencil =
    #ly:arpeggio::brew-chord-bracket
  \new Staff {
    \relative c' {
      \key b \major
      \time 6/8
      b8-.(\arpeggio fis'-.\> cis-.
        e-. gis-. b-.)\!\fermata^\laissezVibrer \bar "||"
    }
  }
  \new Staff {
    \relative c' {
      \clef bass
      \key b \major
      <<
        {
          <a e cis>2.\arpeggio
        }
        \\
        {
          <a, e a,>2.
        }
      >>
    }
  }
>>
```



Voir aussi

Morceaux choisis : Section “Claviers” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Stem” dans *Référence des propriétés internes*.
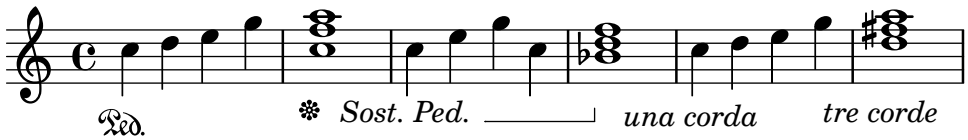
2.2.2 Piano

Ce chapitre traite des aspects de la notation directement liés au piano.

Pédales de piano

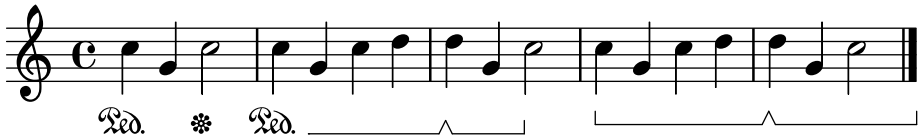
Le piano possède deux pédales, parfois trois, permettant de modifier l’émission du son : une pédale de *tenue* (*sustain*), une pédale de *sourdine* (*una corda* ou *U.C.*) et une pédale *tonale* (*sostenuto* ou *sos.*). La pédale *sustain* se rencontre aussi sur les vibraphones et celestas.

```
\relative {
  c''4\sustainOn d e g
  <c, f a>1\sustainOff
  c4\sostenutoOn e g c,
  <bes d f>1\sostenutoOff
  c4\unaCorda d e g
  <d fis a>1\treCorde
}
```



Trois styles sont à votre disposition pour imprimer les indications de pédale : sous forme de texte, de crochet, ou une combinaison des deux. `text` est le style de notation par défaut pour les pédales de tenue ou de sourdine – le traditionnel « *Ped. ». La pédale tonale, en revanche, utilise `mixed` par défaut.

```
\relative {
  c''4\sustainOn g c2\sustainOff
  \set Staff.pedalSustainStyle = #'mixed
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2\sustainOff
  \set Staff.pedalSustainStyle = #'bracket
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2
  \bar "|."
}
```



Le placement des commandes de pédale correspond au mouvement de la pédale de tenue pendant le jeu. Garder une pédale enclenchée jusqu'à la barre finale s'obtient tout simplement en omettant la commande de relâcher.

Les indications de pédale peuvent s'inscrire dans un contexte `Dynamics`, ce qui aura pour effet de leur attribuer une ligne en propre.

Voir aussi

Manuel de notation : [Liaisons de prolongation], page 57.

Morceaux choisis : Section "Claviers" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Piano_pedal_engraver" dans *Référence des propriétés internes*, Section "PianoPedalBracket" dans *Référence des propriétés internes*, Section "SustainEvent" dans *Référence des propriétés internes*, Section "SostenutoPedal" dans *Référence des propriétés internes*, Section "SustainPedal" dans *Référence des propriétés internes*, Section "SustainPedalLineSpanner" dans *Référence des propriétés internes*, Section "SostenutoEvent" dans *Référence des propriétés internes*, Section "SostenutoPedalLineSpanner" dans *Référence des propriétés internes*, Section "UnaCordaPedal" dans *Référence des propriétés internes*, Section "UnaCordaEvent" dans *Référence des propriétés internes*, Section "UnaCordaPedalLineSpanner" dans *Référence des propriétés internes*.

2.2.3 Accordéon

Symboles de jeux

De nombreux accordéons possèdent plusieurs jeux d'anches capables de jouer à l'unisson ou bien à l'octave inférieure ou supérieure par rapport aux notes écrites. Chaque facteur d'accordéon donne des noms différents aux *tirettes* (*shifts*) qui permettent de sélectionner les différents jeux d'anches tels que *hautbois*, *musette* ou *bandonéon*, de telle sorte qu'un système de symbole a fini par voir le jour afin de faciliter les instructions pour exécuter un morceau.

Une liste des différents symboles est disponible à l'annexe Section "Registre pour arccordéon" dans *Manuel de notation*.

Morceaux choisis

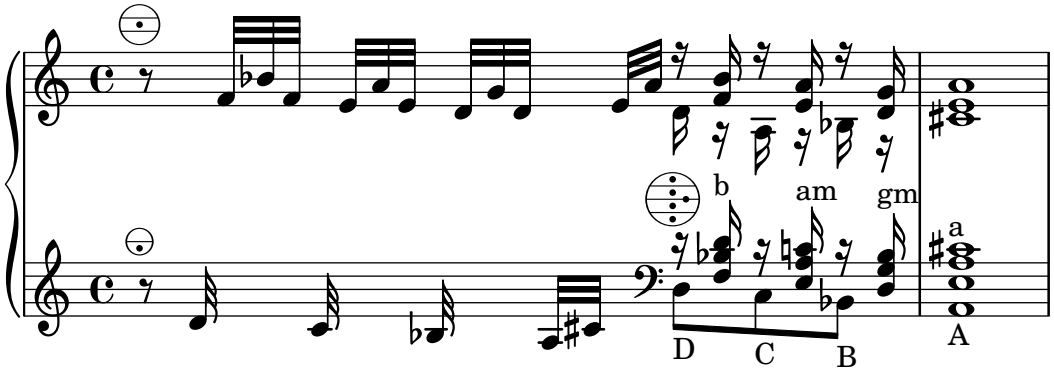
Symboles de registre pour accordéon

Les symboles spécifiques aux registres d'accordéon sont disponible tant sous forme de `\markup` que d'événements musicaux indépendants – un changement de registre a tendance à intervenir entre des événements musicaux réels. Les registres de basse ne sont pas très standardisés. Les différentes commandes disponibles sont regroupées à l'annexe Section "Registres d'accordéon" dans *Manuel de notation*.

```
      \\
      { d r a r bes r }
    >> |
    <cis e a>1
}

\new Staff \relative {
  \clef treble
  \freeBass "1"
  r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
  \clef bass \stdBass "Master"
  <<
    { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
```

```
      <e a cis>1^"a" }
      \\
      { d8_"D" c_"C" bes_"B" | a1_"A" }
    >>
  }
>>
```



Voir aussi

Morceaux choisis : Section “Keyboards” dans *snippets*.

2.2.4 Harpe

Cette partie s'intéresse aux particularités en matière de notation pour la harpe.

Généralités sur la harpe

Certaines caractéristiques de la musique pour harpes sont abordées dans d'autres chapitres, tels que :

- Les glissandos, l'une des techniques spécifique à la harpe, [Glissando], page 146.
- Le *bisbigliando*, qui s'écrit comme un trémolo, [Répétitions en trémolo], page 170.
- Les harmoniques naturelles sont présentées dans [Harmoniques], page 367.
- L'interprétation des arpeggios est abordée dans [Arpèges], page 150.
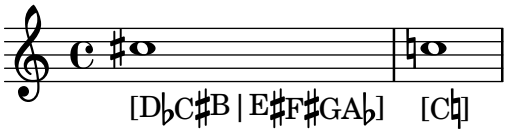
Voir aussi

Manuel de notation : [Arpèges], page 150, [Glissando], page 146, [Harmoniques], page 367, [Répétitions en trémolo], page 170.

Pédales de harpe

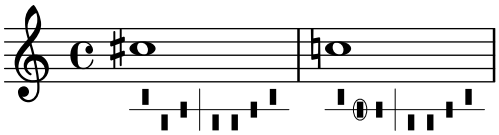
Les harpes comportent sept cordes par octave qui peuvent sonner naturel, dièse ou bémol. Si chacune des cordes de la harpe celtique (*lever harp*) est accordée individuellement, celles d'une harpe à pédalier ayant la même note de base sont contrôlées par une seule pédale. De gauche à droite, elles correspondent aux notes ré, do, si, et mi, fa, sol, la pour la grande harpe. Les trois premières pédales sont réservées au pied gauche, les quatre dernières au pied droit. Leur position peut être indiquée par une marque textuelle :

```
\textLengthOn
cis''1_\markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c''!1_\markup \concat \vcenter {
  [ C \natural ]}
```



ou bien sous forme de diagramme :

```
\textLengthOn
cis''1_\markup { \harp-pedal "^v-|vv-^" }
c''!1_\markup { \harp-pedal "^o--|vv-^" }
```

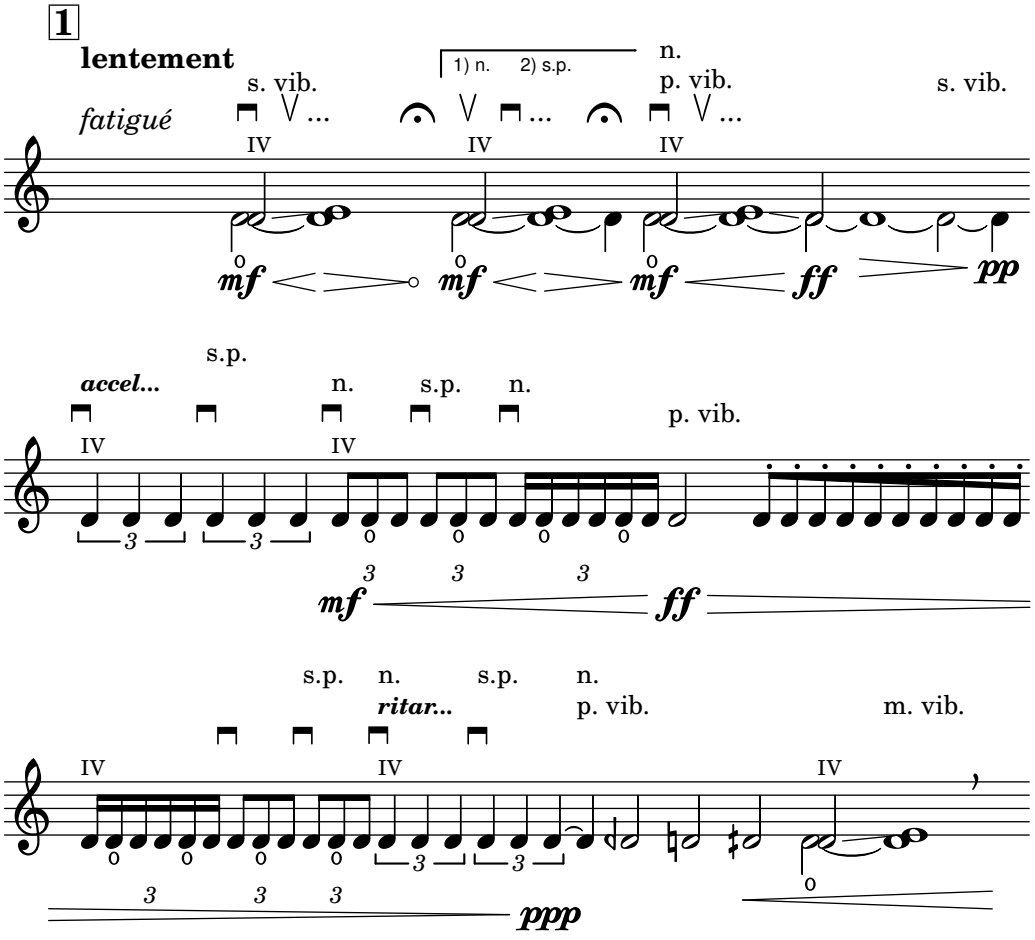


Bémol si la pédale est relâchée (ou en haut), bécarre si elle est bloquée sur le cran du milieu, et dièse si elle est tout à fait enfoncée. La commande `\harp-pedal` prend en argument une chaîne de caractères, où `^` indique que la pédale est relâchée ou en haut (bémol), `-` qu'elle est bloquée sur le cran du milieu (bécarre), `v` qu'elle est tout à fait enfoncée (dièse) et `|` représente le séparateur (entre gauche et droite de l'instrumentiste). Faire précéder un symbole par un `o` permet de l'inscrire dans un cercle.

Voir aussi

Manuel de notation : [Commentaires textuels], page 256, Section "Markups spécifiques aux instruments (en anglais)" dans *Manuel de notation*.

2.3 Cordes non frettées



Cette section dispense des informations supplémentaires et utiles à l'écriture pour les cordes non frettées, et tout spécialement pour les cordes d'orchestre.

2.3.1 Vue d'ensemble de la notation pour cordes non frettées

Il existe peu d'éléments spécifiques à l'écriture pour instruments à cordes non frettées. La musique est notée sur une portée et ne nécessite généralement qu'une seule voix. Le recours à deux voix distinctes peut cependant permettre de traiter efficacement des doubles notes ou des divisions.

Références en matière de cordes non frettées

La majorité des éléments de notation nécessaires en matière de cordes d'orchestre et autres instruments à archet est abordée dans d'autres chapitres de la documentation :

- Les indications textuelles telles que « pizz. » ou « arco » sont ajoutées comme du texte simple – voir à ce sujet [Commentaires textuels], page 256.
- Les indications de doigtés, incluant les indications du pouce, sont décrites dans [Doigtés], page 236.
- Des numéros de corde peuvent être ajoutés (généralement en chiffre romain pour les instruments à archet) – voir à ce sujet [Indications du numéro de corde], page 370.
- Les doubles notes sont généralement indiquées en écrivant un accord, voir [Notes en accords], page 173. Les précisions pour jouer des accords peuvent être ajoutées, comme l'indique [Arpèges], page 150.
- Un squelette de partition est disponible à l'annexe Section "Modèles pour quatuor à cordes" dans *Manuel d'initiation*. D'autres informations se trouvent dans les exemples de code.

Voir aussi

Manuel d'initiation : Section "Modèles pour quatuor à cordes" dans *Manuel d'initiation*.

Manuel de notation : [Arpèges], page 150, [Commentaires textuels], page 256, [Doigtés], page 236, [Notes en accords], page 173.
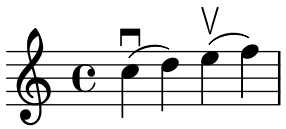
Morceaux choisis : Section "Cordes non frettées" dans *Morceaux choisis*.

Indications d'archet

Les indications d'archet se créent comme des articulations, elles sont décrites dans [Articulations et ornements], page 127.

Les indications d'archet, poussé (`\upbow`) et tiré (`\downbow`), peuvent se combiner à des liaisons comme ici :

```
\relative { c''4(\downbow d) e(\upbow f) }
```
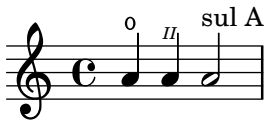


Des chiffres romains peuvent s'ajouter pour indiquer les numéros de corde (en lieu et place de chiffres arabes cerclés), comme expliqué dans [Indications du numéro de corde], page 370.

Alternativement, les indications de corde peuvent se traiter sous forme de *markup*, et un script indiquer une corde à vide.

```
a'4 \open
\romanStringNumbers
a'\2
```

```
a'2^\markup { \small "sul A" }
```



Commandes prédéfinies

`\downbow`, `\upbow`, `\open`, `\romanStringNumbers`.

Voir aussi
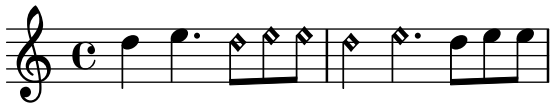
Manuel de notation : [Articulations et ornements], page 127, [Indications du numéro de corde], page 370, [Liaisons d'articulation], page 139.

Harmoniques

Harmoniques naturels
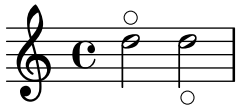
Les harmoniques naturels sont indiqués de différentes manières : une tête de note en forme de losange signifie généralement que vous devez effleurer la corde à l'endroit où vous l'auriez pincée si la note avait été normale.

```
\relative d'' {
  d4 e4.
  \harmonicsOn
  d8 e e
  d4 e4.
  \harmonicsOff
  d8 e e
}
```



Une autre façon de procéder consiste à faire surmonter la note normale d'un petit cercle. Ceci indique que la note écrite doit être jouée en harmonique :

```
d''2^\flageolet d''_\flageolet
```
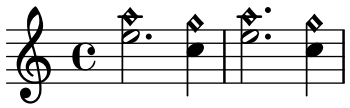


Un plus petit cercle peut être créé, comme indiqué dans les exemples de code contenus dans [Références en matière de cordes non frettées], page 366.

Harmoniques artificiels

Les harmoniques artificiels sont indiqués par une double tête de note : l'une normale, indique la note à pincer, et l'autre, en forme de losange, indique l'endroit où la corde doit être effleurée.

La propriété `harmonicDots`, lorsqu'elle est activée, permet d'ajouter un point aux notes pointées affublées d'un `\harmonic`.

```
<e a\harmonic>2.  <c g'\harmonic>4
\set harmonicDots = ##t
<e a\harmonic>2.  <c g'\harmonic>4
```



Voir aussi

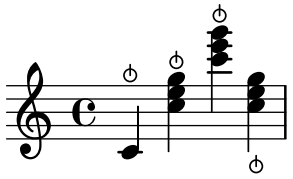
Glossaire musicologique : Section “harmonics” dans *Glossaire*.

Manuel de notation : [Références en matière de cordes non frettées], page 366, [Têtes de note spécifiques], page 40.
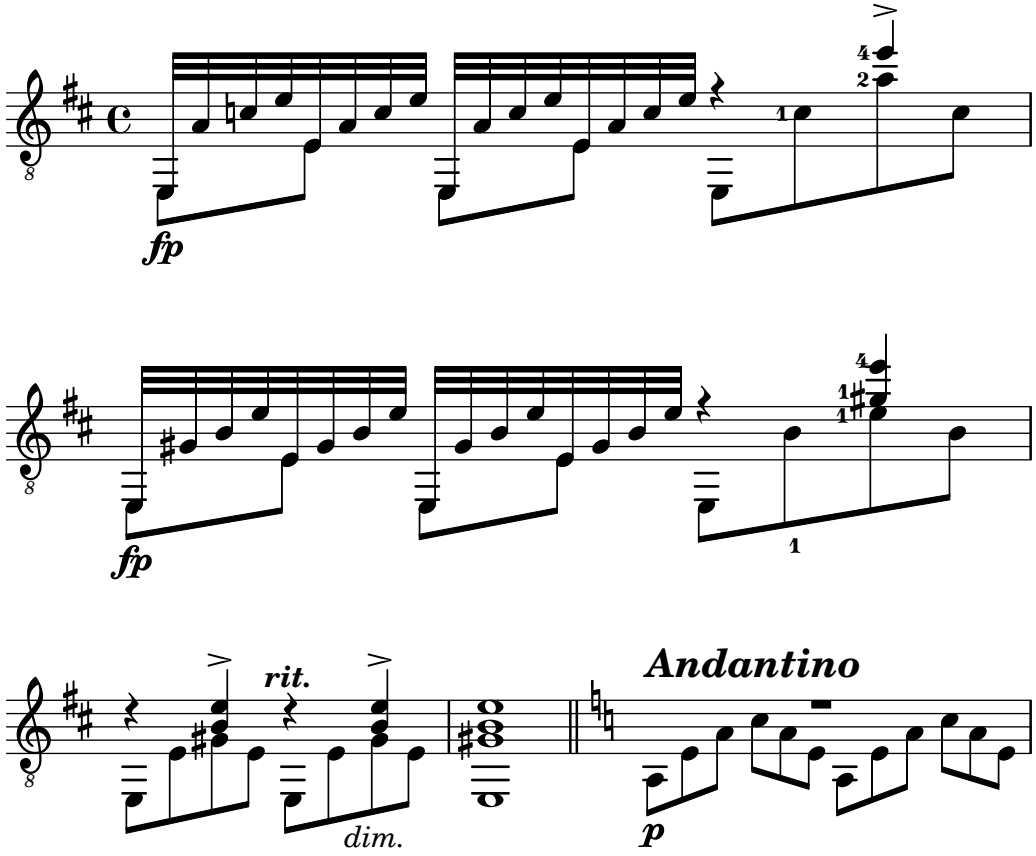
Snap (Bartók) pizzicato
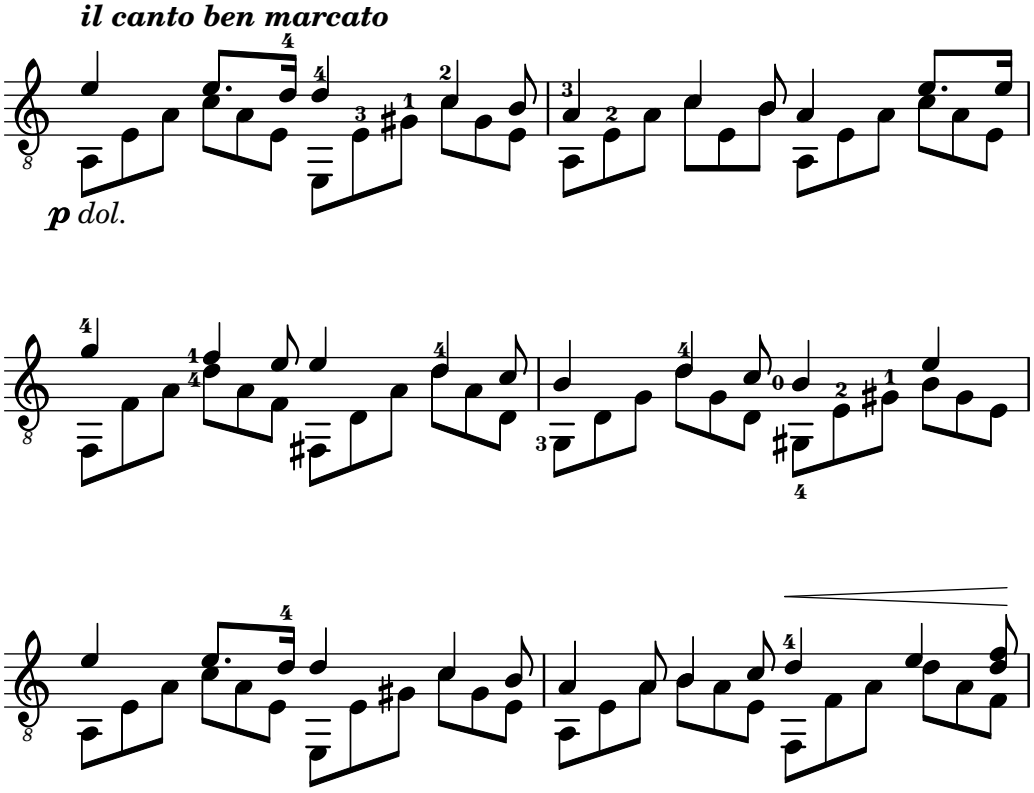
Un *snap pizzicato*, aussi appelé « Bartok pizz » est un type de pizzicato pour lequel la corde est tirée vers le haut (plutôt que sur le côté) de telle sorte qu’elle vienne frapper le manche.

```
\relative {
  c'4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



2.4 Instruments à cordes frettées





Cette section traite de différents aspects de la notation propre aux instruments à cordes frettées.

2.4.1 Vue d'ensemble des cordes frettées

Nous allons aborder, dans les paragraphes qui suivent, les particularités communes aux différents instruments à cordes frettées.

Références en matière de cordes frettées

La musique pour instruments à cordes frettées est généralement notée sur une seule portée, en notation traditionnelle ou en tablature, les deux étant parfois combinées. Il est aussi courant en musique populaire d'utiliser des diagrammes d'accord au-dessus de la portée traditionnelle. La guitare et le banjo sont des instruments transpositeurs, sonnant une octave au-dessous de ce qui est écrit. Les partitions pour ces instruments utilisent donc la clé de sol ottava bassa `"treble_8"` – ou une instruction `\transposition c` pour un rendu MIDI correct. Vous pourrez trouver ailleurs dans la documentation d'autres éléments aussi utilisés par les instruments à cordes frettées :

- Les doigtés s'obtiennent comme indiqué au chapitre [Doigtés], page 236.
- En plus des *Laissez vibrer*, les liaisons peuvent intervenir sur des accords arpégés ou des trémolos. Reportez-vous au chapitre [Liaisons de prolongation], page 57.
- Des indications quant au support polyphonique se trouvent au chapitre [Résolution des collisions], page 183.
- La notation des sons harmoniques se trouve à la section [Harmoniques], page 367.

Voir aussi

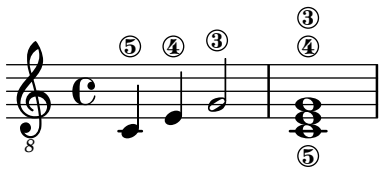
Manuel de notation : [Arpèges], page 150, [Clefs], page 18, [Doigtés], page 236, [Liaisons de prolongation], page 57, Section A.14 [Liste des signes d'articulation], page 805, [Noms d'instrument], page 217, [Résolution des collisions], page 183, [Saisie de musique en parallèle], page 194.

Indications du numéro de corde

Une corde sur laquelle une note doit être jouée peut être indiquée en attachant `\numéro` à cette note prise dans une construction de type accord `<>`.

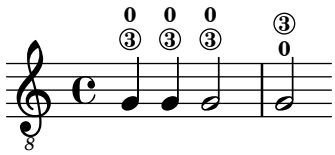
> **Note :** Les numéros de corde **doivent** être définis dans une construction de type accord même s'il n'y a qu'une seule note.

```
\clef "treble_8"
c4\5 e\4 g2\3
<c\5 e\4 g\3>1
```



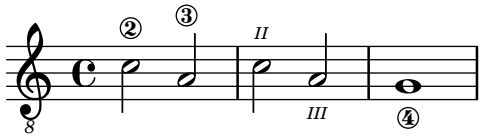
Quand les indications de doigté et de numéro de corde sont attachées à une même note, leur positionnement se fera en fonction de l'ordre dans lequel elles apparaissent dans le code et **uniquement** si elles interviennent au sein d'une construction d'accord : le positionnement des doigtés est géré différemment selon qu'ils s'appliquent à l'accord entier ou à des notes isolées indépendantes d'un accord.

```
\clef "treble_8"
g4\3-0
g-0\3
<g\3-0>2
<g-0\3>
```



Les numéros de corde peuvent aussi, comme traditionnellement pour les cordes non frettées, s'imprimer en chiffres romains placés sous la portée plutôt qu'en surplomb.

```
\clef "treble_8"
c'2\2
a\3
\romanStringNumbers
c'\2
\set stringNumberOrientations = #'(down)
a\3
\arabicStringNumbers
g1\4
```



Morceaux choisis

La plupart des comportements en matière d'indications de numéro de corde, à savoir l'objet `StringNumber`, et y compris leur positionnement, est comparable au traitement des doigtés – voir [Doigtés], page 236.

Commandes prédéfinies

`\arabicStringNumbers`, `\romanStringNumbers`.

Voir aussi

Manuel de notation : [Doigtés], page 236.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

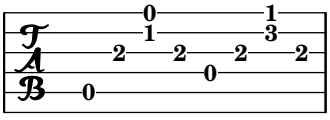
Référence des propriétés internes : Section "StringNumber" dans *Référence des propriétés internes*, Section "Fingering" dans *Référence des propriétés internes*.

Tablatures par défaut

La notation en tablature est utilisée pour certains instruments à cordes pincées. Les hauteurs n'y sont pas indiquées par des têtes de note mais par des chiffres ou autres symboles qui indiquent sur quelle corde et à quelle case chaque note doit être jouée. Des notes devant être jouées simultanément seront alors alignées verticalement.

Par défaut, la première corde est la plus aiguë et correspond à la ligne supérieure du `TabStaff`. Les cordes suivent par défaut l'accordage traditionnel d'une guitare (à six cordes). Les notes sont imprimées sous forme de tablature, dans les contextes `TabStaff` et `TabVoice` qui comportent une clef spécifique ajoutée automatiquement.

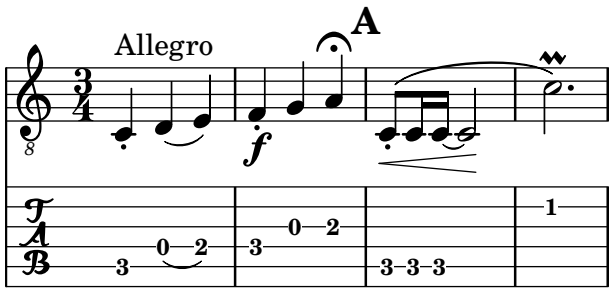
```
\new TabStaff \relative {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```



Par défaut, les tablatures ne comportent aucune marque de durée ni de symbole musical tel que des nuances.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c ~ 2\!
  c'2.\prall\)
}

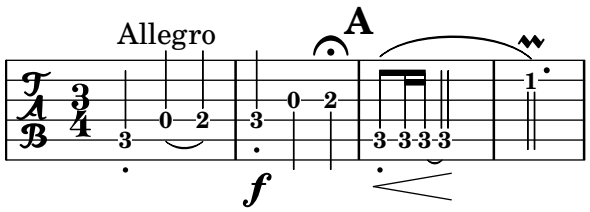
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff   { \symbols }
  >>
}
```



Pour obtenir les symboles de notation habituelle dans une tablature, il faut appliquer la commande `\tabFullNotation` au contexte `TabStaff`. Vous noterez par ailleurs que les blanches sont affublées d'une double hampe afin de ne pas les confondre avec des noires.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c ~ 2\!
  c'2.\prall\)
}

\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```

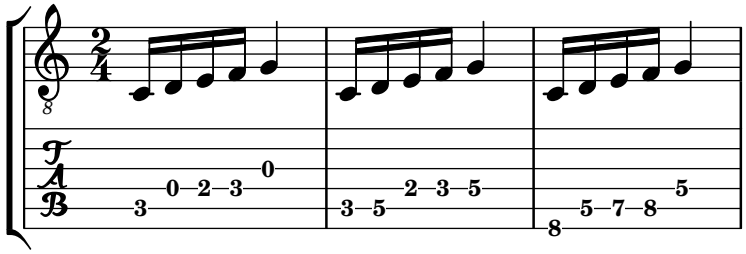


Quand aucune corde n'est précisée, LilyPond choisit automatiquement la corde pour laquelle la position est la moins élevée, avec une préférence pour une corde à vide. Vous pouvez préférer qu'une note donnée soit jouée sur une corde particulière, auquel cas l'affectation directe du numéro de corde suffit. L'absence d'indication des numéros de corde en notation traditionnelle se gère au niveau des stencils. Il est cependant plus facile de jouer sur la propriété `minimumFret`, dont la valeur par défaut est fixée à 0, ce qui correspond aux cordes à vide.

Cependant, et en dépit d'une affectation de `minimumFret`, une corde à vide aura toujours préséance. Ce comportement se modifie par l'activation de `restrainOpenStrings`.

```
\layout { \omit Voice.StringNumber }
\new StaffGroup <<
   \new Staff \relative {
     \clef "treble_8"
     \time 2/4
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     c,16 d e f g4
   }
   \new TabStaff \relative {
     c16 d e f g4
```

```
    c,16\5 d\5 e\4 f\4 g4\4
    \set TabStaff.minimumFret = #5
    \set TabStaff.restrainOpenStrings = ##t
    c,16 d e f g4
  }
>>
```



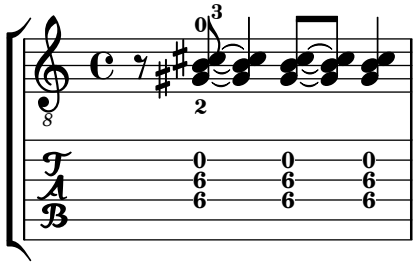
La répétition d'une construction en accord s'indique par un `q` – voir [Répétition d'accords], page 175. Cette fonctionnalité, bien qu'opérationnelle en mode tablature, supprime entre autres les numéros de corde et doigtés. Il vous faudra donc préalablement recourir explicitement à l'instruction

```
\chordRepeats #'(string-number-event fingering-event)
```

si vous utilisez la répétition d'accords dans vos tablatures. Cette instruction est abrégée en `\tabChordRepeats`.

```
guitar = \relative {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ 8 q4
}

\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
```

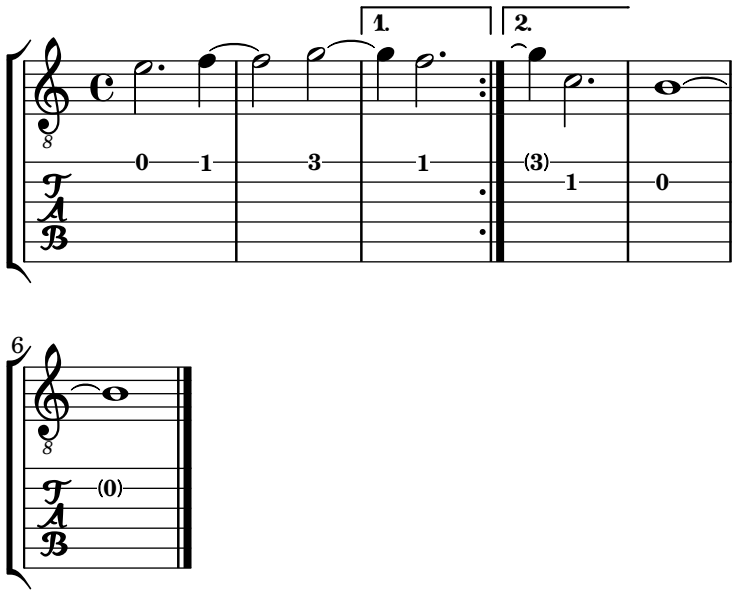


Lorsqu'une liaison de prolongation intervient à l'occasion d'un saut de ligne, la note est répétée, entre parenthèses. Il en va de même pour la seconde alternative d'une répétition.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~
  }
```

```
  \alternative {
     \volta 1 { g4 f2. }
     \volta 2 { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```

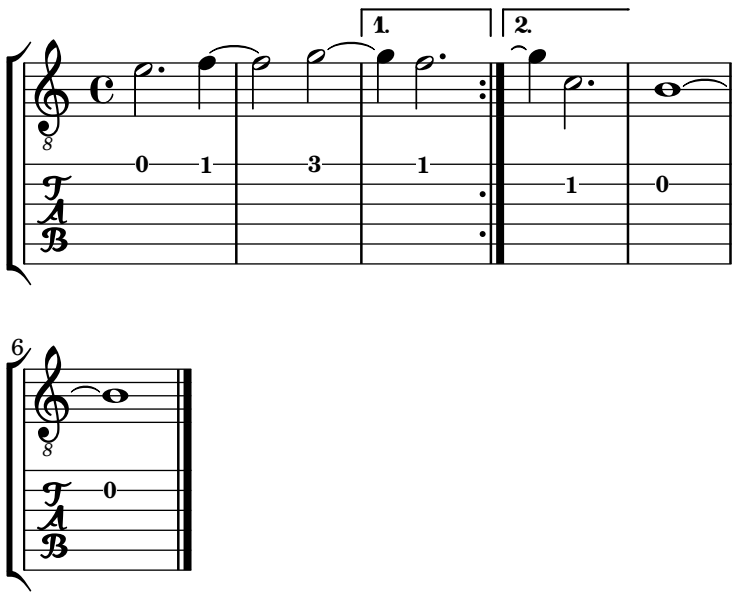


La commande `\hideSplitTiedTabNotes` permet d'éviter d'imprimer ces cases entre parenthèses.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~ }
```

```
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

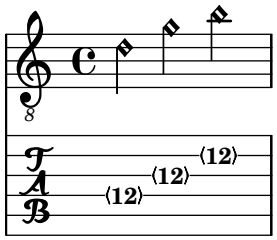
\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```



Les indications de sons harmoniques ainsi que les glissandos peuvent être ajoutés aux tablatures.

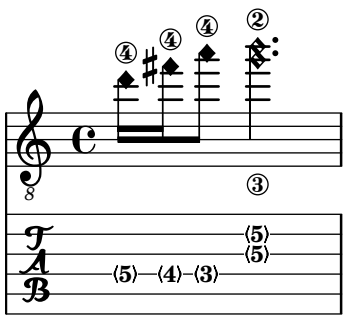
```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
```

```
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```



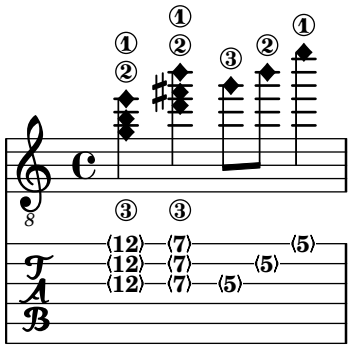
Vous noterez que la commande `\harmonic` s'attache toujours à une note unique (parfois contenue dans un accord) et non à l'ensemble d'un accord. Ceci ne fonctionne donc que pour une harmonique au douzième fret d'une corde à vide. Toute autre harmonique devrait être défini directement par LilyPond. Vous pouvez l'y aider en indiquant la case où le doigt viendrait se placer sur le manche.

```
fretHarmonics = {
  \harmonicByFret #5 d16\4
  \harmonicByFret #4 d16\4
  \harmonicByFret #3 d8\4
  \harmonicByFret #5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \fretHarmonics
    }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Un harmonique peut accessoirement se calculer à partir de la longueur de corde par rapport au doigté de cet harmonique.

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \ratioHarmonics
    }
    \new TabStaff { \ratioHarmonics }
  >>
}
```



Des désinences peuvent s'ajouter en notation de tablature. Elles s'indiquent par adjonction d'un `\^` à la note ou à l'accord de départ, et se terminent à la note ou l'accord suivant. Différents styles sont disponibles : le style par défaut imprime une courbe terminée par une flèche vers le haut ou le bas, le style `'hold` une ligne horizontale discontinue, `'pre-bend` une ligne verticale terminée en flèche, et `'pre-bend-hold` une ligne verticale terminée en flèche suivie d'une horizontale discontinue.

```
bend-styles = {
  <>^"default"
  f'4\^ g'4\^ f'2

  <>^"'hold"
  \grace f'4\^ g'1\bendHold \^ g'1

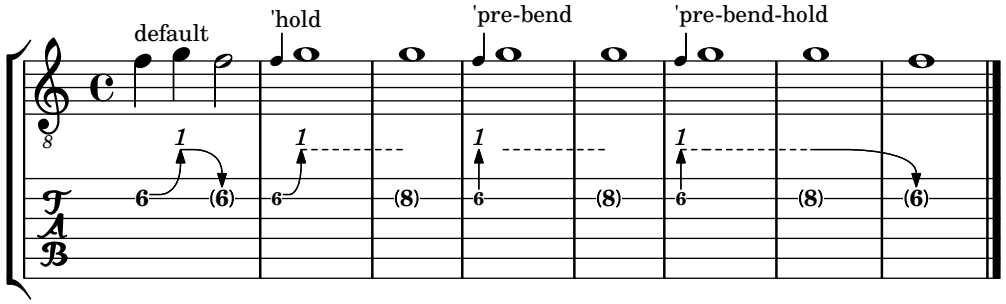
  <>^"'pre-bend"
  \grace f'4\preBend \^ g'1\bendHold \^ g'1

  <>^"'pre-bend-hold"
  \grace f'4\preBendHold \^ g'1\bendHold \^ g'1\^ f'

  \bar "|."
}

\score {
  \new StaffGroup
```

```
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \bend-styles
    }
    \new TabStaff \bend-styles
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabStaff
      minimumFret = #5
    }
    \context {
      \TabVoice
      \consists "Bend_spanner_engraver"
    }
  }
}
```



Les cordes à vide ne font habituellement pas l'objet d'une chute ou d'un saut. Il faut, pour qu'une corde soit concernée, basculer sa propriété `bend-me` à `#t`, et à `#f` pour exclure les autres notes de la désinence.

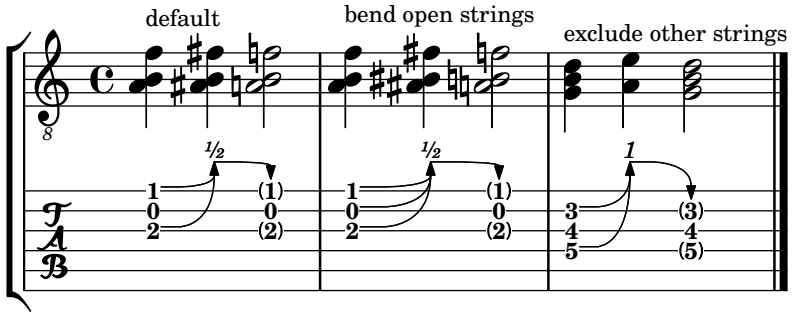
```
mus = {
  <>^"default"
  <a b f'>4\^
  <ais b fis'>\^
  <a b f'>2

  <>^"bend open strings"
  <a \tweak bend-me ##t b f'>4\^
  <ais \tweak bend-me ##t bis fis'>\^
  <a b f'>2

  <>^"exclude other strings"
  <g \tweak bend-me ##f b\3 d'>4\^
  <a e'\2 >\^
  <g \tweak bend-me ##f b\3 d'>2
```

```
  \bar "|."
}

\score {
  \new StaffGroup
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \mus
    }
    \new TabStaff \mus
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabVoice
      \consists "Bend_spanner_engraver"
    }
  }
}
```



Dans une succession de chutes ou sauts, la désinence de départ pourra nécessiter un réglage particulier de `details.successive-level`. La fonction `bendStartLevel`, qui prend en argument un entier, est là pour le gérer.

```
printNext = -\tweak details.target-visibility ##t \etc

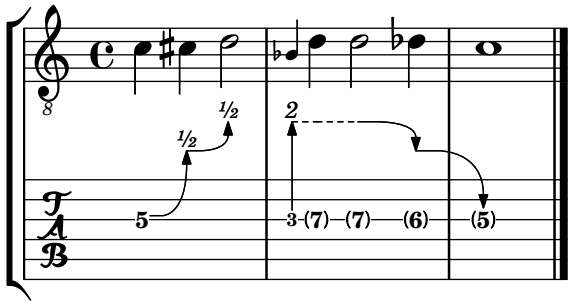
mus = {
  c'4\3\^ cis'\3 \^ d'2\3

  \grace bes4\3\preBendHold \bendStartLevel 2 \printNext \^
  d'4\3\bendHold \^ d'2\3\^ des'4\3 \^ c'1\3

  \bar "|."
}

\score {
  \new StaffGroup
```

```
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \mus
    }
    \new TabStaff \mus
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabVoice
      \consists "Bend_spanner_engraver"
    }
  }
}
```

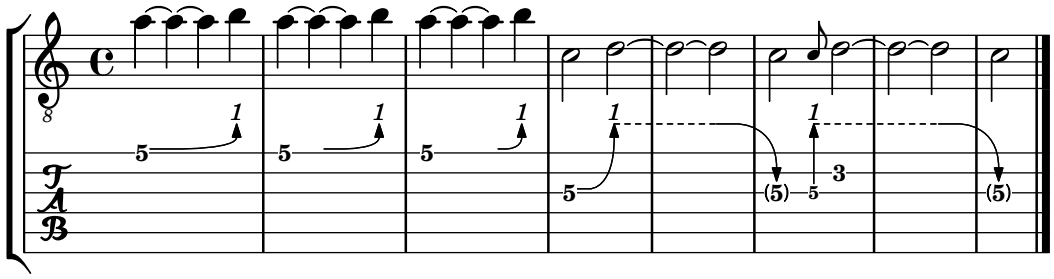


L'extension du `BendSpanner` se termine par défaut sur la note ou l'accord qui suit, même en présence d'une liaison de prolongation. Un élément `NoteColumn` particulier peut être sauté à l'aide d'un `\skipNC`. Un groupe de `NoteColumns` peut se sauter en le faisant précéder d'un `\skipNCs` et suivre d'un `\endSkipNCs`.

```
bends-with-ties-and-skips = {
  a'4~\^ \skipNC a'4~ \skipNC a'4 b'4
  a'4~ a'4~\^ \skipNC a'4 b'4
  a'4~ a'4~ a'4\^ b'4
  c'2\^ d'~ \bendHold \^ \skipNC d'~ d'\^ c'
  \grace { c'8-\preBendHold \^ }
  \skipNCs d'2~ d'2~ \endSkipNCs d'\^ c'2
  \bar "|."
}

\score {
  \new StaffGroup
  <<
    \new Staff {
      \clef "G_8"
      \bends-with-ties-and-skips
    }
    \new TabVoice \bends-with-ties-and-skips
```

```
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabStaff
      minimumFret = #3
      restrainOpenStrings = ##t
    }
    \context {
      \TabVoice
      \consists "Bend_spanner_engraver"
    }
  }
}
```



Commandes prédéfinies

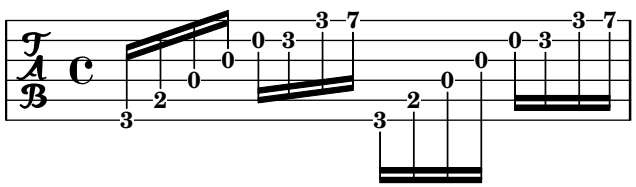
`\skipNCs`, `\skipNC`, `\endSkipNCs`.

Morceaux choisis

Hampes et ligatures en mode tablature

La direction des hampes se gère dans les tablatures tout comme en notation traditionnelle. Les ligatures peuvent être mises à l'horizontale comme le montre cet exemple.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
    g,,16 b d g b d g b
  }
}
```



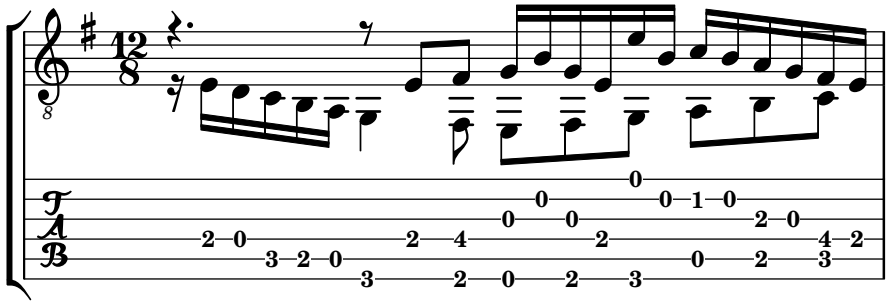
Polyphonie en mode tablature

Une section polyphonique s'obtient dans un `TabStaff` de la même manière que dans une portée normale.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  <<
    \new StaffGroup = "tab with traditional" <<
      \new Staff = "guitar traditional" <<
        \clef "treble_8"
        \new Voice = "upper" \upper
        \new Voice = "lower" \lower
      >>
      \new TabStaff = "guitar tab" <<
        \new TabVoice = "upper" \upper
        \new TabVoice = "lower" \lower
      >>
    >>
  >>
}
```



Harmoniques sur corde à vide en tablature

Voici comment obtenir des harmoniques sur corde à vide (harmoniques naturelles) dans une tablature.

```
openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = #3
  \override TextSpanner.dash-fraction = #0.3
  \override TextSpanner.dash-period = #1
```

```
%first harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "1st harm. "
\harmonicByFret #12 e,2\6\startTextSpan
\harmonicByRatio #1/2 e,\6\stopTextSpan

%second harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "2nd harm. "
\harmonicByFret #7 e,\6\startTextSpan
\harmonicByRatio #1/3 e,\6
\harmonicByFret #19 e,\6
\harmonicByRatio #2/3 e,\6\stopTextSpan
%\harmonicByFret #19 < e,\6 a,\5 d\4 >
%\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >

%third harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "3rd harm. "
\harmonicByFret #5 e,\6\startTextSpan
\harmonicByRatio #1/4 e,\6
\harmonicByFret #24 e,\6
\harmonicByRatio #3/4 e,\6\stopTextSpan
\break

%fourth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "4th harm. "
\harmonicByFret #4 e,\6\startTextSpan
\harmonicByRatio #1/5 e,\6
\harmonicByFret #9 e,\6
\harmonicByRatio #2/5 e,\6
\harmonicByFret #16 e,\6
\harmonicByRatio #3/5 e,\6\stopTextSpan

%fifth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "5th harm. "
\harmonicByFret #3 e,\6\startTextSpan
\harmonicByRatio #1/6 e,\6\stopTextSpan
\break

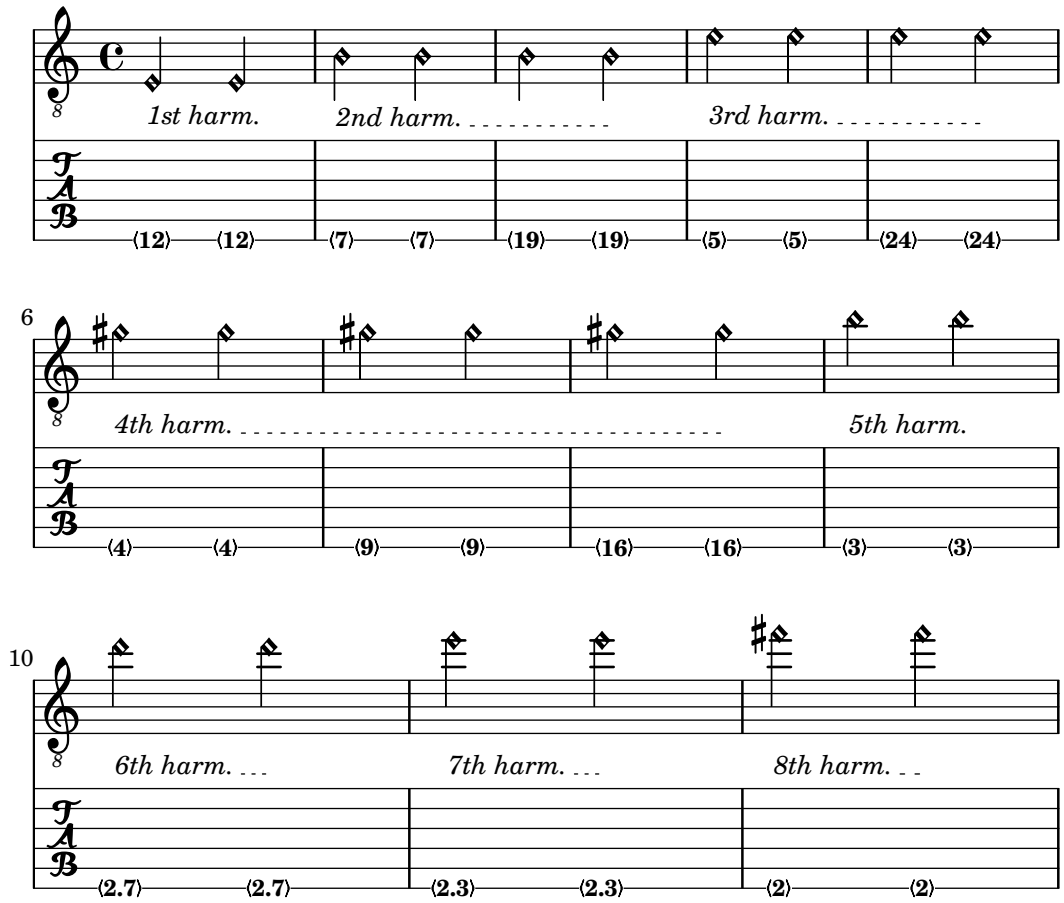
%sixth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "6th harm. "
\harmonicByFret #2.7 e,\6\startTextSpan
\harmonicByRatio #1/7 e,\6\stopTextSpan

%seventh harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "7th harm. "
\harmonicByFret #2.3 e,\6\startTextSpan
```

```
  \harmonicByRatio #1/8 e,\6\stopTextSpan

  %eighth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "8th harm. "
  \harmonicByFret #2 e,\6\startTextSpan
  \harmonicByRatio #1/9 e,\6\stopTextSpan
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}
```



Harmoniques et tablature

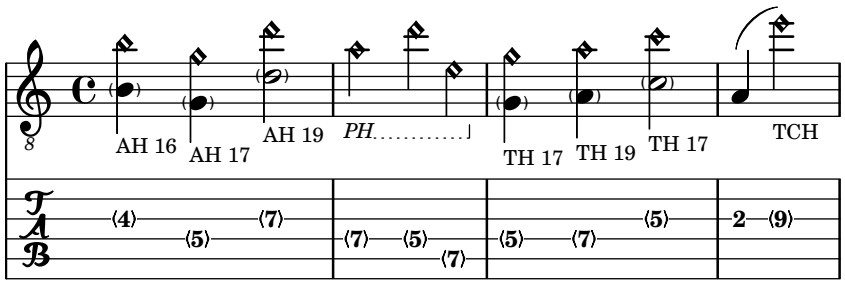
Harmoniques et tablature (harmoniques artificielles).

```
pinchedHarmonics = {
   \textSpannerDown
   \override TextSpanner.bound-details.left.text =
      \markup {\halign #-0.5 \teeny "PH" }
      \override TextSpanner.style =
         #'dashed-line
   \override TextSpanner.dash-period = #0.6
   \override TextSpanner.bound-details.right.attach-dir = #1
   \override TextSpanner.bound-details.right.text =
      \markup { \draw-line #'(0 . 1) }
   \override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup { \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup { \teeny "AH 17" }
  <\parenthesize d' d''\harmonic>2_\markup { \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d''\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup { \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup { \teeny "TH 19" }
  <\parenthesize c'\3 c''\harmonic>2_\markup { \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e''\harmonic>2. )_\markup { \teeny "TCH" }
}

frettedStrings = {
  %artificial harmonics (AH)
  \harmonicByFret #4 g4\3
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 g2\3
  %pinched harmonics (PH)
  \harmonicByFret #7 d2\4
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 a4\5
  %tapped harmonics (TH)
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 d4\4
  \harmonicByFret #5 g2\3
  %touch harmonics (TCH)
  a4 \harmonicByFret #9 g2.\3
}
```

```
\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}
```



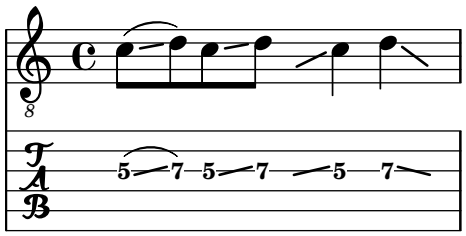
Glissando et tablature

Un glissando s'indique dans un `TabStaff` tout comme dans un `Staff`.

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
  \stemDown \hideNotes
  g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = #4
      \override Glissando.springs-and-rods =
                          #ly:spanner::set-spacing-rods
```

```
      \override Glissando.thickness = #2
      \omit StringNumber
      % or:
      %\override StringNumber.stencil = ##f
    }
  }
}
```



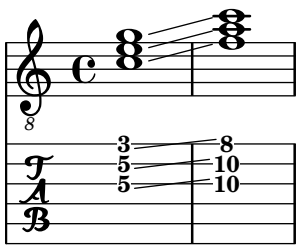
Glissando d'accords et tablature

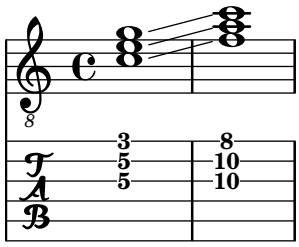
Un glissando sur des accords s'indique dans un `TabStaff` de la même manière que dans un `Staff`, à ceci près que nous aurons besoin des numéros de corde afin de déterminer correctement les frets d'arrivée.

```
myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \myMusic
  >>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \with { \override Glissando.style = #'none } {
      \myMusic
    }
  >>
}
```

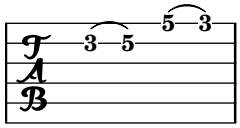




Hammer on et pull off

Hammer-on et *pull-off* peuvent s'indiquer par des liaisons.

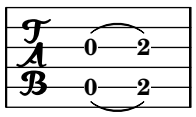
```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



Hammer on et pull off gérés par les voix

L'arc des *hammer-on* et *pull-off* est ascendant dans les voix une et trois, et descendant dans les voix deux et quatre.

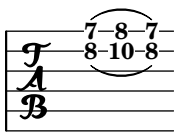
```
\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}
```



Hammer on et pull off avec accords

Dans le cadre de notes en accord, les *hammer-on* et *pull-off* sont indiqués par un arc simple. Vous obtiendrez néanmoins un arc double en réglant la propriété `doubleSlurs` sur `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```



Voir aussi

Manuel de notation : [Glissando], page 146, [Hampes], page 245, [Harmoniques], page 367, [Répétition d'accords], page 175, [Répétitions explicites], page 157.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

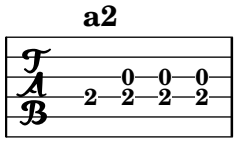
Référence des propriétés internes : Section "Beam" dans *Référence des propriétés internes*, Section "TabNoteHead" dans *Référence des propriétés internes*, Section "TabStaff" dans *Référence des propriétés internes*, Section "TabVoice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les accords ne subissent aucun traitement particulier ; de ce fait, la sélection automatique des cordes peut attribuer une même corde pour deux notes différentes de l'accord.

Afin que `\partCombine` fonctionne avec des tablatures, on doit ajouter au contexte `TabStaff` des voix fantômes :

```
melodia = \partCombine { e4 g g g } { e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodia }
  >>
>>
```



Le support des modes de jeu propres à la guitare se limite aux sons harmoniques et aux glissandos.

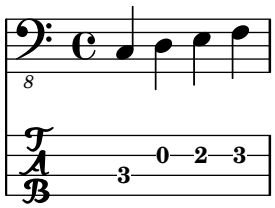
Tablatures personnalisées

Sous LilyPond, la case correspondant à une note jouée sur une corde donnée est calculée automatiquement. Pour ce faire, l'accordage doit être spécifié. L'accordage des cordes est donné par la propriété `stringTunings`.

LilyPond possède des accordages prédéfinis pour le banjo, la mandoline, la guitare et la guitare basse ainsi que le ukulele et les cordes d'orchestre. LilyPond calcule automatiquement la transposition correspondant à ces accordages. L'exemple ci-dessous est pour guitare basse, il sonnera donc une octave en dessous de ce qui est écrit.

```
<<
  \new Voice \with {
    \omit StringNumber
  } {
    \clef "bass_8"
    \relative {
      c,4 d e f
    }
  }
  \new TabStaff \with {
```

```
    stringTunings = #bass-tuning
  } {
    \relative {
      c,4 d e f
    }
  }
>>
```



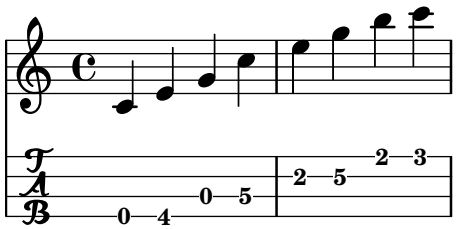
L'accordage par défaut est `guitar-tuning_`; il correspond à l'accordage standard d'une guitare : mi la ré sol si mi (EADGBE). D'autres accordages prédéfinis sont disponibles : `guitar-open-g-tuning`, `mandolin-tuning` et `banjo-open-g-tuning`. Les accordages prédéfinis sont répertoriés dans le fichier `ly/string-tunings-init.ly`.

LilyPond vous permet de créer n'importe quel accordage. L'accordage du contexte en cours se détermine à l'aide de la fonction `\stringTuning`. Celle-ci prend deux arguments : une représentation symbolique qui gardera l'accordage en mémoire, et une construction d'accord définissant la hauteur des différentes cordes. Les hauteurs fournies s'expriment impérativement en mode absolu – voir [Hauteurs avec octave absolue], page 1. La corde ayant le numéro le plus élevé (généralement la note la plus basse) est mentionnée en premier.

Nous pouvons ainsi définir par exemple l'accordage d'un instrument à quatre cordes accordées do sol ré la, soit en anglais `a''`, `d''`, `g'`, et `c'` :

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    #(define custom-tuning #{ \stringTuning <c' g' d'' a''> #})
    \set Staff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



La propriété `stringTunings` permet aussi au `FretBoards` de calculer automatiquement les diagrammes de frets.

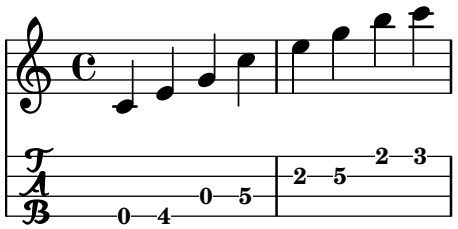
L'accordage fait partie de la clé permettant d'identifier les diagrammes prédéfinis – voir [Tablatures prédéfinies], page 403.

Nous pourrions donc écrire l'exemple précédent ainsi :

```
custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



L'accordage est constitué, en interne, par une liste Scheme des hauteurs de note correspondant aux cordes à vide, une note pour chaque corde, classée par numéro de corde de 1 à n, où la corde 1 est la plus haute dans la tablature et n la plus basse. Cela revient généralement à classer les cordes de la plus aiguë à la plus grave, mais certains instruments (comme le ukulele) n'ont pas les cordes classées par hauteur.

Chaque hauteur de corde incluse dans un accordage est un objet LilyPond de type *pitch*. Les objets *pitch* sont créés par la fonction `ly:make-pitch` – voir Section A.22 [Fonctions Scheme], page 863.

La fonction `\stringTuning` permet de créer de tels objets à partir de la saisie d'un accord.

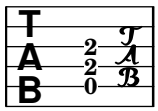
LilyPond calcule automatiquement le nombre de cordes à représenter dans la tablature (`TabStaff`) ainsi que dans le `FretBoard` en comptant le nombre d'éléments définis dans le `stringTunings`.

Les différents contextes `TabStaff` utiliseront par défaut un même accordage personnalisé dès lors que votre fichier comportera une clause

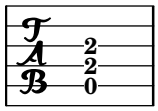
```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

LilyPond dispose d'une clef de tablature moderne.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```



2



Cette clef moderne prend en charge les tablatures de quatre à sept cordes.

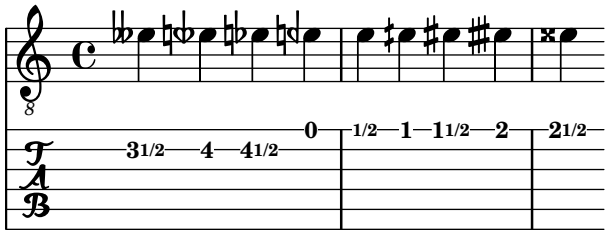
Un `TabStaff` peut contenir des micro-intervalles tels les quarts de ton, qui interveiennent dans les chutes ou sauts. L'assertion `supportNonIntegerFret = ##t` devra se placer au niveau du contexte `Score`. Les micro-intervalles ne sont toutefois pas pris en charge dans un contexte `FretBoards`.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

custom-tuning = \stringTuning <e, a, d ges beh eeh'>

mus = \relative {
  eeses'4
  eeseh
  ees
  eeh
  e
  eih
  eis
  eisih
  eisis
}

<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \with { stringTunings = \custom-tuning } \mus
>>
```



Voir aussi

Manuel de notation : Section A.22 [Fonctions Scheme], page 863, [Hauteurs avec octave absolue], page 1, [Tablatures prédéfinies], page 403.

Fichiers d'initialisation : `ly/string-tunings-init.ly`, `scm/tablature.scm`.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Tab_note_heads_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

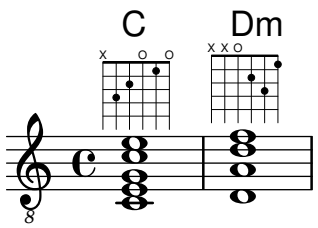
Le calcul automatique des tablatures se révèle inapproprié pour les instruments dont l'ordre des cordes ne correspond pas à l'ordre des hauteurs, comme le ukulele.

Tablatures sous forme d'étiquette

On peut ajouter des diagrammes d'accord au-dessus de n'importe quelle note, en tant qu'objets `\markup`. Ces objets contiennent toutes les informations du diagramme d'accord. Il y a pour les définir trois interfaces « diagramme d'accord » (*fret-diagram* en anglais) : *standard*, *terse* et *verbose*. Ces trois interfaces produiront des diagrammes similaires mais permettent des niveaux de personnalisation différents. Des détails à propos des interfaces de type `\markup` peuvent être trouvés à l'annexe Section A.11.5 [Instrument Specific Markup], page 785.

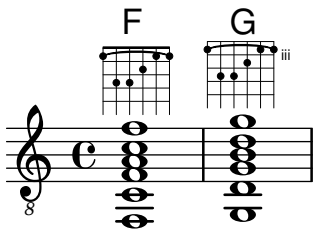
Dans l'interface standard des diagrammes d'accord, on indiquera le numéro de corde et le numéro de case pour chaque point placé sur une corde. Les cordes à vide et étouffées peuvent aussi être indiquées.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



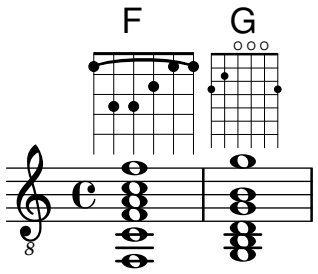
Les indications de barré peuvent aussi être ajoutées au diagramme d'accord dans l'interface standard :

```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram "c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
    }
  }
>>
```



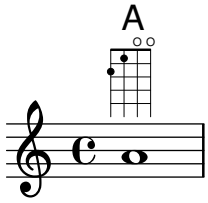
La taille du diagramme d'accord ainsi que le nombre de cases représentées peuvent aussi être modifiés dans l'interface standard.

```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram "h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
    }
  }
>>
```



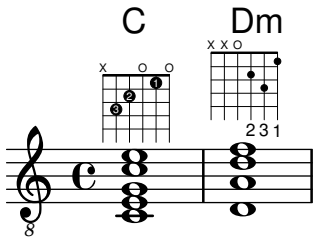
Le nombre de cordes dans les diagrammes d'accord peut être modifié dans l'interface standard pour s'adapter à différents instruments tels que le banjo et le ukulele.

```
<<
  \new ChordNames {
    \chordmode {
      a1
    }
  }
  \new Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram "w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



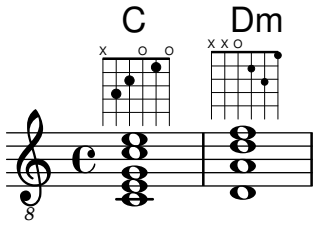
Des indications de doigtés peuvent être ajoutées, et le positionnement de ces doigtés peut être modifié dans l'interface standard.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
    }
  }
>>
```



La taille ainsi que la position des points peuvent être contrôlées dans l'interface standard.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



Pour les gauchers qui inversent les cordes, LilyPond permet d'imprimer les diagrammes de fret correctement.

```
\markup
  \center-column {
    "C"
    "(gaucher)"
     \override #`(fret-diagram-details . ((handedness . ,LEFT)))
     \fret-diagram "6-x;5-3-3;4-2-2;3-o;2-1;1-o;"
  }
```

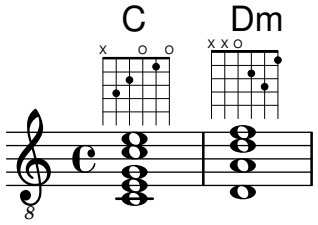
C
(gaucher)



Dans l'interface `fret-diagram-terse`, les numéros de corde sont omis ; les numéros de corde sont induits par la présence de points-virgules. Il y a un point-virgule pour chaque corde du diagramme. Le premier point-virgule correspondant au plus haut numéro de corde, le dernier à la première corde. Les cordes étouffées, les cordes à vide ainsi que les numéros de case peuvent y être indiqués.

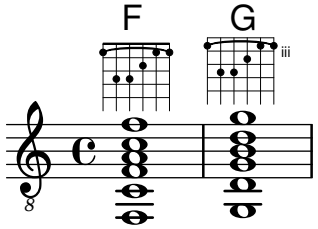
```
<<
```

```
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3;2;o;1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2;3;1;"
    }
  }
>>
```



Les indications de barré peuvent être incluses dans l'interface `fret-diagram-terse`.

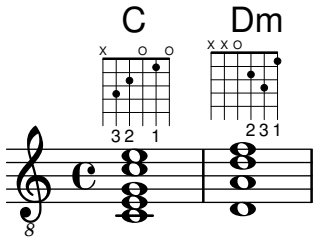
```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram-terse "1-(;3;3;2;1;1-);"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram-terse "3-(;5;5;4;3;3-);"
    }
  }
>>
```



Les indications de doigtés peuvent être incluses dans l'interface `fret-diagram-terse`.

```
<<
  \new ChordNames {
```

```
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \override Voice.TextScript.fret-diagram-details.finger-code =
      #'below-string
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3-3;2-2;o;1-1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2-2;3-3;1-1;"
    }
  }
>>
```



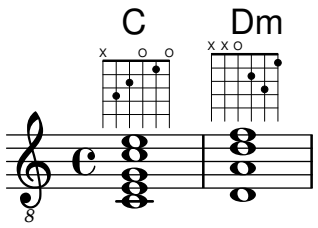
Les autres propriétés des diagrammes d'accord doivent être ajustées en utilisant la commande `\override` dans l'interface `fret-diagram-terse`.

Il n'est possible d'inclure qu'une seule indication par corde dans un *markup* `fret-diagram-terse`. Il faudra, pour en inclure plusieurs, utiliser un *markup* `fret-diagram` ou `fret-diagram-verbose`.

L'interface `fret-diagram-verbose` est au format d'une liste Scheme. Chaque élément de la liste décrit un objet devant être placé dans le diagramme d'accord.

```
<<
    \new ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
```

```
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



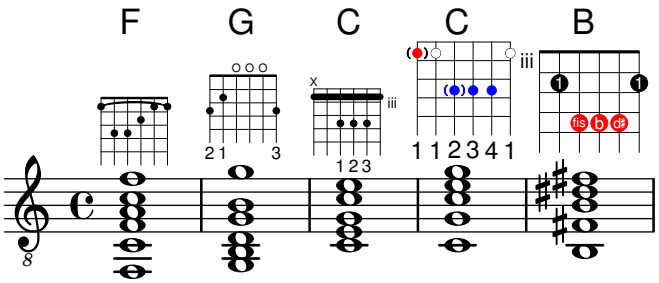
Les indications de doigté et de barré peuvent être décrites dans l'interface `fret-diagram-verbose`. Particularité propre à l'interface `fret-diagram-verbose` : l'indication de capodastre dans le diagramme d'accord. L'indication de capodastre est une petite ligne transversale aux cordes. La case avec le capodastre est la case la plus basse du diagramme d'accord.

Les points d'indication de doigté peuvent se colorier ou être mis entre parenthèses ; la couleur des parenthèses est indépendante de celle du point.

Des *markups* peuvent par ailleurs venir s'insérer dans les points.

```
<<
    \new ChordNames {
      \chordmode {
        f1 g c c b
      }
    }
  \new Staff {
    \clef "treble_8"
    \override Voice.TextScript.fret-diagram-details.finger-code =
      #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
```

```
          (open 4)
          (open 3)
          (open 2)
          (place-fret 1 3 3)
        )
      }
      <c e g c' e'>1^\markup {
        \fret-diagram-verbose #'(
          (capo 3)
          (mute 6)
          (place-fret 4 5 1)
          (place-fret 3 5 2)
          (place-fret 2 5 3)
        )
      }
      \override Voice.TextScript.size = 1.4
      <c g c' e' g'>1^\markup {
        \fret-diagram-verbose #'(
          (place-fret 6 3 1 red parenthesized default-paren-color)
          (place-fret 5 3 1 inverted)
          (place-fret 4 5 2 blue parenthesized)
          (place-fret 3 5 3 blue)
          (place-fret 2 5 4 blue)
          (place-fret 1 3 1 inverted)
        )
      }
      \override Voice.TextScript.size = 1.5
      <b, fis b dis' fis'>1^\markup
        \override #'(fret-diagram-details . ((finger-code . in-dot)))
        \fret-diagram-verbose #`(
          (place-fret 5 2 1)
          (place-fret 4 4 "fis" red)
          (place-fret 3 4 "b" red)
          (place-fret
            2 4
            ,#{ \markup
                  \concat {
                    \vcenter "d"
                    \fontsize #-5
                    \musicglyph "accidentals.sharp"} #}
            red)
          (place-fret 1 2 1)
        )
    }
  >>
```



Toutes les autres propriétés du diagramme d'accord doivent être indiquées en utilisant la commande `\override` lorsque l'on utilise l'interface `fret-diagram-verbose`.

La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur grâce aux propriétés de l'interface `fret-diagram-interface`. Des détails se trouvent dans Section "fret-diagram-interface" dans *Référence des propriétés internes*. Pour un diagramme d'accord, les propriétés de l'interface dépendent de `Voice.TextScript`.

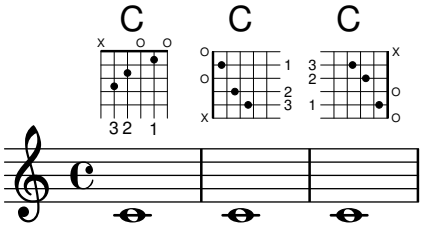
Morceaux choisis

Orientation des diagrammes de fret

Les diagrammes de fret peuvent s'orienter de trois manières différentes. Ils s'aligneront par défaut sur la corde du haut ou le sommet du fret.

```
\include "predefined-guitar-fretboards.ly"

<<
  \chords {
    c1
    c1
    c1
  }
  \new FretBoards {
    \chordmode {
      c1
      \override FretBoard.fret-diagram-details.orientation =
        #'landscape
      c1
      \override FretBoard.fret-diagram-details.orientation =
        #'opposing-landscape
      c1
    }
  }
  \new Voice {
    c'1
    c'1
    c'
  }
>>
```



Personnalisation des diagrammes de fret

Les propriétés d'un diagramme de fret sont modifiables grâce au `fret-diagram-details`. Lorsqu'ils sont générés sous forme de `\markup`, rien n'empêche de modifier les diagrammes en jouant sur les réglages de l'objet `Voice.TextScript` ou bien directement sur le *markup*.

```
<<
  \chords { c1 | c | c | d }

  \new Voice = "mel" {
    \textLengthOn
    % Set global properties of fret diagram
    \override TextScript.size = #'1.2
    \override TextScript.fret-diagram-details.finger-code = #'in-dot
    \override TextScript.fret-diagram-details.dot-color = #'white

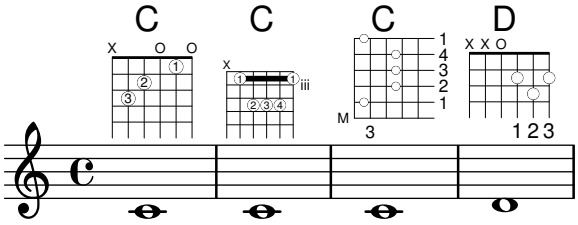
    %% C major for guitar, no barre, using defaults
       % terse style
    c'1^\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

    %% C major for guitar, barred on third fret
       % verbose style
       % size 1.0
       % roman fret label, finger labels below string, straight barre
    c'1^\markup {
      % standard size
      \override #'(size . 1.0) {
        \override #'(fret-diagram-details . (
                     (number-type . roman-lower)
                     (finger-code . in-dot)
                     (barre-type . straight))) {
          \fret-diagram-verbose #'((mute 6)
                                   (place-fret 5 3 1)
                                   (place-fret 4 5 2)
                                   (place-fret 3 5 3)
                                   (place-fret 2 5 4)
                                   (place-fret 1 3 1)
                                   (barre 5 1 3))
        }
      }
    }

    %% C major for guitar, barred on third fret
       % verbose style
       % landscape orientation, arabic numbers, M for mute string
       % no barre, fret label down or left, small mute label font
    c'1^\markup {
      \override #'(fret-diagram-details . (
                   (finger-code . below-string)
                   (number-type . arabic)
                   (label-dir . -1)
                   (mute-string . "M")
                   (orientation . landscape)
```

```
                    (barre-type . none)
                    (xo-font-magnification . 0.4)
                    (xo-padding . 0.3))) {
          \fret-diagram-verbose #'((mute 6)
                                   (place-fret 5 3 1)
                                   (place-fret 4 5 2)
                                   (place-fret 3 5 3)
                                   (place-fret 2 5 4)
                                   (place-fret 1 3 1)
                                   (barre 5 1 3))
        }
      }

      %% simple D chord
         % terse style
         % larger dots, centered dots, fewer frets
         % label below string
      d'1^\markup {
        \override #'(fret-diagram-details . (
                    (finger-code . below-string)
                    (dot-radius . 0.35)
                    (dot-position . 0.5)
                    (fret-count . 3))) {
          \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
        }
      }
    }
  >>
```



Voir aussi

Manuel de notation : Section “Markups spécifiques aux instruments (en anglais)” dans *Manuel de notation*.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

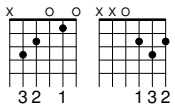
Référence des propriétés internes : Section “fret-diagram-interface” dans *Référence des propriétés internes*.

Tablatures prédéfinies

Les diagrammes d’accord peuvent être affichés en utilisant le contexte `FretBoards`. Par défaut le contexte `FretBoards` affichera des diagrammes d’accord stockés dans une table de correspondance :

```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
```

```
  }
}
```



Les diagrammes d'accord définis par défaut sont dans le fichier `predefined-guitar-fretboards.ly`. Les diagrammes d'accord sont stockés en fonction des notes de l'accord ainsi que de l'accordage (`stringTunings`) utilisé. Le fichier d'initialisation `predefined-guitar-fretboards.ly` contient les diagrammes d'accord prédéfinis uniquement pour l'accordage standard (`guitar-tuning`). Des diagrammes d'accords peuvent être définis pour d'autres instruments ou d'autres accordages en suivant les exemples du fichier `predefined-guitar-fretboards.ly`.

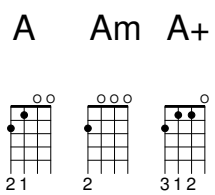
Les diagrammes de fret propres au ukulele se trouvent dans le fichier `predefined-ukulele-fretboards.ly`.

```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}
```



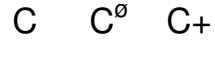
Les diagrammes de fret propres à la mandoline se trouvent dans le fichier `predefined-mandolin-fretboards.ly`.

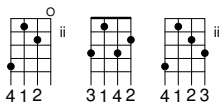
```
\include "predefined-mandolin-fretboards.ly"

myChords = \chordmode { c1 c:m7.5- c:aug }

\new ChordNames {
  \myChords
}

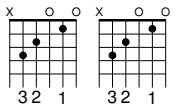
\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```





Les notes des accords peuvent être entrées aussi bien comme musique simultanée qu'en utilisant le mode accord (voir [Généralités sur le mode accords], page 445).

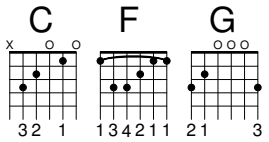
```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {c1}
  <c' e' g'>1
}
```



Il est courant d'afficher simultanément le nom des accords et les diagrammes d'accord correspondants. Ceci s'obtient en mettant en parallèle un contexte `ChordNames` et un contexte `FretBoards`, tout en affectant aux deux la même musique.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 f g
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```

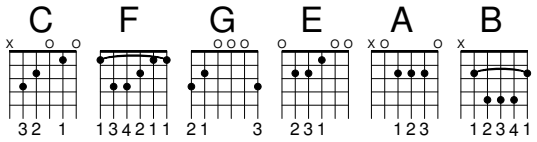


Les diagrammes d'accord prédéfinis sont transposables tant qu'il y a un diagramme correspondant à l'accord transposé dans la base des diagrammes d'accord.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 f g
}

mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \new ChordNames {
    \mychordlist
  }
  \new FretBoards {
```

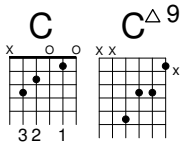
```
    \mychordlist
  }
>>
```



La table des diagrammes d'accord contient sept types d'accord (majeur, mineur, augmenté, diminué, septième de dominante, majeur sept, mineur sept) pour chacune des 17 fondamentales possibles. Une liste complète des diagrammes d'accords prédéfinis se trouve à l'annexe [Tablatures prédéfinies], page 403. S'il n'y a pas d'entrée dans la table pour un accord donné, le graveur `Fretboard_engraver` calculera le diagramme d'accord en utilisant la fonctionnalité automatique décrite dans [Tablatures automatiques], page 414.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



Des diagrammes d'accord peuvent être ajoutés à la table des diagrammes d'accord. Pour ajouter un diagramme d'accord, il faut spécifier l'accord correspondant au diagramme, l'accord utilisé et la définition du diagramme. Cette définition de diagramme peut être aussi bien de type *terse* que *verbose*.

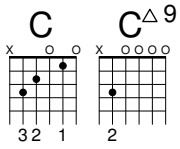
```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        "x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
```

```
  }
  \new FretBoards {
    \mychords
  }
>>
```



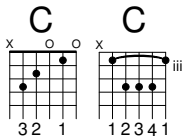
On peut enregistrer différents diagrammes pour un même accord en les définissant à des octaves différentes. Notez qu'il faut un intervalle de deux octaves, le premier servant à la transposition.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
                        #guitar-tuning
                        #(offset-fret 2
                           (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



En plus des diagrammes d'accord, LilyPond possède une liste interne de carrures d'accord. Les carrures d'accords sont des diagrammes d'accord qui peuvent être transposés le long du manche. Les carrures d'accords peuvent être ajoutées à la liste interne et être ensuite utilisées pour définir des accords prédéfinis. Puisqu'elles sont transposables le long du manche, les carrures d'accord ne contiennent généralement pas de corde à vide. Tout comme les diagrammes d'accord, les carrures d'accord sont définies grâce aux interfaces `fret-diagram-terse` ou `fret-diagram-verbose`.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

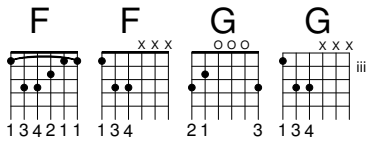
\addChordShape #'powerf #guitar-tuning "1-1;3-3;3-4;x;x;x;"
```

```
% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
                        \chordmode { f'' }
                        #guitar-tuning
                        #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
                        \chordmode { g'' }
                        #guitar-tuning
                        #(offset-fret 2
                           (chord-shape 'powerf guitar-tuning))

mychords = \chordmode{
  f1 f'' g g''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur grâce aux propriétés de l'interface `fret-diagram-interface`. Pour plus d'information, consultez Section "fret-diagram-interface" dans *Référence des propriétés internes*. Pour un diagramme d'accord donné, les propriétés de l'interface dépendent de `FretBoards.FretBoard`.

Morceaux choisis

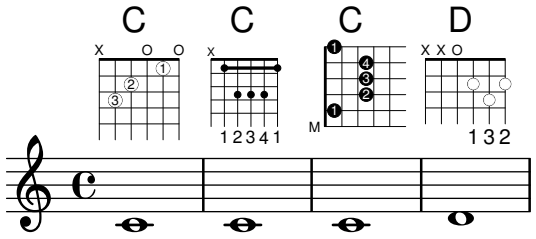
Personnalisation de diagrammes de fret

Les propriétés d'un diagramme de fret sont définies par les `fret-diagram-details`. En matière de diagramme de fret, les adaptations s'appliquent à l'objet `FretBoards.FretBoard`. Un `FretBoards` est comparable à un `Voice` : il s'agit d'un contexte du plus bas niveau, et il n'est donc pas primordial de l'instancier de manière explicite pour adapter ses propriétés.

```
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        #"x;1-1-(;3-2;3-3;3-4;1-1-);"

% shorthand
oo = #(define-music-function
       (grob-path value)
       (list? scheme?)
       #{ \once \override $grob-path = #value #})
```

```
<<
  \new ChordNames {
    \chordmode { c1 | c | c | d }
  }
  \new FretBoards {
    % Set global properties of fret diagram
    \override FretBoards.FretBoard.size = #'1.2
    \override FretBoard.fret-diagram-details.finger-code = #'in-dot
    \override FretBoard.fret-diagram-details.dot-color = #'white
    \chordmode {
      c
      \oo FretBoard.size #'1.0
      \oo FretBoard.fret-diagram-details.barre-type #'straight
      \oo FretBoard.fret-diagram-details.dot-color #'black
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
      c'
      \oo FretBoard.fret-diagram-details.barre-type #'none
      \oo FretBoard.fret-diagram-details.number-type #'arabic
      \oo FretBoard.fret-diagram-details.orientation #'landscape
      \oo FretBoard.fret-diagram-details.mute-string #"M"
      \oo FretBoard.fret-diagram-details.label-dir #LEFT
      \oo FretBoard.fret-diagram-details.dot-color #'black
      c'
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
      \oo FretBoard.fret-diagram-details.dot-radius #0.35
      \oo FretBoard.fret-diagram-details.dot-position #0.5
      \oo FretBoard.fret-diagram-details.fret-count #3
      d
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>
```



Création de diagrammes de fret prédéfinis pour d'autres instruments

La liste des diagrammes standards prédéfinis pour la guitare peut être augmentée d'autres définitions spécifiques à d'autres instruments. Voici comment définir un nouvel accordage ainsi que quelques diagrammes prédéfinis pour le « cuatro vénézuélien ».

Cet exemple illustre aussi la manière d'ajouter des doigtés aux accords ; ils serviront de référence pour la boucle d'accord et seront indiqués dans les diagrammes et le `TabStaff`, mais pas dans la musique.

Ces diagrammes ne peuvent pas être transposés, dans la mesure où ils contiennent des informations sur les cordes. Ceci est amené à évoluer.

```
% add FretBoards for the Cuatro
%   Note: This section could be put into a separate file
%      predefined-cuatro-fretboards.ly
%      and \included into each of your compositions

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        #"o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        #"2-2;o;1-1;o;"

% end of potential include file /predefined-cuatro-fretboards.ly


#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
```

```
      \set chordChanges = ##t
      \primerosNames
    }

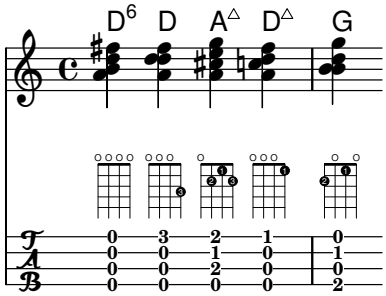
    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = 4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        #(ly:make-moment 1 16)
    }
  }
  \midi { }
}
```



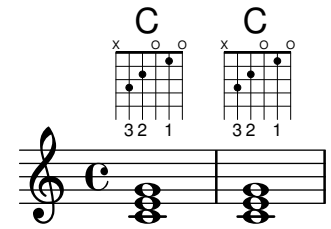
Changement d'accord et diagramme de fret

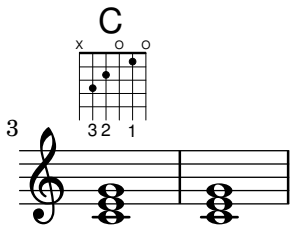
Vous pouvez opter pour n'imprimer les diagrammes de fret qu'à l'occasion d'un changement d'accord ou de saut de ligne.

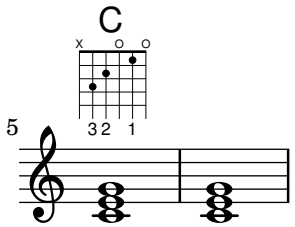
```
\include "predefined-guitar-fretboards.ly"
```

```
myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```







Diagrammes de fret alternatifs

Vous pouvez tout à fait créer des tables de diagrammes de fret supplémentaires, notamment dans l'optique d'un fret alternatif pour un accord donné.

Avant de pouvoir utiliser un diagramme alternatif, vous devrez avoir alimenté une table à cet effet. Les différents diagrammes seront ajoutés à cette table.

Il peut aussi bien s'agir d'une table vide, que de la recopie d'une table existante.

La table servant de base pour les diagrammes prédéfinis est sélectionnée par la propriété `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"

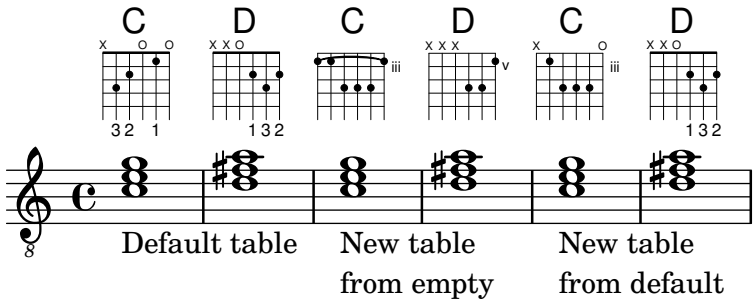
% Make a blank new fretboard table
#(define custom-fretboard-table-one
   (make-fretboard-table))
```

```
% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two
   (make-fretboard-table default-fret-table))

% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode {c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode {c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-one
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-two
      c1 | d1 |
    }
  }
  \new Staff {
    \clef "treble_8"
    <<
      \chordmode {
        c1 | d1 |
        c1 | d1 |
        c1 | d1 |
      }
      {
        s1_\markup "Default table" | s1 |
        s1_\markup \column {"New table" "from empty"} | s1 |
        s1_\markup \column {"New table" "from default"} | s1 |
      }
    >>
  }
>>
```



Voir aussi

Manuel de notation : [Généralités sur le mode accords], page 445, [Tablatures automatiques], page 414, [Tablatures personnalisées], page 389, [Tablatures prédéfinies], page 403.

Fichiers d'initialisation : `ly/predefined-guitar-fretboards.ly`, `ly/predefined-guitar-ninth-fretboards.ly`, `ly/predefined-ukulele-fretboards.ly`, `ly/predefined-mandolin-fretboards.ly`.

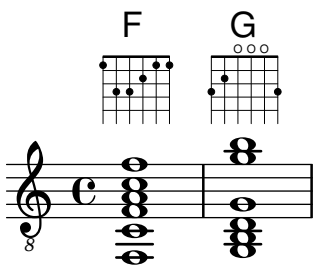
Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "fret-diagram-interface" dans *Référence des propriétés internes*.

Tablatures automatiques

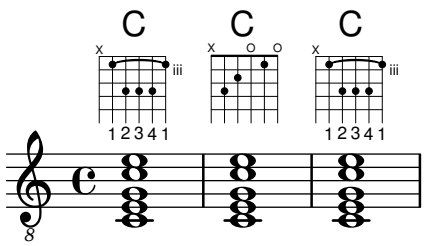
Les diagrammes d'accord peuvent être créés automatiquement ; il suffit d'affecter les notes à un contexte `FretBoards`. Si aucun diagramme prédéfini n'est disponible pour les notes entrées avec l'accordage actuel (`stringTunings`), les cordes et cases correspondant aux notes seront automatiquement calculées.

```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b' g'>1
  }
>>
```



Dans la mesure où aucun diagramme prédéfini n'est chargé par défaut, le calcul automatique des diagrammes d'accord est le comportement par défaut. Dès que les diagrammes par défaut sont chargés, le calcul automatique peut être activé ou désactivé par des commandes prédéfinies :

```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        "x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \new ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \new FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
    <c e g c' e'>1
  }
>>
```

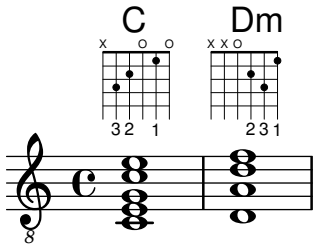


Le calculateur se trouvera parfois incapable de trouver un diagramme d'accord convenable. On peut souvent y remédier en assignant les notes aux cordes. Dans bien des cas, il suffit de placer manuellement une seule note pour que les autres soient alors placées de manière appropriée par le contexte `FretBoards`.

Il est possible d'ajouter des indications de doigté aux diagrammes de fret.

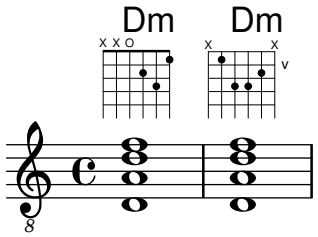
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
```

```
    <d a d' f'>1
  }
>>
```



La propriété `minimumFret` permet de définir la case minimale qui servira à calculer les cordes et les cases du contexte `FretBoard`.

```
<<
  \new ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \new FretBoards {
    <d a d' f'>1
    \set FretBoards.minimumFret = #5
    <d a d' f'>1
  }
  \new Staff {
    \clef "treble_8"
    <d a d' f'>1
    <d a d' f'>1
  }
>>
```



Les cordes et les cases du contexte `FretBoards` sont liées à la propriété `stringTunings`, qui a là même signification que dans le contexte `TabStaff`. Voir [Tablatures personnalisées], page 389, pour plus d'information sur la propriété `stringTunings`.

La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur au travers des propriétés de l'interface `fret-diagram-interface`. Pour un diagramme d'accord `FretBoards` donné, les propriétés de l'interface dépendent de `FretBoards.FretBoard`.

Commandes prédéfinies

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

Voir aussi

Manuel de notation : [Tablatures personnalisées], page 389.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

Référence des propriétés internes : Section “fret-diagram-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements
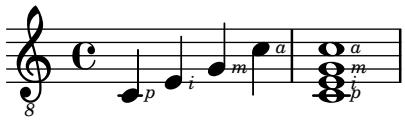
Le calcul automatique des diagrammes de fret se révèle inapproprié pour les instruments dont l'ordre des cordes ne correspond pas à l'ordre des hauteurs.

Doigtés pour la main droite

Les doigtés de main droite *p-i-m-a* doivent être entrés à l'aide de l'instruction `\rightHandFinger` suivie d'un nombre.

> **Note :** Lorsque vous utilisez l'instruction `\rightHandFinger` dans un accord, une espace **doit** précéder la fermeture de la construction par un `>`.

```
\clef "treble_8"
c4\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c'\rightHandFinger #4
<c\rightHandFinger #1 e\rightHandFinger #2
 g\rightHandFinger #3 c'\rightHandFinger #4 >1
```



Pour plus de clarté, vous pouvez traduire ou abréger la commande `\rightHandFinger`, par exemple en `doigtMainDroite` ou même `MD` en tête du fichier source :

```
MD = #rightHandFinger \etc
```

La plupart des comportements en matière de doigtés pour la main droite, à savoir l'objet `StrokeFinger`, est comparable aux doigtés ordinaires – voir [Doigtés], page 236.

Morceaux choisis

Positionnement des doigtés main droite

Le positionnement des doigtés main droite, grâce à une propriété spécifique, peut se contrôler finement, comme l'indique l'exemple suivant.

```
#(define RH rightHandFinger)

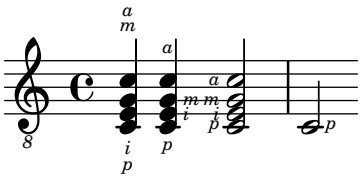
\relative c {
  \clef "treble_8"

  \set strokeFingerOrientations = #'(up down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4
```

```
  \set strokeFingerOrientations = #'(left)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >2

  \set strokeFingerOrientations = #'(right)
  c\RH #1
}
```

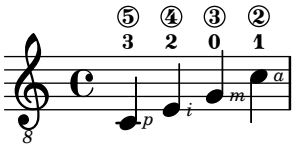


Doigtés, indications de corde, et doigtés main droite

L'exemple suivant illustre comment combiner des doigtés pour la main gauche, des indications de corde et des doigtés pour la main droite.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5\RH #1 >4
  <e-2\4\RH #2 >4
  <g-0\3\RH #3 >4
  <c-1\2\RH #4 >4
}
```



Voir aussi

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "StrokeFinger" dans *Référence des propriétés internes*.

2.4.2 Guitare

La plupart des aspects en matière de notation pour guitare sont traités dans la partie commune aux instruments frettés. Il subsiste cependant quelques particularités que nous allons maintenant examiner.
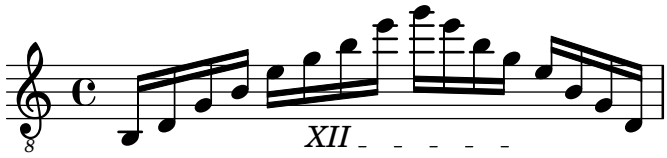
Parfois l'utilisateur aimerait créer des documents de type recueil de chansons, où l'on ne trouve que des accords au-dessus des paroles. Dans la mesure où LilyPond est un éditeur de partitions, il n'est pas l'outil optimal pour des documents sans partition. Une meilleure alternative serait de recourir à un traitement de texte, un éditeur de texte ou, pour les utilisateurs expérimentés, un logiciel typographique tel que GuitarTex.

Indication de la position et du barré

Cet exemple montre comment indiquer les positions et les barrés :

```
\relative {
```

```
  \clef "treble_8"
  b,16 d g b e
  \textSpannerDown
  \override TextSpanner.bound-details.left.text = "XII "
  g16\startTextSpan
  b16 e g e b g\stopTextSpan
  e16 b g d
}
```



Voir aussi
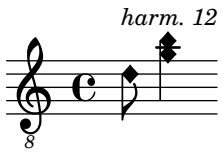
Manuel de notation : [Indication textuelle avec extension], page 257.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

Indication des harmoniques et notes étouffées

Des têtes de note spéciales peuvent servir à indiquer les notes étouffées et les sons harmoniques. Les sons harmoniques sont souvent détaillés grâce à des indications textuelles.

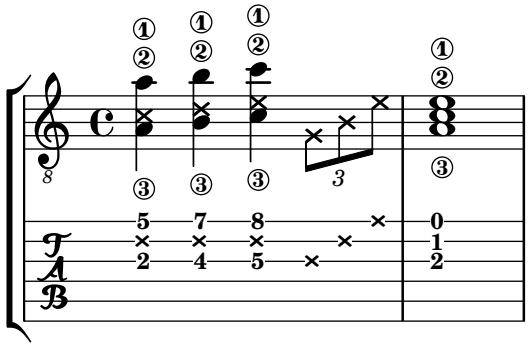
```
\relative {
  \clef "treble_8"
  \override NoteHead.style = #'harmonic-mixed
  d'8^\markup { \italic \fontsize #-2 "harm. 12" } <g b>4
}
```



Les notes étouffées, ou *notes fantômes*, se rencontrent aussi bien sur une portée normale que dans une tablature :

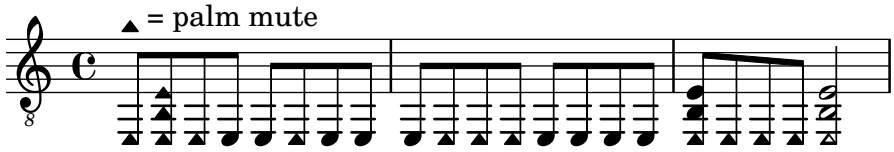
```
music = \relative {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \tuplet 3/2 { g8 b e }
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
  \new TabStaff {
```

```
    \music
  }
>>
```



Le *palm mute*, appelé aussi parfois *chop*, est une technique de jeu pour la guitare électrique ; elle est connue sous le nom de pizzicato par les joueurs de guitare classique. Elle consiste à poser la main droite sur les cordes juste au-dessus du chevalet, de façon à étouffer plus ou moins légèrement les notes. LilyPond permet d'indiquer ce style de jeu en affectant un profil spécifique aux têtes de note.

```
\new Voice { % Warning: explicit Voice instantiation is
             %    required to have palmMuteOff work properly
             %    when palmMuteOn comes at the beginning of
             %    the piece.
  \relative c, {
    \clef "G_8"
    \palmMuteOn
    e8^\markup { \musicglyph "noteheads.s2do"  = palm mute }
    < e b' e > e
    \palmMuteOff
    e e  \palmMute e e e |
    e8 \palmMute { e e e } e e e e |
    < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2
  }
}
```



Voir aussi

Manuel de notation : [Têtes de note spécifiques], page 40, Section 1.1.4 [Têtes de note], page 40.

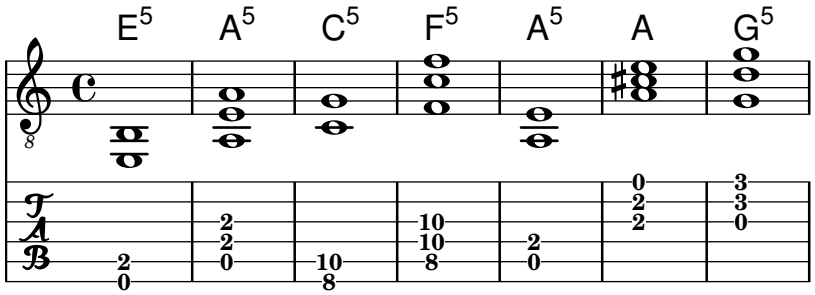
Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

Indication de *power chord*

Les *power chords* – terme anglais signifiant littéralement « accords de puissance » – s'indiquent aussi bien en mode accord que dans une construction en accord. La quinte est exceptionnellement indiquée dans le chiffrage de ces accords, contrairement aux accords habituels (parfaits ou non) :

```
ChordsAndSymbols = {
```

```
  \chordmode {
    e,,1:5
    a,,:5.8
    \set TabStaff.restrainOpenStrings = ##t
    \set minimumFret = #8
    c,:5
    f,:5.8
  }
  \set minimumFret = #2
  \set restrainOpenStrings = ##f
  <a, e> <a cis' e'>
  <g d' g'>
}
\score {
  <<
    \new ChordNames {
    \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “power chord” dans *Glossaire*.

Manuel de notation : [Extension et altération d’accords], page 447, [Impression des noms d’accord], page 450.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

2.4.3 Banjo

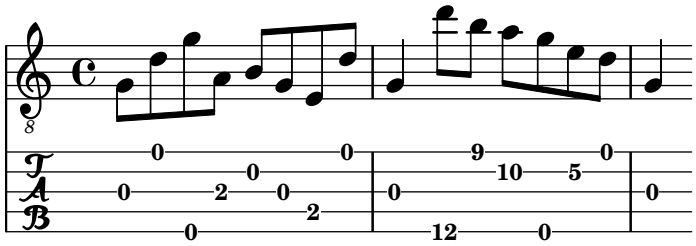
Tablatures pour banjo

LilyPond permet d’écrire des tablatures de base pour le banjo à cinq cordes. Pour ce faire, pensez à utiliser le format de tablature pour banjo, afin d’avoir le bon nombre de cordes et le bon accordage :

```
music = {
  g8 d' g'\5 a b g e d' |
```

```
  g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
  g4
}

<<
  \new Staff \with { \omit StringNumber }
  { \clef "treble_8"  \music }
  \new TabStaff \with {
    tablatureFormat = #fret-number-tablature-format-banjo
    stringTunings = #banjo-open-g-tuning
  }
  { \music }
>>
```



LilyPond prend en charge un certain nombre d'accordages courants pour banjo : `banjo-c-tuning` sol do sol si ré (gCGBD), `banjo-modal-tuning` sol ré sol do ré (gDGCD), `banjo-open-d-tuning` la ré fa# la ré (aDF#AD) et `banjo-open-dm-tuning` la ré fa la ré (aDFAD).

Ces accordages peuvent être convertis pour banjo à quatre cordes au moyen de la fonction `four-string-banjo` :

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Voir aussi

Fichiers d'initialisation : `ly/string-tunings-init.ly`.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

2.4.4 Luth

Tablatures pour luth

LilyPond prend en charge les tablatures pour le luth.

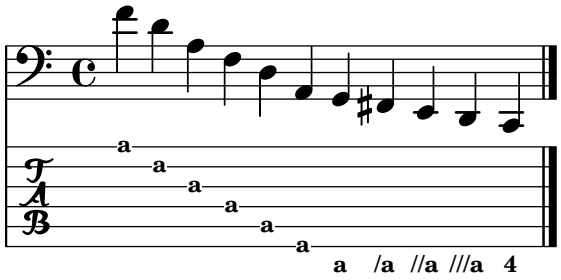
Les cordes supplémentaires s'ajoutent à l'aide de la commande `additionalBassStrings` qui permet de définir les hauteurs respectives de ces cordes. Elles viendront s'imprimer ainsi au-dessous de la ligne inférieure : a, /a, //a, ///a, 4, 5, etc.

`fret-letter-tablature-format`, et probablement `fretLabels`, fournis en argument à `tablatureFormat`, autoriseront des personnalisations plus avancées.

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  <<
    \new Staff { \clef bass \cadenzaOn  \m }
    \new TabStaff \m
  >>
  \layout {
    \context {
```

```
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



Problèmes connus et avertissements

L'utilisation de `FretBoards` avec des `additionalBassStrings` n'est pas prise en charge et conduira à un résultat insatisfaisant.

2.5 Percussions

2.5.1 Vue d'ensemble des percussions

La notation rythmique sert avant tout aux parties de percussions ou de batterie, mais on peut aussi s'en servir à des fins pédagogiques, pour montrer le rythme d'une mélodie.

Références en matière de notation pour percussions

- Certains instruments à percussion se notent sur une portée rythmique. Vous trouverez des informations détaillées à ce sujet aux rubriques [Gravure de lignes rythmiques], page 84, et [Initialisation de nouvelles portées], page 197.
- Le rendu MIDI des percussions fait l'objet d'une rubrique dédiée : Section 3.5 [Génération de fichiers MIDI], page 556.

Voir aussi

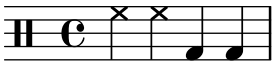
Manuel de notation : Section 3.5 [Génération de fichiers MIDI], page 556, [Gravure de lignes rythmiques], page 84, [Initialisation de nouvelles portées], page 197.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Notation de base pour percussions

Les parties de percussions peuvent être saisies avec le mode `\drummode`, qui est l'équivalent du mode standard utilisé pour les notes à hauteur déterminée. Le moyen plus simple pour saisir une partie de percussion est d'utiliser la commande `\drums`, qui crée un contexte spécifique :

```
\drums {
  hihat4 hh bassdrum bd
}
```



Il s'agit en fait d'un raccourci pour
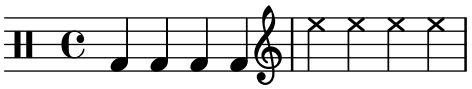
```
\new DrumStaff \drummode {
  hihat4 hh bassdrum bd
  }
```



Chaque instrument de percussion peut avoir, dans le fichier LilyPond, un nom complet et un nom raccourci. Ces noms sont inventoriés à l'annexe Section 3.5 [Génération de fichiers MIDI], page 556.

Notez bien que l'utilisation de hauteurs (comme un `cis4`) dans un contexte `DrumStaff` déclenchera immanquablement une erreur. Un contexte `DrumStaff` fait automatiquement appel à une clef spécifique ; vous pouvez la spécifier explicitement ou utiliser une autre clef.

```
\drums {
  \clef percussion
  bd4 4 4 4
  \clef treble
  hh4 4 4 4
}
```



Certains instruments à percussion connaissent quelques problèmes quant à leur prise en charge au niveau de la génération de fichiers MIDI ; de plus amples informations sont disponibles à la rubrique Section 3.5 [Génération de fichiers MIDI], page 556.

Voir aussi

Manuel de notation : Section 3.5 [Génération de fichiers MIDI], page 556, Section A.15 [Notes utilisées en percussion], page 807.
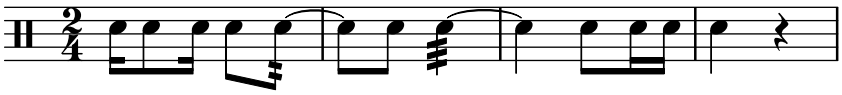
Fichiers d'initialisation : `ly/drumpitch-init.ly`

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Roulements de tambour
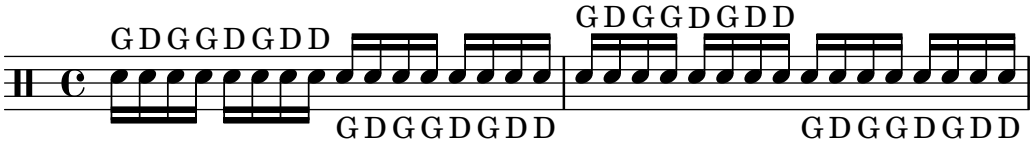
Les roulements de tambour s'indiquent par une triple barre en travers des hampes. Qu'il s'agisse d'une noire ou d'une durée plus longue, cette triple barre s'affiche explicitement. Dans le cas de croches, seules deux barres traversent les hampes (la troisième faisant office de ligature). Si ce roulement s'applique à des notes plus courtes que la croche, LilyPond fait apparaître une seule barre en travers des hampes en supplément du nombre de barres de ligature. Ces différents graphismes s'obtiennent à l'aide d'une notation de trémolo, en suivant les préceptes mentionnés à la rubrique [Répétitions en trémolo], page 170.

```
\drums {
  \time 2/4
  sn16 8 16 8 8:32 ~
  8 8 4:32 ~
  4 8 16 16
  4 r4
}
```



Les coups de baguette peuvent s'indiquer à l'aide de *markups* `"D"` et `"G"` au-dessus ou en dessous des notes comme indiqué à la rubrique Section 5.4.2 [Direction et positionnement], page 667. Vous devrez peut-être jouer sur la propriété `staff-padding` pour obtenir une ligne de base satisfaisante.

```
\drums {
  \repeat unfold 2 {
    sn16^"G" 16^"D" 16^"G" 16^"G" 16^"D" 16^"G" 16^"D" 16^"D"
    \stemUp
    sn16_"G" 16_"D" 16_"G" 16_"G" 16_"D" 16_"G" 16_"D" 16_"D"
  }
}
```



Voir aussi

Manuel de notation : Section 5.4.2 [Direction et positionnement], page 667, [Répétitions en trémolo], page 170.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Hauteurs en percussions

Certains instruments à percussion émettent des hauteurs, comme le xylophone, le vibraphone ou les timbales ; ils utilisent donc des portées classiques. Cette possibilité est abordée dans d'autres parties du manuel.

Voir aussi

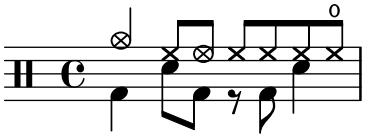
Manuel de notation : Section 3.5.5 [Gestion des instruments MIDI], page 562.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Portées de percussion

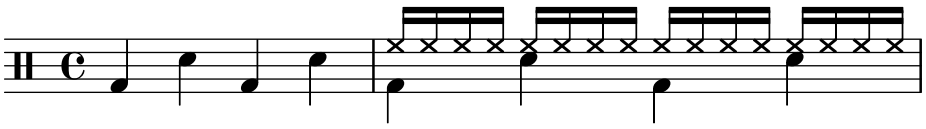
Une partie de percussions utilisant plusieurs instruments requiert en général une portée de plusieurs lignes, où chaque hauteur sur la portée représente un instrument à percussion. La gravure d'une telle musique requiert que les notes soient situées dans des contextes `DrumStaff` et `DrumVoice`.

```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



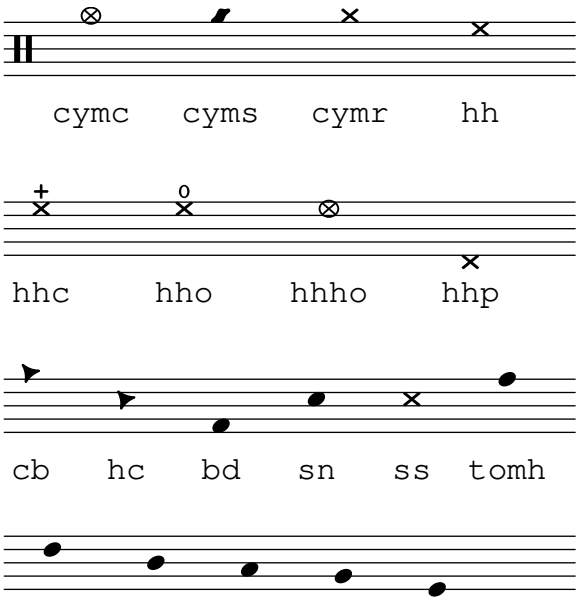
L'exemple ci-dessus montre une notation polyphonique détaillée. La notation polyphonique abrégée peut être employée – voir la rubrique Section "J'entends des Voix" dans *Manuel d'initiation* – comme ici :

```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
  }
>>
```



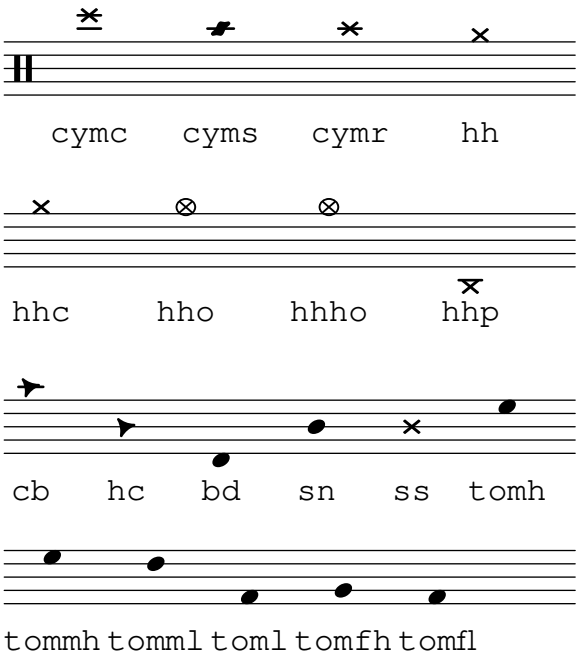
On peut choisir d'autres mises en forme si l'on définit la propriété `drumStyleTable` dans le contexte `DrumVoice`. Quelques variables prédéfinies sont disponibles :

`drums-style`
: La notation par défaut : une batterie standard, sur une portée de cinq lignes.

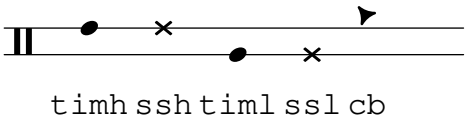


Le plan de la batterie peut inclure jusqu'à six toms différents. Bien sûr, vous n'êtes pas obligé de tous les utiliser si la musique en prévoit moins ; par exemple, les trois toms des lignes du milieu sont `tommh`, `tomml`, et `tomfh`.

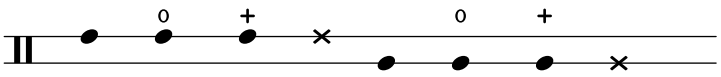
`agostini-drums-style`
: Inventée par le percussionniste français Dante Agostini en 1965, cette notation s'est répandue au-delà de l'hexagone.



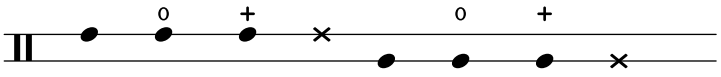
`timbales-style`
: Ce style permet de saisir une partie de timbales, sur une portée à deux lignes.



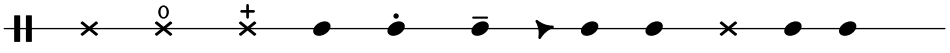
`congas-style`
: Ce style produit une portée à deux lignes pour une partie de congas.

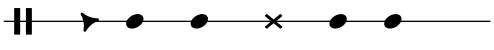


`bongos-style`
: Ce style produit une portée à deux lignes pour une partie de bongos.



`percussion-style`
: Ce style permet de saisir toute sorte de percussions sur des portées d'une ligne.





Il est par ailleurs possible de définir son propre style, comme indiqué dans [Personnalisation de portées de percussion], page 428.

Voir aussi

Manuel d'initiation : Section "J'entends des Voix" dans *Manuel d'initiation*.

Manuel de notation : [Personnalisation de portées de percussion], page 428.

Fichiers d'initialisation : `ly/drumpitch-init.ly`.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

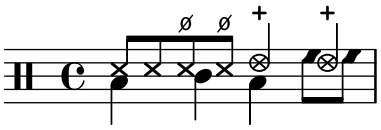
Personnalisation de portées de percussion

LilyPond offre la possibilité de construire son propre style de percussion grâce à une définition de la propriété `drumStyleTable`. Les notations existantes peut se redéfinir au travers d'une liste associative dans laquelle chaque entrée doit comporter quatre éléments : un nom, le style de tête de note (ou `default`), un éventuel signe d'articulation (ou `#f` dans le cas contraire) et, enfin, le positionnement de la tête de note sur la portée. Cette liste devra alors être convertie en table de hachage à l'aide de la fonction `alist->hash-table`.

```
#(define mydrums '(
         (bassdrum        default   #f           -1)
         (snare           default   #f           0)
         (hihat           cross     #f           1)
         (halfopenhihat   cross     "halfopen"   1)
         (pedalhihat      xcircle   "stopped"    2)
         (lowtom          diamond   #f           3)))

up = \drummode { hh8 hh hhho hhho hhp4 hhp }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



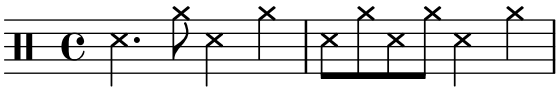
De nouveaux noms peuvent venir s'ajouter à ces notations personnalisées grâce à la variable `drumPitchNames` qui peut être redéfinie en tant que liste associative (ou augmentée par ajout d'une nouvelle liste aux valeurs existantes comme indiqué ci-dessous). Ceci permet par ailleurs de définir des alias : raccourcis pour la saisie de certaines notations.

```
drumPitchNames =
  #(append
    '((leftsnap . sidestick)
      (rightsnap . ridecymbal))
    drumPitchNames)

drumPitchNames.ls = #'sidestick
drumPitchNames.rs = #'ridecymbal

\drums {
  leftsnap4. rightsnap8 leftsnap4 rightsnap
  ls8 rs ls rs ls4 rs
```

```
}
```



De la même manière, la propriété `drumPitchTable` associe une hauteur spécifique (autrement dit un son instrumental différent tel que fourni par les fontes sonores MIDI disponibles) à chaque notation. Cette propriété requiert d'être définie sous forme de table de hachage, convertie elle aussi à partir d'une liste associative (enregistrée par défaut en tant que variable `midiDrumPitches`). La redéfinition de ces associations s'effectue comme indiqué ci-dessus, soit en créant l'intégralité d'une liste associative, soit certains de ses membres. L'exemple ci-dessous démontre la manière de créer un jeu entier de notation, avec sa propre syntaxe de saisie, sa notation personnalisée et les instruments MIDI correspondants.

```
drumPitchNames.dbass      = #'dbass
drumPitchNames.dba        = #'dbass  % 'db is in use already
drumPitchNames.dbassmute  = #'dbassmute
drumPitchNames.dbm        = #'dbassmute
drumPitchNames.do         = #'dopen
drumPitchNames.dopenmute  = #'dopenmute
drumPitchNames.dom        = #'dopenmute
drumPitchNames.dslap      = #'dslap
drumPitchNames.ds         = #'dslap
drumPitchNames.dslapmute  = #'dslapmute
drumPitchNames.dsm        = #'dslapmute
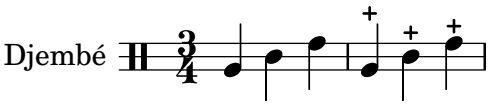
#(define djembe-style
  '((dbass      default  #f         -2)
    (dbassmute  default  "stopped"  -2)
    (dopen      default  #f          0)
    (dopenmute  default  "stopped"   0)
    (dslap      default  #f          2)
    (dslapmute  default  "stopped"   2)))

midiDrumPitches.dbass     = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen     = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap     = b
midiDrumPitches.dslapmute = ais

test = \drummode { dba4 do ds dbm dom dsm }

\score {
  \new DrumStaff \with {
    \override StaffSymbol.line-count = #3
    instrumentName = "Djembé "
    drumStyleTable = #(alist->hash-table djembe-style)
    drumPitchTable = #(alist->hash-table midiDrumPitches)
  } {
    \time 3/4
    \test
```

```
  }
  \layout {}
  \midi {}
}
```



Voir aussi

Fichiers d'initialisation : `ly/drumpitch-init.ly`.

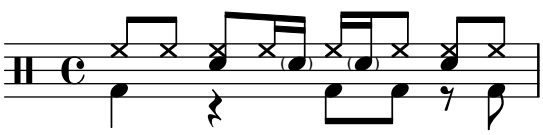
Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Référence des propriétés internes : Section "DrumStaff" dans *Référence des propriétés internes*, Section "DrumVoice" dans *Référence des propriétés internes*.

Notes fantômes

Des notes fantômes, qu'on appelle aussi en anglais *dead, muted, silenced ou false* notes, peuvent être créées pour les parties de percussion, grâce à la commande `\parenthesize` décrite à la rubrique [Parenthèses], page 244.

```
\new DrumStaff
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ 8] <hh sn> hh16
        \parenthesize sn hh
        \parenthesize sn hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 8 r8 bd
      }
    >>
  }
>>
```

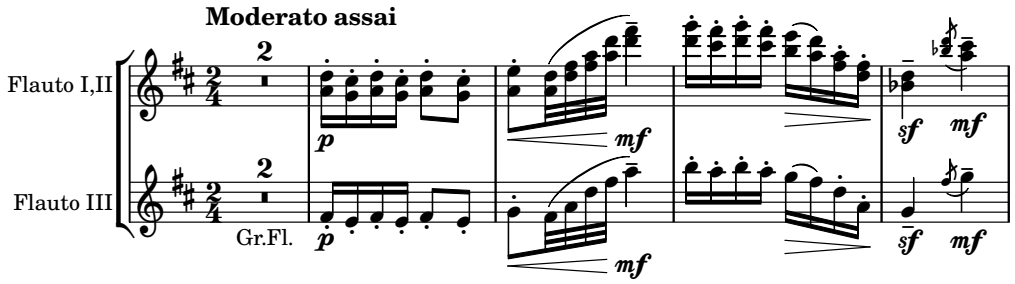


Voir aussi

Manuel de notation : [Parenthèses], page 244.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

2.6 Instruments à vent



Ce chapitre traite de certains aspects particuliers en matière de notation pour instruments à vent.

2.6.1 Vue d'ensemble des instruments à vent

Nous allons aborder ici quelques aspects communs à la plupart des instruments à vent.

Références en matière d'instruments à vent

Ce qui caractérise les partitions pour instruments à vent a trait principalement à la respiration et à l'attaque :

- Les respirations s'indiquent par des silences ou des [Signes de respiration], page 143.
- Un jeu *legato* s'indique par des [Liaisons d'articulation], page 139.
- Les différents types d'attaque – *legato*, détaché ou piqué – s'indiquent en principe par des signes d'articulation, parfois agrémentés de liaisons. Voir à ce sujet [Articulations et ornements], page 127, et Section A.14 [Liste des signes d'articulation], page 805.
- Un *Flatterzunge* (trémolo dental ou trémolo en roulant les r) s'indique par une marque de trémolo et une étiquette textuelle attachée à la note concernée. Voir à ce sujet [Répétitions en trémolo], page 170.

D'autres aspects de la notation s'appliquent aussi aux instruments à vent :

- De nombreux instruments à vent sont transpositeurs ; voir [Instruments transpositeurs], page 28.
- Les glissandos sont l'une des caractéristiques du trombone à coulisse, bien que d'autres instruments puissent y parvenir en jouant sur les pistons ou des clés ; consulter [Glissando], page 146.
- Des glissandos harmoniques sont réalisables par les cuivres. Ils sont traditionnellement indiqués par des [Notes d'ornement], page 119.
- Les inflexions en fin de note sont abordées au chapitre [Chutes et sauts], page 145.
- Les « bruitage » de clé ou de piston s'indiquent souvent par le style `cross` ou des [Têtes de note spécifiques], page 40.
- Les bois peuvent émettre des harmoniques dans le bas de leur registre. On les indique avec un `flageolet` – voir Section A.14 [Liste des signes d'articulation], page 805.
- En ce qui concerne les cuivres, la sourdine s'indique en principe par une étiquette textuelle. Cependant, lorsque les changements sont nombreux et rapides, il est d'usage de recourir aux articulations `stopped` et `open`. Pour de plus amples détails, voir [Articulations et ornements], page 127, et Section A.14 [Liste des signes d'articulation], page 805.
- La sourdine du cor d'harmonie s'indique par un `stopped`. Voir le chapitre [Articulations et ornements], page 127.

Morceaux choisis

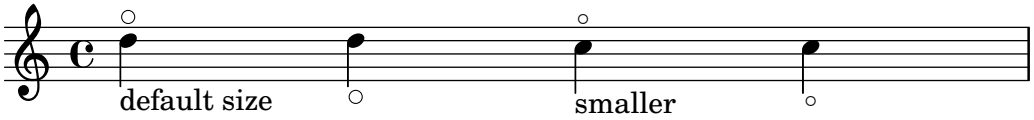
Modifier la taille d'un \flageolet

Il est possible de rapetisser le cercle d'un `\flageolet` grâce à un `\tweak`.

```
smallFlageolet = \tweak font-size -3 \flageolet

\layout { ragged-right = ##f }

\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



Voir aussi

Manuel de notation : [Articulations et ornements], page 127, [Chutes et sauts], page 145, [Glissando], page 146, [Instruments transpositeurs], page 28, [Liaisons d'articulation], page 139, Section A.14 [Liste des signes d'articulation], page 805, [Notes d'ornement], page 119, [Répétitions en trémolo], page 170, [Signes de respiration], page 143, [Têtes de note spécifiques], page 40.

Morceaux choisis : Section "Vents" dans *Morceaux choisis*.

Doigtés pour vents

Tout instrument à vent, hormis le trombone à coulisse, fait appel à plusieurs doigts pour produire un son. Les exemples ci-dessous vous donnent un aperçu de différentes manières d'indiquer des doigtés.

La gestion des diagrammes de doigté spécifiques aux bois est abordée plus en profondeur au chapitre Section 2.6.3.1 [Diagrammes pour bois], page 436.

Morceaux choisis

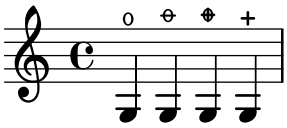
Symboles de doigtés pour instruments à vent

Des symboles spécifiques peuvent être obtenus en combinant les glyphes disponibles, ce qui est tout à fait indiqué en matière d'instrument à vent.

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
  (+ (ly:self-alignment-interface::centered-on-x-parent g)
     (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \relative c'{
    g\open
    \once \override TextScript.staff-padding = #-1.0
    \centermarkup
    g^\markup {
      \combine
```

```
        \musicglyph "scripts.open"
        \musicglyph "scripts.tenuto"
    }
    \centermarkup
    g^\markup {
      \combine
        \musicglyph "scripts.open"
        \musicglyph "scripts.stopped"
    }
    g\stopped
  }
}
```



Doigtés pour flûte à bec

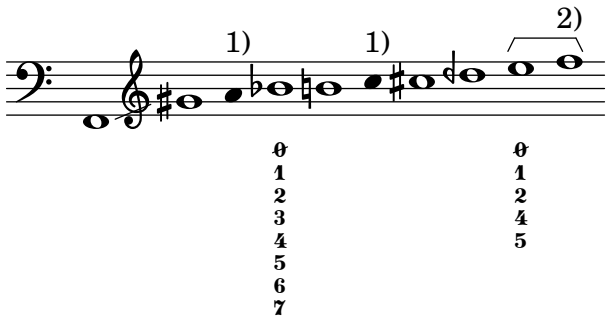
Cet exemple illustre la manière de créer et afficher des indications de doigté pour instrument à vent.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset = #(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
       (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \omit Stem
    \omit Flag
    \consists "Horizontal_bracket_engraver"
  }
  {
    \clef bass
    \set Score.timing = ##f
    f,1*1/4 \glissando
    \clef violin
    gis'1*1/4
    \stemDown a'4^\markup {1)}
    \centermarkup
    \once \override TextScript.padding = #2
    bes'1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 3 \finger 4 \finger 5 \finger 6 \finger 7} }
    b'1*1/4
    c''4^\markup {1)}
    \centermarkup
```

```
    \once \override TextScript.padding = #2
    cis''1*1/4
    deh''1*1/4
    \centermarkup
    \once \override TextScript.padding = #2
    \once \override Staff.HorizontalBracket.direction = #UP
    e''1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 4 \finger 5} }\startGroup
    f''1*1/4^\markup {2)}\stopGroup
  }
}
```



Voir aussi

Manuel de notation : Section 2.6.3.1 [Diagrammes pour bois], page 436.

Morceaux choisis : Section “Vents” dans *Morceaux choisis*.

2.6.2 Cornemuse

Voici quelques informations spécifiques à la cornemuse.

Définitions pour la cornemuse

LilyPond inclut des définitions spécifiques destinées à la notation pour cornemuse écossaise ; pour les utiliser, il suffit d'ajouter
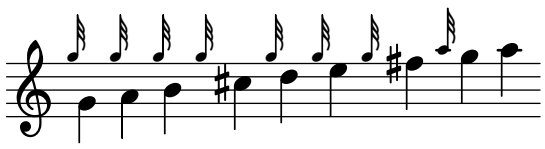
```
\include "bagpipe.ly"
```

en début de fichier. Vous bénéficierez ainsi de commandes courtes pour les appoggiatures spéciales et idiomatiques de la cornemuse. Par exemple, `\taor` est un raccourci pour

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` prend également en charge les définitions de hauteurs pour la cornemuse ; vous n'avez donc pas à vous soucier d'employer `\relative` ou `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



La musique pour cornemuse est traditionnellement écrite en ré majeur. Dans la mesure où c'est la seule tonalité possible, l'usage est de ne pas indiquer l'armure explicitement. À cet effet,

pensez à commencer votre partition par `\hideKeySignature` – ou bien `\showKeySignature` si, pour une raison ou pour une autre, vous tenez à afficher l'armure.

Des compositions actuelles peuvent ne pas respecter cette tonalité traditionnelle, auquel cas les do et fa devraient être abaissés en utilisant `c-flat` ou `f-flat` ; ils seront représentés par une note en forme de croix.

Lorsqu'on joue des œuvres moins cérémonieuses que pour une parade ou un défilé, peut survenir un sol aigu, au doux nom de « piobaireachd », et que l'on indiquera par `g-flat`.

Voir aussi

Morceaux choisis : Section "Vents" dans *Morceaux choisis*.

Exemple pour la cornemuse
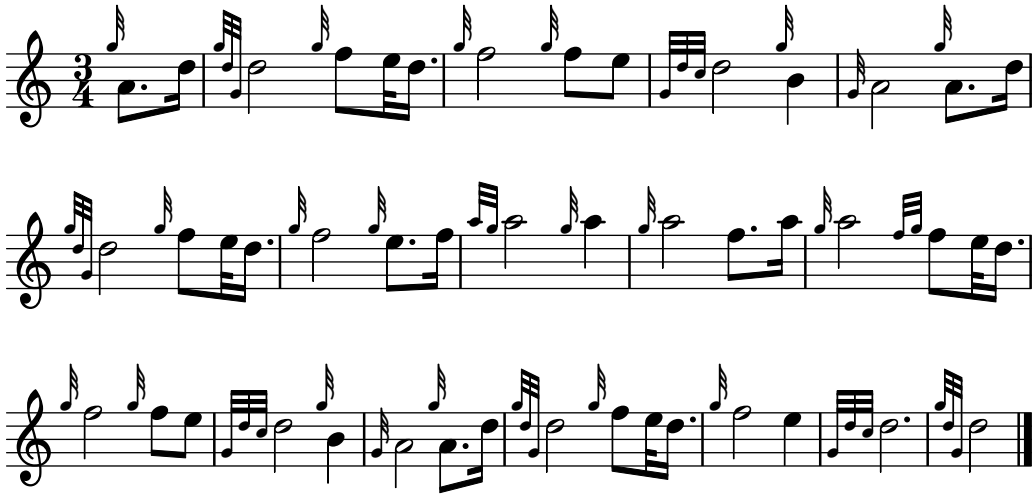
Et voici en guise d'exemple, à quoi ressemble le chant populaire *Amazing Grace*, noté dans l'idiome de la cornemuse.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
  \context { \Score \remove "Bar_number_engraver" }
}

\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{
  \hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrwd d2.
  \slurd d2
  \bar "|."
}
```





Voir aussi

Morceaux choisis : Section “Vents” dans *Morceaux choisis*

2.6.3 Bois

Nous allons ici nous intéresser aux spécificités de la section des bois.

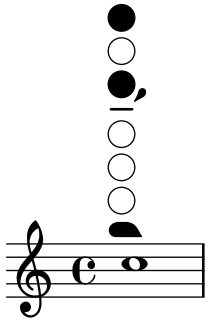
2.6.3.1 Diagrammes pour bois

Les doigtés pour obtenir une note particulière peuvent s’afficher sous forme graphique. LilyPond dispose de diagrammes pour la plupart des bois, et tout particulièrement les instruments suivants :

- piccolo
- flûte
- hautbois
- clarinette
- clarinette basse
- saxophone
- basson
- contrebasson

Les diagrammes sont générés en tant qu’objet de type *markup* :

```
c''1^\markup
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
```



Les clés ou trous peuvent être partiellement enfoncés ou bouchés :

```
\textLengthOn
```

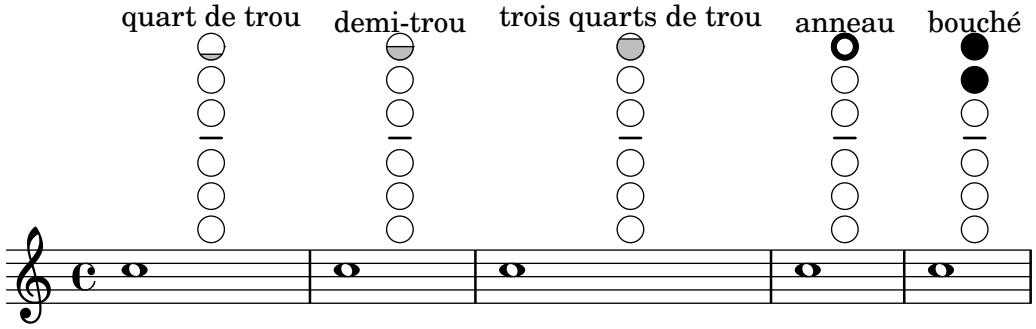
```
c''1^\markup {
  \center-column {
   "quart de trou"
   \woodwind-diagram #'flute #'((cc . (one1q))
                                (lh . ())
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "demi-trou"
   \woodwind-diagram #'flute #'((cc . (one1h))
                                (lh . ())
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "trois quarts de trou"
   \woodwind-diagram #'flute #'((cc . (one3q))
                                (lh . ())
                                (rh . ()))
  }
}

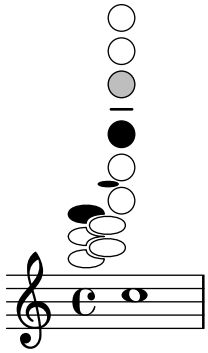
c''1^\markup {
  \center-column {
   "anneau"
   \woodwind-diagram #'flute #'((cc . (oneR))
                                (lh . ())
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "bouché"
   \woodwind-diagram #'flute #'((cc . (oneF two))
                                (lh . ())
                                (rh . ()))
  }
}
```



L'indication du doigté permettant de triller s'obtient en grisant une position :

```
c''1^\markup {
  \woodwind-diagram #'bass-clarinet
                    #'((cc . (threeT four))
                       (lh . ())
                       (rh . (b fis)))
}
```



Certaines combinaisons particulières en matière de trille sont possibles :

```
\textLengthOn
c''1^\markup {
  \center-column {
   "quart de trou et anneau"
   \woodwind-diagram #'flute #'((cc . (one1qTR))
                                (lh . ())
                                (rh . ()))
  }
}

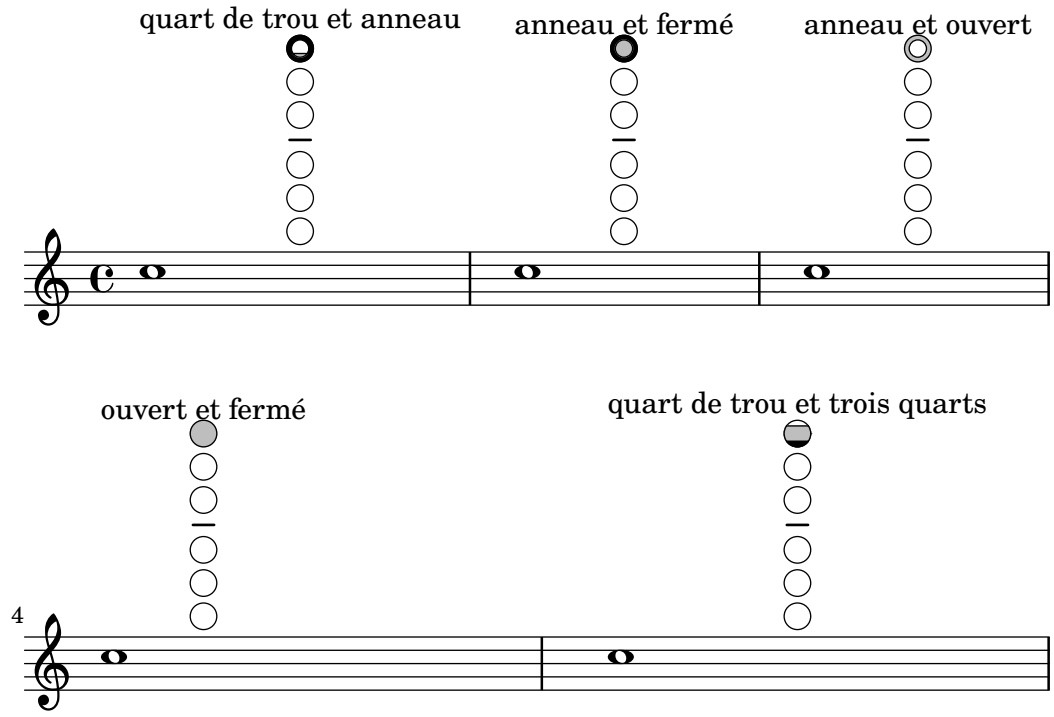
c''1^\markup {
  \center-column {
   "anneau et fermé"
   \woodwind-diagram #'flute #'((cc . (oneTR))
                                (lh . ())
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "anneau et ouvert"
   \woodwind-diagram #'flute #'((cc . (oneRT))
                                (lh . ())
```

```
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "ouvert et fermé"
   \woodwind-diagram #'flute #'((cc . (oneT))
                                (lh . ())
                                (rh . ()))
  }
}

c''1^\markup {
  \center-column {
   "quart de trou et trois quarts"
   \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                (lh . ())
                                (rh . ()))
  }
}
```



Bien que cela ne produise pas de notation, vous pouvez obtenir la liste de toutes les possibilités pour un instrument donné, en utilisant dans un fichier les instructions `#(print-keys-verbose '`*vent*`)` – affichage à l'écran – ou `#(print-keys-verbose '`*vent*` (current-error-port))` – génération d'un fichier de journalisation.

De nouveaux diagrammes sont réalisables, bien que ceci requiert de maîtriser le langage Scheme et n'est pas à la portée de tous les utilisateurs. Des gabarits sont contenus dans les fichiers `scm/define-woodwind-diagrams.scm` et `scm/display-woodwind-diagrams.scm`.

Morceaux choisis

Liste des diagrammes de doigtés pour bois

Voici les différents instruments à vent de la section des bois pour lesquels LilyPond peut, à ce jour, afficher des doigtés.

```
\layout {
  indent = 0
}

\relative c' {
  \textLengthOn
  c1^
  \markup {
    \center-column {
      'tin-whistle
      " "
       \woodwind-diagram
                  #'tin-whistle
                  #'()
    }
  }

  c1^
  \markup {
    \center-column {
      'piccolo
      " "
       \woodwind-diagram
                  #'piccolo
                  #'()
    }
  }

  c1^
  \markup {
    \center-column {
       'flute
       " "
       \woodwind-diagram
          #'flute
          #'()
    }
  }
  c1^\markup {
    \center-column {
      'oboe
      " "
      \woodwind-diagram
        #'oboe
        #'()
    }
  }
```

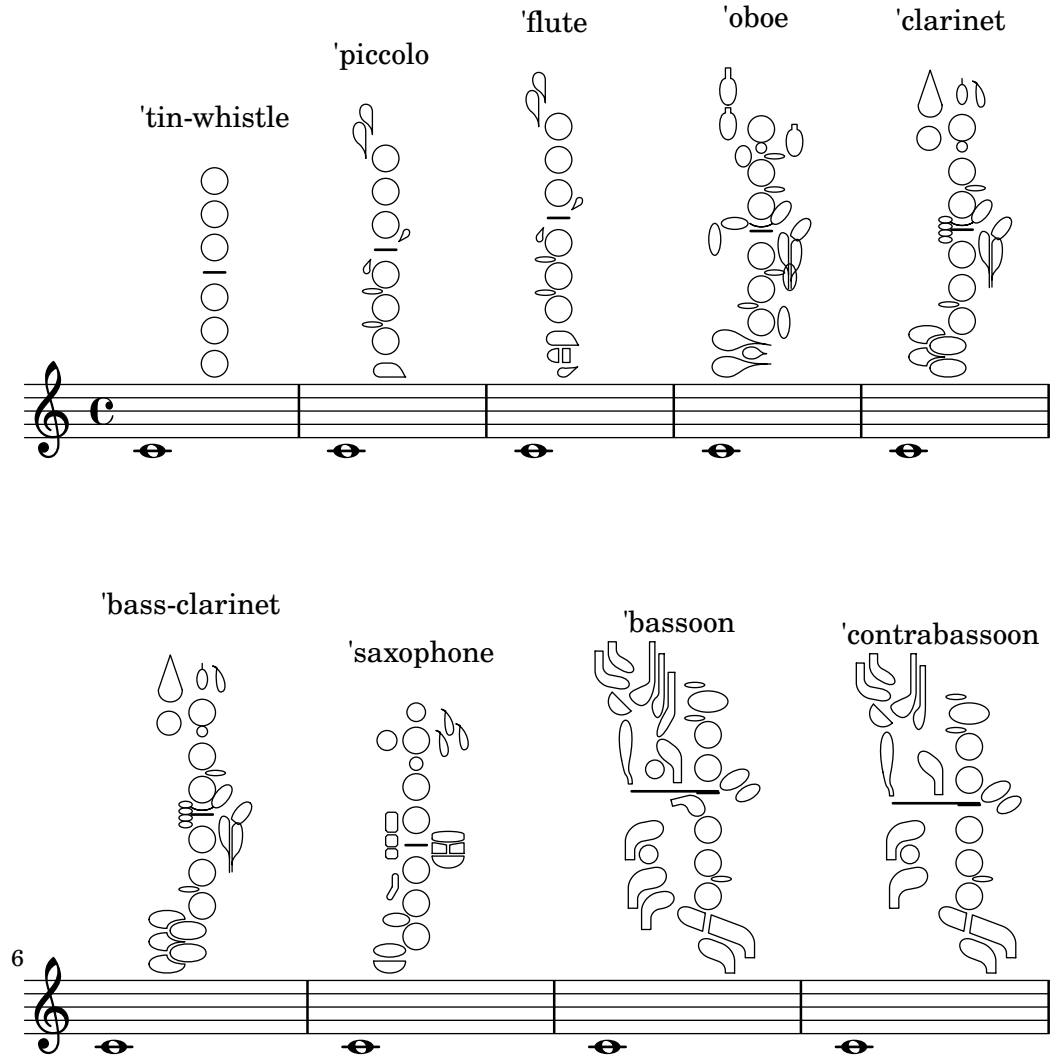
```
    c1^\markup {
      \center-column {
        'clarinet
        " "
        \woodwind-diagram
          #'clarinet
          #'()
      }
    }

    c1^\markup {
      \center-column {
        'bass-clarinet
        " "
        \woodwind-diagram
          #'bass-clarinet
          #'()
      }
    }

    c1^\markup {
      \center-column {
        'saxophone
        " "
        \woodwind-diagram
          #'saxophone
          #'()
      }
    }

    c1^\markup {
      \center-column {
        'bassoon
        " "
        \woodwind-diagram
          #'bassoon
          #'()
      }
    }

    c1^\markup {
      \center-column {
        'contrabassoon
        " "
        \woodwind-diagram
          #'contrabassoon
          #'()
      }
    }
  }
```

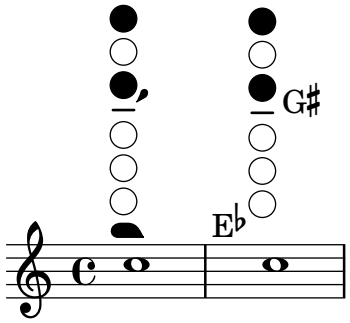


Ajout de texte à un diagramme de doigté

Dans certains cas, vous pouvez opter pour l'affichage textuel d'une clé située à côté d'un trou plutôt que sa représentation graphique.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
      #'((cc . (one three))
         (lh . (gis))
         (rh . (ees)))

  c^\markup
    \override #'(graphical . #f) {
      \woodwind-diagram
        #'piccolo
        #'((cc . (one three))
           (lh . (gis))
           (rh . (ees)))
    }
}
```

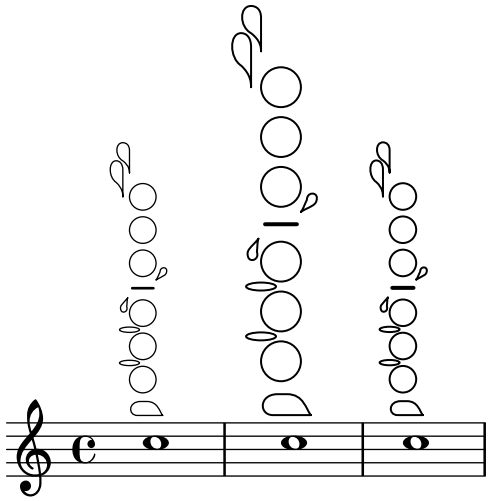


Modification de la taille d'un diagramme pour bois

La taille et l'épaisseur des diagrammes de doigté pour bois est modifiable à souhait.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
      #'()

  c^\markup
    \override #'(size . 1.5) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
  c^\markup
    \override #'(thickness . 0.15) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
}
```
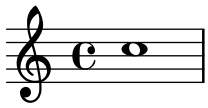


Liste des différents diagrammes de doigtés pour bois

Le code suivant permet d'obtenir une liste de toutes les possibilités en matière de doigtés pour bois, tels qu'ils sont définis dans le fichier `scm/define-woodwind-diagrams.scm`. Cette liste sera produite dans le fichier de journalisation, mais pas sous forme de musique. Pour un affichage en console, supprimez la partie `(current-error-port)` des commandes.

```
#(print-keys-verbose 'piccolo (current-error-port))
```

```
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))

\score {c''1}
```



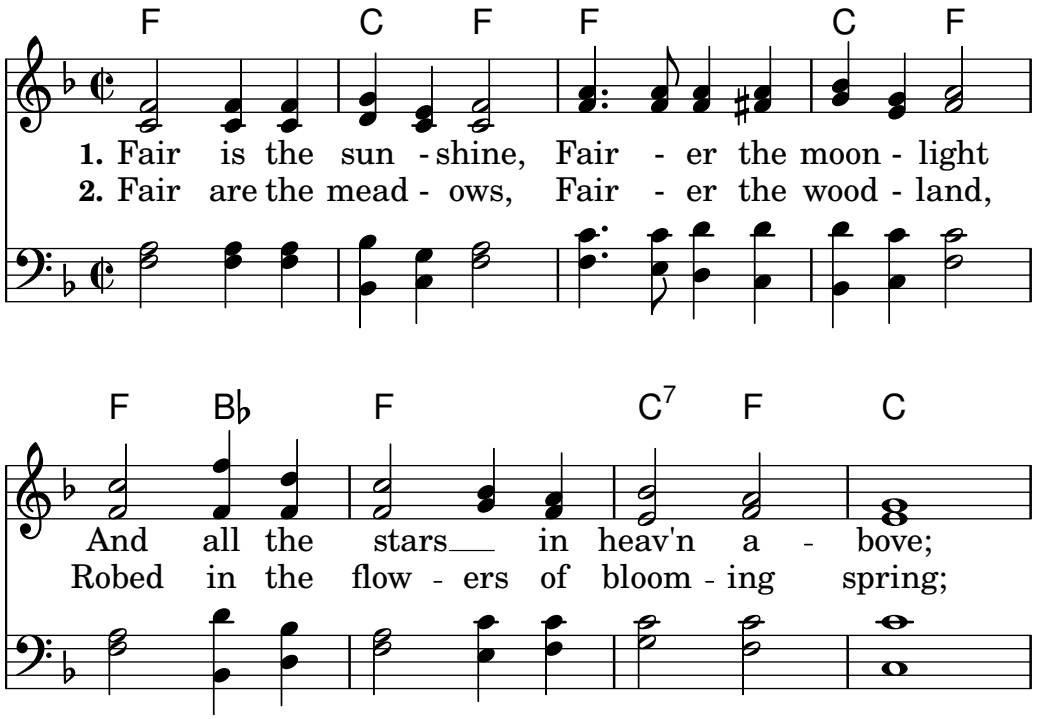
Voir aussi

Fichiers d'initialisation : `scm/define-woodwind-diagrams.scm`, `scm/display-woodwind-diagrams.scm`.

Morceaux choisis : Section "Vents" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*, Section "instrument-specific-markup-interface" dans *Référence des propriétés internes*.

2.7 Notation des accords



Les accords se saisissent soit comme des notes normales, soit à l'aide d'un mode dédié ; ils seront rendus selon l'une des diverses nomenclatures occidentales. Les accords peuvent aussi se présenter sous forme nominale, ou bien en basse figurée.

2.7.1 Mode accords

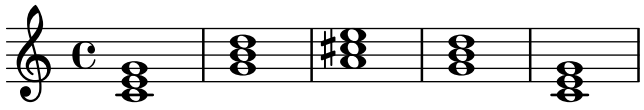
Le mode accords permet de saisir des accords en indiquant leur structure plutôt que les notes qui les composent.

Généralités sur le mode accords

Un accord peut se saisir en tant que musique simultanée, comme nous l'avons vu à la rubrique [Notes en accords], page 173.

LilyPond dispose aussi d'un « mode accords » au sein duquel sera considérée la structure des accords, tels qu'ils se présentent dans les traditions occidentales, plutôt que les différentes hauteurs qui les composent. Ce mode est tout à fait adapté pour ceux qui sont plus habitués aux accords nommés. Pour plus d'information quant aux différentes façons de libeller votre code, voir Section 5.4.1 [Modes de saisie], page 666.
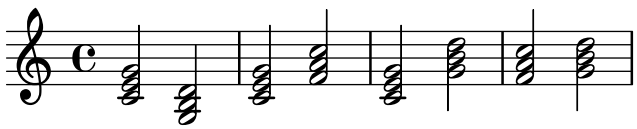
```
\chordmode { c1 g a g c }
```



Tout accord saisi dans ce mode dédié constitue un élément musical à part entière ; il pourra donc par exemple être transposé comme n'importe quel ensemble de hauteurs simultanées. `\chordmode` travaille en absolu ; une instruction `\relative` restera sans effet au sein d'un bloc `chordmode`. Notez toutefois que les hauteurs absolues sont une octave plus haut en `\chordmode` qu'en mode notes traditionnel.

Mode notes et mode accords peuvent tout à fait cohabiter dans une séquence musicale :

```
\relative {
  <c' e g>2 <g b d>
  \chordmode { c2 f }
  <c e g>2 <g' b d>
  \chordmode { f2 g }
}
```



Voir aussi

Glossaire musicologique : Section "Accord" dans *Glossaire*.

Manuel de notation : Section 5.4.1 [Modes de saisie], page 666, [Notes en accords], page 173.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

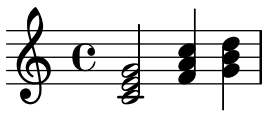
Problèmes connus et avertissements

Les raccourcis d'articulation ou d'ornementation ne sont pas disponibles en mode accord – voir [Articulations et ornements], page 127.

Accords courants
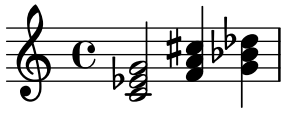
Dans le mode accords, introduit par la commande `\chordmode`, les accords ne sont indiqués que par leur note fondamentale, à laquelle on peut adjoindre une durée.

```
\chordmode { c2 f4 g }
```



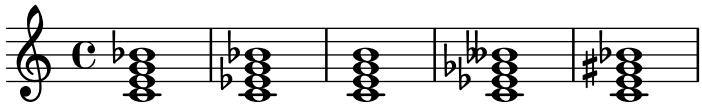
On peut cependant obtenir des accords différents – mineur, augmenté ou diminué – en utilisant le caractère deux points (`:`).

```
\chordmode { c2:m f4:aug g:dim }
```

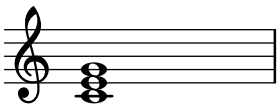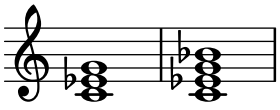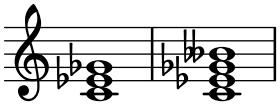


Les accords de septième sont aisément stipulables :

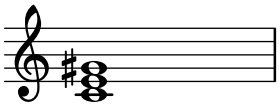
```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



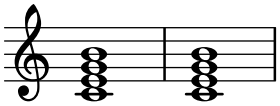
Voici les différents modificateurs d'accord de trois sons ou de septième. Par défaut, la septième ajoutée sera mineure ; la septième de dominante est donc l'accord de septième de base. Toutes les altérations sont relatives à la septième de dominante. Une table étendue des modificateurs et de leur utilisation est à l'annexe Section A.2 [Modificateurs d'accord usuels], page 703.

Modificateur	**Action**	**Exemple**
Néant	Action par défaut ; produit une triade majeure.	
m, m7	Accord mineur. Ce modificateur abaisse la tierce, et la septième s'il y en a une.	
dim, dim7	Accord diminué. Ce modificateur minorise la tierce, diminue la quinte et la septième s'il y en a.	

aug — Accord augmenté. Ce modificateur augmente la quinte.



maj, maj7 — Accord de septième majeure. Ce modificateur majorise la septième. Le `7` à la suite du `maj` est facultatif. Ce modificateur ne sert pas à créer une triade majeure.



Voir aussi

Manuel de notation : [Extension et altération d'accords], page 447, Section A.2 [Modificateurs d'accord usuels], page 703.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

Problèmes connus et avertissements

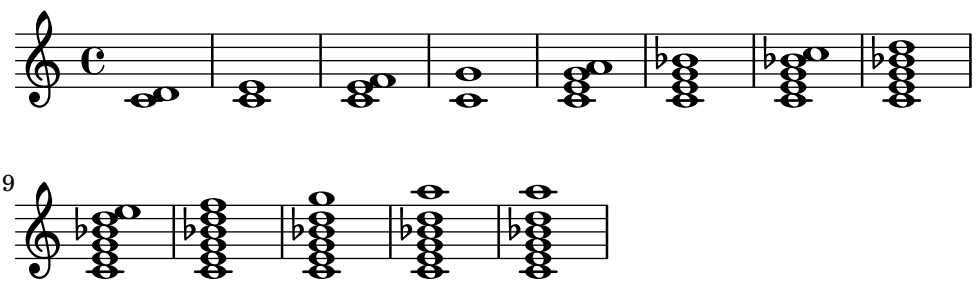
Un accord ne devrait comporter qu'un seul modificateur de qualité. La présence de plusieurs modificateurs ne déclenchera ni avertissement, ni erreur, mais le résultat pourrait être surprenant. Un accord qui n'est pas constructible à l'aide d'un unique modificateur devra faire l'objet d'une altération de ses composantes, comme indiqué à la rubrique [Extension et altération d'accords], page 447.

Extension et altération d'accords

Le mode accords permet d'élaborer des accords complexes. Ainsi, on peut enrichir l'accord par des notes ajoutées, enlever certaines notes, augmenter ou diminuer certains intervalles, ajouter la note de basse ou créer un renversement.

Le premier nombre qui suit le caractère deux-points (`:`) permet de déterminer l'étendue d'un accord. L'accord sera construit par ajout à la fondamentale, d'autant de tierces consécutives que nécessaire pour atteindre le nombre spécifié. N'oubliez pas que la septième ajoutée à un accord est minorée par défaut. Lorsque le dernier degré ne correspond pas à une tierce – la sixte par exemple – les tierces seront empilées jusqu'à celle directement inférieure au degré mentionné, qui sera alors ajouté pour conclure l'accord. L'étendue maximale autorisée est la treizième ; toute étendue plus élevée sera interprétée comme un accord de treizième.

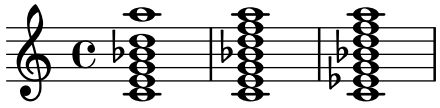
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



De manière exceptionnelle, `c:5` produit un *power chord*, accord formé de la fondamentale et la quinte.

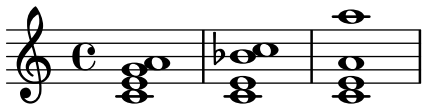
Dans la mesure où un accord de treizième majeure ne sonne pas très bien avec la onzième, la onzième est automatiquement enlevée, sauf à l'avoir explicitement spécifiée.

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



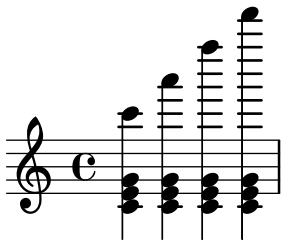
On peut enrichir l'accord par des notes ajoutées, que l'on indique après le chiffrage principal et que l'on sépare par des points, sans oublier que si l'on y ajoute une septième, celle-ci sera minorée et non majeure.

```
\chordmode {
  c1:3.5.6 c:3.7.8 c:3.6.13
}
```
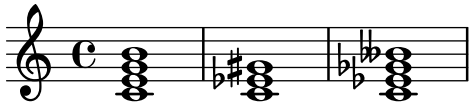


Les notes ajoutées peuvent monter aussi haut que de besoin.

```
\chordmode {
  c4:3.5.15 c:3.5.20 c:3.5.25 c:3.5.30
}
```



On peut augmenter ou diminuer certains intervalles au moyen des signes `-` ou `+` au degré considéré. L'altération de l'un des degrés automatiquement inclus dans la structure de base d'un accord s'effectue de la même manière.

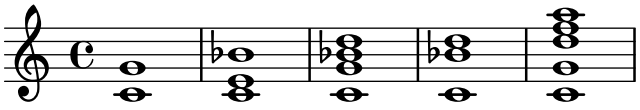
```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



Après avoir ajouté des notes à un accord, on peut aussi en enlever certaines, en les spécifiant derrière un signe `^` – les séparer par un point lorsqu'il y en a plus d'une.

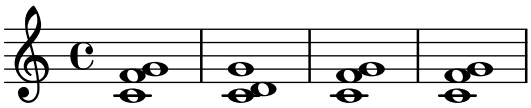
```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
```

```
}
```



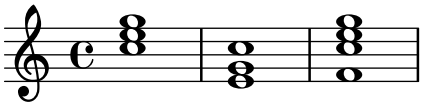
L'ajout du modificateur `sus` permet de créer un accord de suspension. Ceci aura pour effet de supprimer la tierce et d'y ajouter, suivant que vous spécifiez `2` ou `4`, la seconde ou la quarte ; `sus` est alors équivalent à `^3`. `sus4` est équivalent à `5.4`.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4
}
```



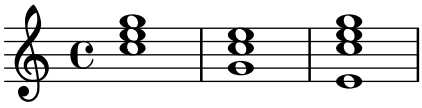
Les accords peuvent être combinés avec une note étrangère à la basse, au moyen de *accord/hauteur*.

```
\chordmode {
  c'1 c'/e c'/f
}
```



Si la note ajoutée appartient déjà à l'accord, la doublure supérieure sera supprimée, ce qui permet d'imprimer un renversement. Pour l'éviter, utilisez la syntaxe `/+`*hauteur*.

```
\chordmode {
  c'1 c'/g c'/+e
}
```



L'automatisation des renversements et la vocification sont abordées dans [Renversement d'accord et vocification spécifique], page 450.

Une table étendue des modificateurs et de leur utilisation est à l'annexe Section A.2 [Modificateurs d'accord usuels], page 703.

Voir aussi
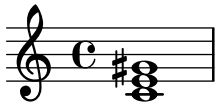
Manuel de notation : Section A.2 [Modificateurs d'accord usuels], page 703, [Renversement d'accord et vocification spécifique], page 450.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

Problèmes connus et avertissements

Aucun nom de note ne peut être indiqué deux fois dans un accord. Ainsi, dans l'accord suivant, seule la quinte augmentée est prise en compte, parce qu'elle est indiquée en dernier :

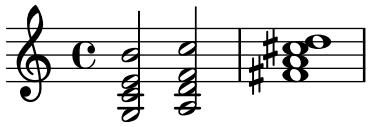
```
\chordmode { c1:3.5.5-.5+ }
```



Renversement d'accord et vocification spécifique

Les modifications d'accord ou l'ajout d'une basse sont complétées par diverses fonctions permettant d'imprimer automatiquement des renversements ou des vocifications particulières, à l'instar du *drop 2* couramment utilisé en jazz.

```
\chordmode {
  \dropNote 2 {
    c2:maj7 d:m7
  }
  \invertChords 1 d1:maj7
}
```



Contrairement à l'ajout de basses comme indiqué dans [Extension et altération d'accords], page 447, ceci n'interfère que sur la façon dont les accords seront imprimés sur une portée, et aucunement sur leur représentation lettrée. Par ailleurs, l'utilisation de ces fonctions ne se limite pas seulement au mode accords ; elles sont aussi disponibles dans une construction d'accord avec `<...>` comme indiqué dans [Notes en accords], page 173.

Voir aussi

Manuel de notation : [Extension et altération d'accords], page 447, [Notes en accords], page 173.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

2.7.2 Gravure des accords

Les accords peuvent se présenter aussi bien sous forme nominative que comme un empilement de notes sur une portée.

Impression des noms d'accord

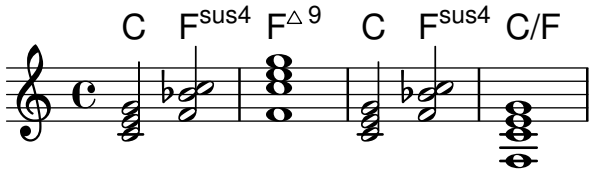
Les chiffrages d'accords sont liés au contexte `ChordNames` :

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```
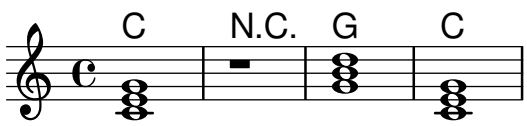


Les accords peuvent être saisis soit en tant que hauteurs simultanées, soit au moyen du mode accords. Le chiffrage affiché sera identique quel que soit le mode de saisie, à moins qu'il n'y ait inversion ou ajout de la basse.

```
chordmusic = \relative {
  <c' e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
}
<<
  \new ChordNames {
    \chordmusic
  }
  {
    \chordmusic
  }
>>
```



L'apparition de silences dans un contexte `ChordNames` déclenchera l'impression d'un *markup* `noChordSymbol`.

```
<<
  \new ChordNames \chordmode {
    c1
    r1
    g1
    c1
  }
  \chordmode {
    c1
    r1
    g1
    c1
  }
>>
```



`\chords { ... }` est un raccourci de `\new ChordNames \chordmode { ... }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```



```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```



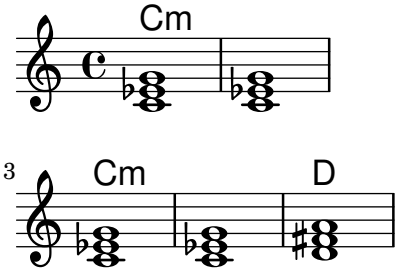
Morceaux choisis

Impression des accords si changement

Vous pouvez faire ressortir les chiffrages d'accords s'ils ne sont imprimés qu'aux changements d'accord ou en début de ligne.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
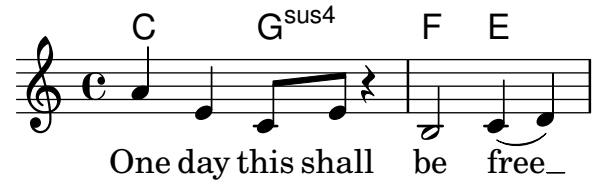
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
    \relative c' { \harmonies }
  }
>>
```



Chanson simple

Assembler des noms d'accords, une mélodie et des paroles permet d'obtenir la partition d'une chanson :

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Glossaire musicologique : Section “Accord” dans *Glossaire*.

Manuel de notation : [Saisie de musique en parallèle], page 194.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Bar_engraver” dans *Référence des propriétés internes*, Section “Chord_name_engraver” dans *Référence des propriétés internes*, Section “ChordNames” dans *Référence des propriétés internes*, Section “ChordName” dans *Référence des propriétés internes*, Section “Volta_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

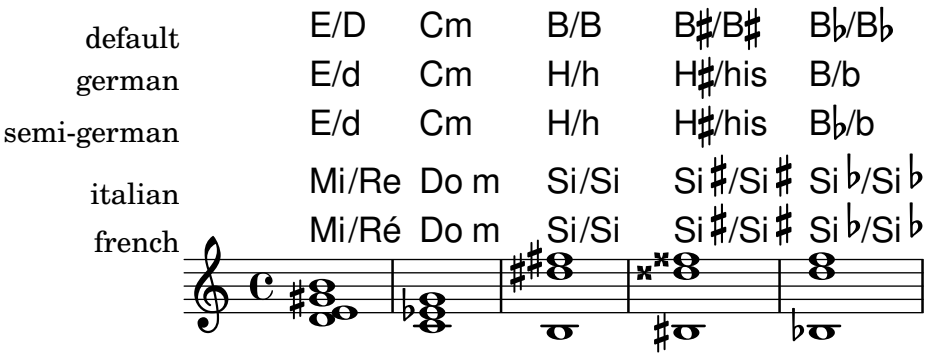
Un accord contenant un renversement ou dont la basse est changée ne sera pas chiffré correctement s’il est saisi en tant que musique simultanée.

Personnalisation des noms d’accord

Il existe plus d’un système de chiffrage d’accords. Le nom des accords varie selon les traditions musicales, et plusieurs symboles représentent un même accord. LilyPond vous permet de créer votre propre nomenclature d’accords, tant au niveau des noms que des symboles qui les représenteront.

Le modèle par défaut des chiffrages d’accord est celui de Klaus Ignatzek pour le jazz (cf. Section “Références bibliographiques” dans *Essai*). Il est possible de créer son propre modèle de chiffrages à l’aide des différentes propriétés mentionnées ci-dessous. LilyPond dispose d’un système alternatif de chiffrage jazz qui a été élaboré grâce à ces même propriétés. Les deux notations, Ignatzek et alternative, sont illustrés à l’annexe Section A.1 [Table des noms d’accord], page 702.

En plus des différents systèmes de nommage, le nom de la fondamentale varie selon la langue utilisée. Les instructions `\germanChords`, `\semiGermanChords`, `\italianChords` et `\frenchChords` permettent de définir la langue, comme vous pouvez le constater :



Nombre de carnets de chant allemands indiquent un accord mineur par l’emploi de caractères en minuscule, sans le suffixe `m`. Cette fonctionnalité est gérée par la propriété `chordNameLowercaseMinor` :

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```



La représentation d'un accord peut s'adapter à l'aide des propriétés suivantes :

`chordRootNamer`
: Dans les chiffrages d'accord jazz, la note fondamentale de chaque accord est exprimée par une lettre capitale, parfois suivie d'une altération, correspondant à la notation anglo-saxonne de la musique. Cette propriété a pour valeur la fonction qui transforme la hauteur de la note fondamentale en nom de note ; c'est donc en assignant une nouvelle fonction à cette propriété que l'on peut produire des noms de note spéciaux adaptés par exemple aux systèmes de notation d'autres pays.

`majorSevenSymbol`
: Cette propriété définit l'objet ajouté au `chordRootNamer` pour indiquer une septième majeure. Les options prédéfinies sont `whiteTriangleMarkup` (triangle blanc) et `blackTriangleMarkup` (triangle noir).

`additionalPitchPrefix`
: Lorsqu'un chiffrage contient des notes ajoutées, vous pouvez le préfixer d'une annotation. LilyPond n'en ajoute pas par défaut, dans le but de ne pas trop surcharger la partition ; vous pouvez cependant les faire apparaître si elles sont visuellement efficaces.

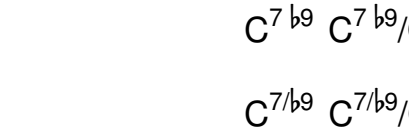
```
\new ChordNames {
  <c e g d'>    % add9
  \set additionalPitchPrefix = "add"
  <c e g d'>    % add9
}
```



`chordNoteNamer`
: Lorsqu'un chiffrage mentionne une note ajoutée (par exemple la basse), les règles utilisées sont par défaut celles définies par la propriété `chordRootNamer` ci-dessus. Cependant, la propriété `chordNoteNamer` permet de régler cet élément indépendamment, par exemple pour imprimer la basse en caractères minuscules.

`chordNameSeparator`
: Les différents termes d'un chiffrage jazz (par exemple les notes de l'accord et la basse) sont habituellement légèrement espacés. La propriété `chordNameSeparator` permet d'indiquer un autre séparateur. Le séparateur entre un chiffrage et sa basse est géré par la propriété `slashChordSeparator`.

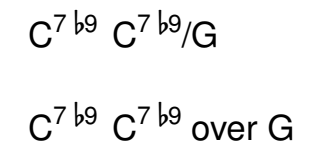
```
\chords {
  c4:7.9- c:7.9-/g
  \set chordNameSeparator = \markup { "/" }
  \break
  c4:7.9- c:7.9-/g
}
```



`slashChordSeparator`
: La note basse d'un accord n'est pas forcément la fondamentale. L'accord est alors « renversé » – certains diront « barré » parce que son chiffrage est habituellement flanqué d'une barre oblique entre l'accord de base et sa basse. La propriété

`slashChordSeparator` permet de modifier ce séparateur – par défaut la barre de fraction.

```
\chords {
  c4:7.9- c:7.9-/g
  \set slashChordSeparator = \markup { " over " }
  \break
  c4:7.9- c:7.9-/g
}
```



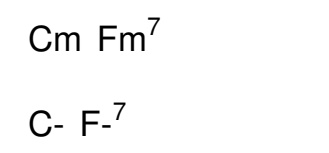
`chordNameExceptions`

Cette propriété recense, sous forme de paire, les accords mis en forme de manière particulière. Le premier élément de chacune des paires répertorie les différentes hauteurs qui constituent l'accord. Le second élément est un *markup* qui sera ajouté au `chordRootNamer` lors de l'impression du chiffrage.

`minorChordModifier`

Les accords mineurs sont habituellement identifiés par un `m` après leur fondamentale. Certaines nomenclatures ont cependant adopté un autre suffixe, comme le signe moins.

```
\chords {
  c4:min f:min7
  \set minorChordModifier = \markup { "-" }
  \break
  c4:min f:min7
}
```



`chordPrefixSpacer`

Le modificateur pour accord mineur, géré par la propriété `minorChordModifier`, est en principe accolé à la fondamentale. Vous pouvez cependant l'espacer de la fondamentale à l'aide de la propriété `chordPrefixSpacer`. Notez bien que cet espacement sera réduit à néant si la fondamentale est altérée.

Commandes prédéfinies

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Morceaux choisis

Modèles de chiffrage d'accords

Il est possible de créer votre propre modèle de chiffrages en réglant la propriété `chordNameExceptions`.

```
% modify maj9 and 6(add9)
% Exception music is chords with markups
```

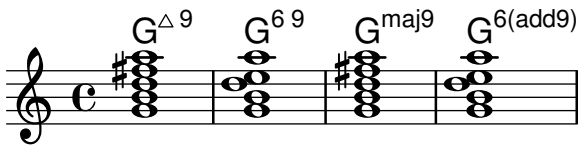
```
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% Convert music to list and prepend to existing exceptions.
chExceptions = #(append
  (sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

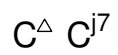
<<
   \new ChordNames \theMusic
   \new Voice \theMusic
>>
```



Chiffrage d'un maj7

La représentation d'un accord de septième majeure se gère par le `majorSevenSymbol`.

```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```

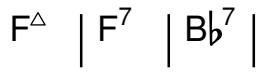


Chiffrages et barres de mesure

L'ajout du graveur `Bar_engraver` à un contexte `ChordNames` permet d'imprimer les barres de mesure entre les chiffrages.

```
\new ChordNames \with {
  \override BarLine.bar-extent = #'(-2 . 2)
  \consists "Bar_engraver"
}

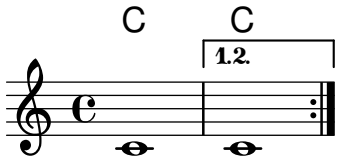
\chordmode {
  f1:maj7 f:7 bes:7
}
```



Crochet de reprise sous les chiffrages d'accord

L'ajout du `Volta_engraver` à la bonne portée permet d'imprimer les crochets de reprise entre les chiffrages et la portée.

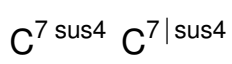
```
\score {
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



Personnalisation du séparateur d'accords

Le séparateur de termes d'un chiffrage peut adopter n'importe quelle forme à l'aide d'un *markup*.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
}
```



Voir aussi

Manuel de notation : Section A.2 [Modificateurs d'accord usuels], page 703, Section A.1 [Table des noms d'accord], page 702.

Essai sur la gravure musicale automatisée : Section "Références bibliographiques" dans *Essai*.

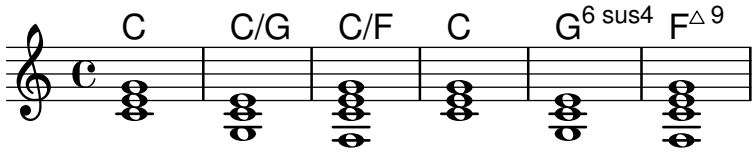
Fichiers d'initialisation : `scm/chords-ignatzek-names.scm`, `scm/chord-entry.scm`, `ly/chord-modifiers-init.ly`.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.
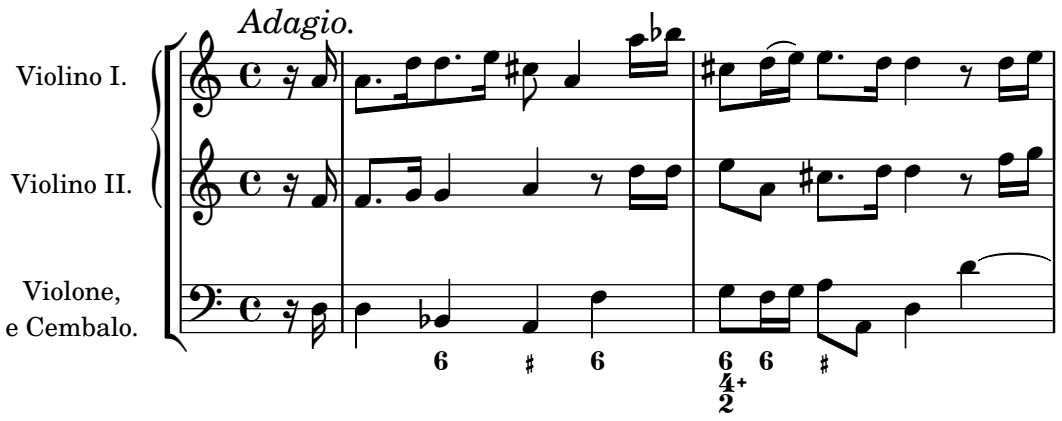
Problèmes connus et avertissements

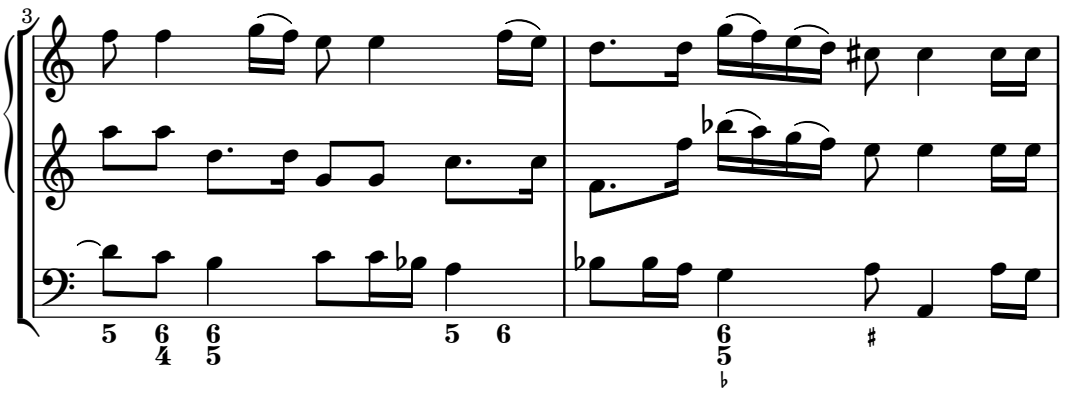
Les chiffrages d'accords ne sont déterminés que par la succession des hauteurs de notes. En d'autres termes, les accords inversés ne sont pas reconnus, ni les notes ajoutées à la basse. C'est pourquoi les accords saisis au moyen de la syntaxe `<...>` peuvent produire des chiffrages étranges.

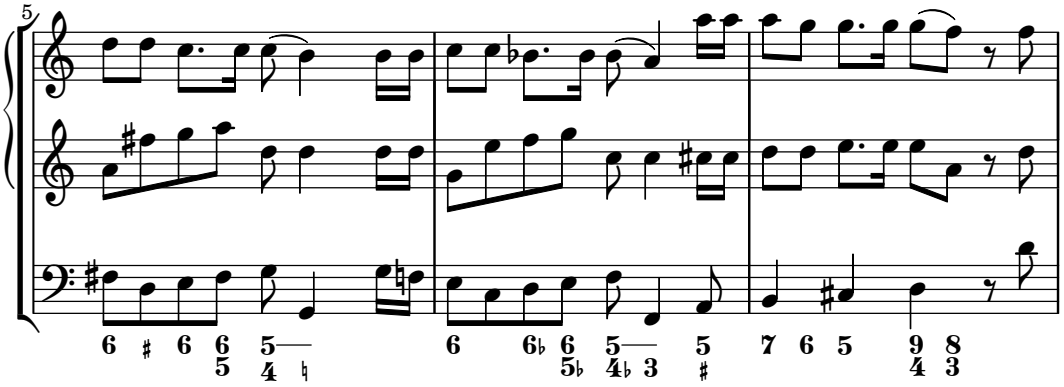
```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```



2.7.3 Basse chiffrée







LilyPond permet de générer des parties de continuo.

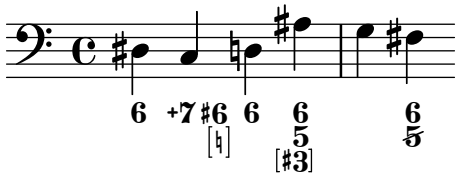
Introduction à la basse chiffrée

Les parties de basse continue étaient très répandues dans la musique baroque et jusqu'à la fin du XVIII^{e} siècle. Comme son nom l'indique, le *continuo* constitue une partie à lui seul, qui se déroule tout au long de l'œuvre pour en donner la structure harmonique.

Les musiciens du *continuo* jouent des claviers (clavecin, orgue) ou de tout autre instrument pouvant réaliser des accords. Leur partie est constituée d'une portée de basse aux notes agrémentées de combinaisons de chiffres et signes indiquant le développement des accords à jouer, ainsi que leur éventuel renversement. Cette notation était avant tout un guide, invitant le musicien à improviser de lui-même l'accompagnement.

LilyPond gère la basse chiffrée.

```
<<
  \new Voice { \clef bass dis4 c d ais g fis}
  \new FiguredBass {
    \figuremode {
      < 6 >4 < 7\+ >8 < 6+ [_!] >
      < 6 >4 <6 5 [3+] >
      < _ >4 < 6 5/>4
    }
  }
>>
```



La gestion de la basse chiffrée se décompose en deux parties. Dans un premier temps, le mode `\figuremode` permet de saisir les accords sous forme chiffrée. Le contexte `FiguredBass` s'occupera ensuite de gérer les objets `BassFigure`. La basse chiffrée pourra être attachée à un contexte `Staff`.

L'expression `\figures { ... }` constitue une raccourci à `\new FiguredBass \figuremode { ... }`.

Bien que la gestion de la basse chiffrée ressemble beaucoup à celle des accords, elle est beaucoup plus simpliste. Le mode `\figuremode` ne fait que stocker des chiffres que le contexte `FiguredBass` se chargera d'imprimer tels quels. En aucune manière ils ne sont transformés en son, et ils ne sont pas rendus dans un fichier MIDI.

Voir aussi

Glossaire musicologique : Section “basse chifrée” dans *Glossaire*.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

Saisie de la basse chiffrée

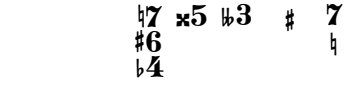
`\figuremode` permet de faire la relation entre ce qui est saisi et le mode de chiffrage. De plus amples informations quant aux différents modes sont regroupées à la rubrique Section 5.4.1 [Modes de saisie], page 666.

En mode de saisie, un chiffrage est délimité par `<` et `>`. La durée est indiquée après le `>` :

```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```


Une altération – y compris un bécarre – permet de modifier l’un des degrés, en adjoignant un `+` (dièse), un `-` (bémol) ou un `!` (bécarre) au chiffre considéré. Une altération double s’obtient en doublant le modificateur. Le chiffre est souvent omis lorsque la tierce est modifiée, ce qui s’obtient en utilisant un `_` en lieu et place du chiffre.

```
\figures {
  <7! 6+ 4-> <5++> <3--> < _+ > < 7 _!>
}
```



Vous pouvez stipuler un intervalle augmenté ou diminué :

```
\figures {
  <6\+ 5/> <7/>
}
```


Vous pouvez barrer un chiffre d’une oblique inversée, typiquement pour une « petite sixte » :

```
\figures {
  <6> <6\\>
}
```


Vous pouvez insérer des crochets :

```
\figures {
  <[12 _!] 8 [6  4]>
}
```

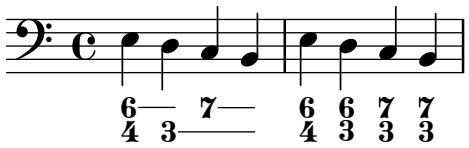
Vous pouvez aussi ajouter des chaînes de caractères ou des étiquettes – cf. Section A.11 [Commandes pour *markup*], page 746.

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```



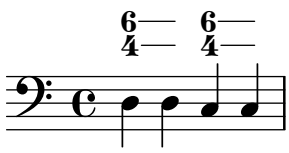
Lorsque des chiffrages se répètent, vous pouvez utiliser des lignes de prolongation.

```
<<
  {
    \clef bass
    e4 d c b,
    e4 d c b,
  }
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 3> <7 3> <7 3>
    \bassFigureExtendersOff
    <6 4>4 <6 3> <7 3> <7 3>
  }
>>
```



En pareil cas, la ligne de prolongation masquera toujours le chiffre qu'elle rappelle dans le chiffrage suivant à moins d'avoir été explicitement interrompue.

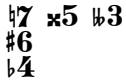
```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



Voici, de manière synthétique, les différents modificateurs disponibles :

Modificateur	Utilisation	Exemple

+, -, ! Altérations



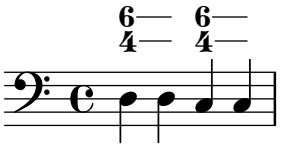
\+, / Augmentation ou diminution d'un degré



\\ Petite sixte



\! Terminaison d'une prolongation



Commandes prédéfinies

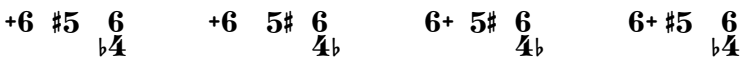
`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Morceaux choisis

Emplacement des altération en basse continue

On peut choisir d'imprimer les altérations et signes plus aussi bien avant qu'après les chiffres, en réglant les propriétés `figuredBassAlterationDirection` et `figuredBassPlusDirection`.

```
\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}
```



Voir aussi

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

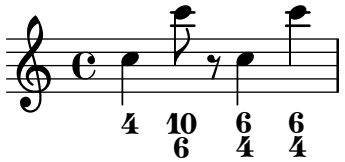
Référence des propriétés internes : Section "BassFigure" dans *Référence des propriétés internes*, Section "BassFigureAlignment" dans *Référence des propriétés internes*, Section "BassFigureBracket" dans *Référence des propriétés internes*, Section "BassFigureContinuation" dans *Référence des propriétés internes*, Section "BassFigureLine" dans *Référence des propriétés internes*, Section "FiguredBass" dans *Référence des propriétés internes*.

Gravure de la basse chiffrée

Une ligne de basse chiffrée s'imprime soit dans un contexte `FiguredBass`, soit dans la plupart des autres contextes du niveau de la portée.

Le contexte `FiguredBass` ne tient aucun compte des notes qui apparaissent sur la portée.

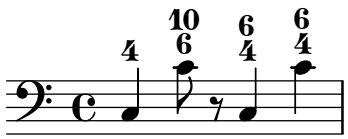
```
<<
  \relative {
    c''4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



Il est impératif, dans cet exemple, d'instancier explicitement le contexte `FiguredBass` pour éviter l'apparition d'une portée supplémentaire vide.

On peut ajouter une basse chiffrée directement à un contexte `Staff`. L'alignement vertical est alors automatiquement ajusté.

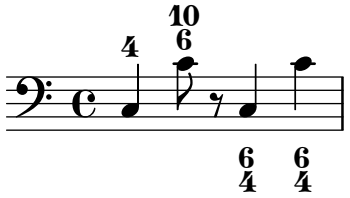
```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff" {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



La basse chiffrée attachée à un contexte `Staff` peut se positionner au-dessus ou en dessous de la portée.

```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    \bassFigureStaffAlignmentDown
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff" {
```

```
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



Commandes prédéfinies

`\bassFigureStaffAlignmentDown`, `\bassFigureStaffAlignmentUp`, `\bassFigureStaffAlignmentNeutral`.

Voir aussi

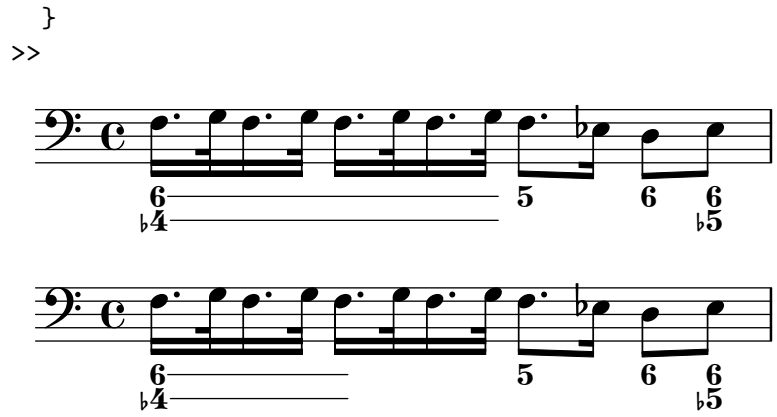
Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

Référence des propriétés internes : Section “BassFigure” dans *Référence des propriétés internes*, Section “BassFigureAlignment” dans *Référence des propriétés internes*, Section “BassFigureBracket” dans *Référence des propriétés internes*, Section “BassFigureContinuation” dans *Référence des propriétés internes*, Section “BassFigureLine” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les lignes de prolongation seront correctes dès lors que notes et chiffrages adoptent des durées identiques.

```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are correct here,
    % with the same rhythm as the bass
    \repeat unfold 4 { <6 4->16. <6 4->32 }
    <5>8. r16 <6>8 <6\! 5->
  }
>>
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are incorrect here,
    % even though the timing is the same
    <6 4->4 <6 4->4
    <5>8. r16 <6>8 <6\! 5->
```



2.8 Musique contemporaine

L'aube du XX^e siècle a vu bourgeonner nombre de techniques et de styles de composition. Qu'il s'agisse des nouveaux développements autour de l'harmonie et du rythme, de l'expansion du spectre des hauteurs et de l'évolution de nombreuses techniques instrumentales, tous ces différents phénomènes ont participé à l'évolution de la notation musicale. Les paragraphes qui suivent sont là pour vous proposer des références et informations quant à ces nouvelles techniques de notation.

2.8.1 Hauteur et harmonie en musique contemporaine

Intéressons-nous tout d'abord à ce qui relève de la notation des hauteurs et à l'harmonie en musique contemporaine.

Généralités en matière de hauteur et d'harmonie

- La notation habituelle des quarts de tons est abordée à la rubrique [Nom des notes dans d'autres langues], page 8.
- Les tonalités inhabituelles sont abordées à la rubrique [Armure], page 22.
- Les pratiques contemporaines en matière d'altération sont abordées à la rubrique [Altérations accidentelles automatiques], page 29.

Notation microtonale

Armures contemporaines et harmonie

2.8.2 Approches du rythme en musique contemporaine

Abordons à présent certaines particularités de la notation du rythme en musique contemporaine.

Généralités sur le rythme en musique contemporaine

- Les métrique composites sont abordées à la rubrique [Métrique], page 69.
- Les bases de la polymétrie sont abordées à la rubrique [Notation polymétrique], page 80.
- Certaines particularités en matière de ligature sont abordées à la rubrique [Liens de croches en soufflet], page 102.
- Les lignes de mensuration (barres de mesures uniquement entre les portées) sont abordées à la rubrique [Regroupement de portées], page 199.

N-olets et musique contemporaine

Métriques contemporaines

Notation polymétrique étendue

Ligatures et musique contemporaine
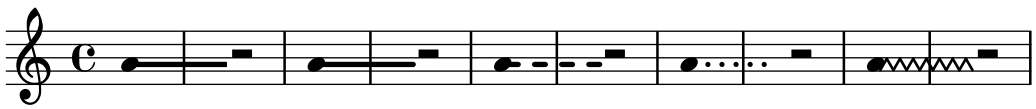
Barres de mesure et musique contemporaine

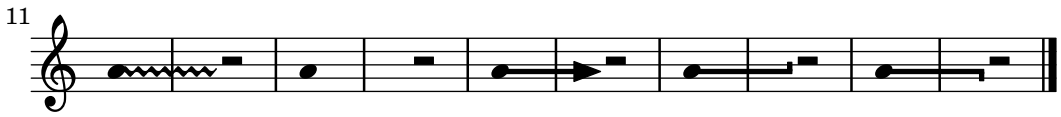
2.8.3 Notation graphique

Les éléments rythmiques peuvent se prolongar par une ligne de durée représentée par un objet graphique `DurationLine`. Cette ligne peut adopter différents styles : `'beam`, `'line`, `'dashed-line`, `'dotted-line`, `'zigzag`, `'trill` ou `'none`. La ligne peut se terminer par un crochet (uniquement pour le style `'beam`) ou une flêche.

```
\layout {
  \context {
    \Voice
    \consists "Duration_line_engraver"
    \omit Stem
    \omit Flag
    \omit Beam
    \override NoteHead.duration-log = 2
  }
}

{
  a'1\- s2 r
  \once \override DurationLine.style = #'line
  a'1\- s2 r
  \once \override DurationLine.style = #'dashed-line
  \once \override DurationLine.dash-period = 2
  a'1\- s2 r
  \once \override DurationLine.style = #'dotted-line
  \once \override DurationLine.dash-period = 1
  \once \override DurationLine.bound-details.right.padding = 1
  a'1\- s2 r
  \once \override DurationLine.thickness = 2
  \once \override DurationLine.style = #'zigzag
  a'1\- s2 r
  \once \override DurationLine.style = #'trill
  a'1\- s2 r
  \once \override DurationLine.style = #'none
  a'1\- s2 r
  \once \override DurationLine.bound-details.right.end-style = #'arrow
  a'1\- s2 r
  \override DurationLine.bound-details.right.end-style = #'hook
  a'1\- s2 r
  \override DurationLine.details.hook-direction = #DOWN
  a'1\- s2 r
  \bar "|."
}
```





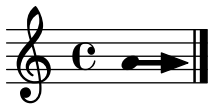
Problèmes connus et avertissements

Lorsque l'objet `DurationLine` atteint la fin de la partition, son extrémité est absente, pour des raisons techniques. Une solution de contournement pourrait ressembler à :

```
\layout {
  \context {
    \Voice
    \consists "Duration_line_engraver"
    \omit Stem
    \omit Flag
    \omit Beam
    \override NoteHead.duration-log = 2
  }
}

lastEndStyle =
#(define-music-function (end-style)(symbol?)
#{
  \override DurationLine.after-line-breaking =
    #(lambda (grob)
      (let* ((orig (ly:grob-original grob))
             (siblings (if (ly:grob? orig)
                           (ly:spanner-broken-into orig) '()))
             (last-grob (if (pair? siblings) (last siblings) #f)))
        (if last-grob
            (ly:grob-set-nested-property!
              last-grob
              '(bound-details right-broken end-style) end-style))))
#})

{
  \once \override DurationLine.bound-details.right.end-style = #'arrow
  \lastEndStyle #'arrow
  a'1\-
  \bar "|."
}
```



2.8.4 Techniques de partition contemporaine

2.8.5 Nouvelles techniques instrumentales

2.8.6 Informations complémentaires et exemples pertinents

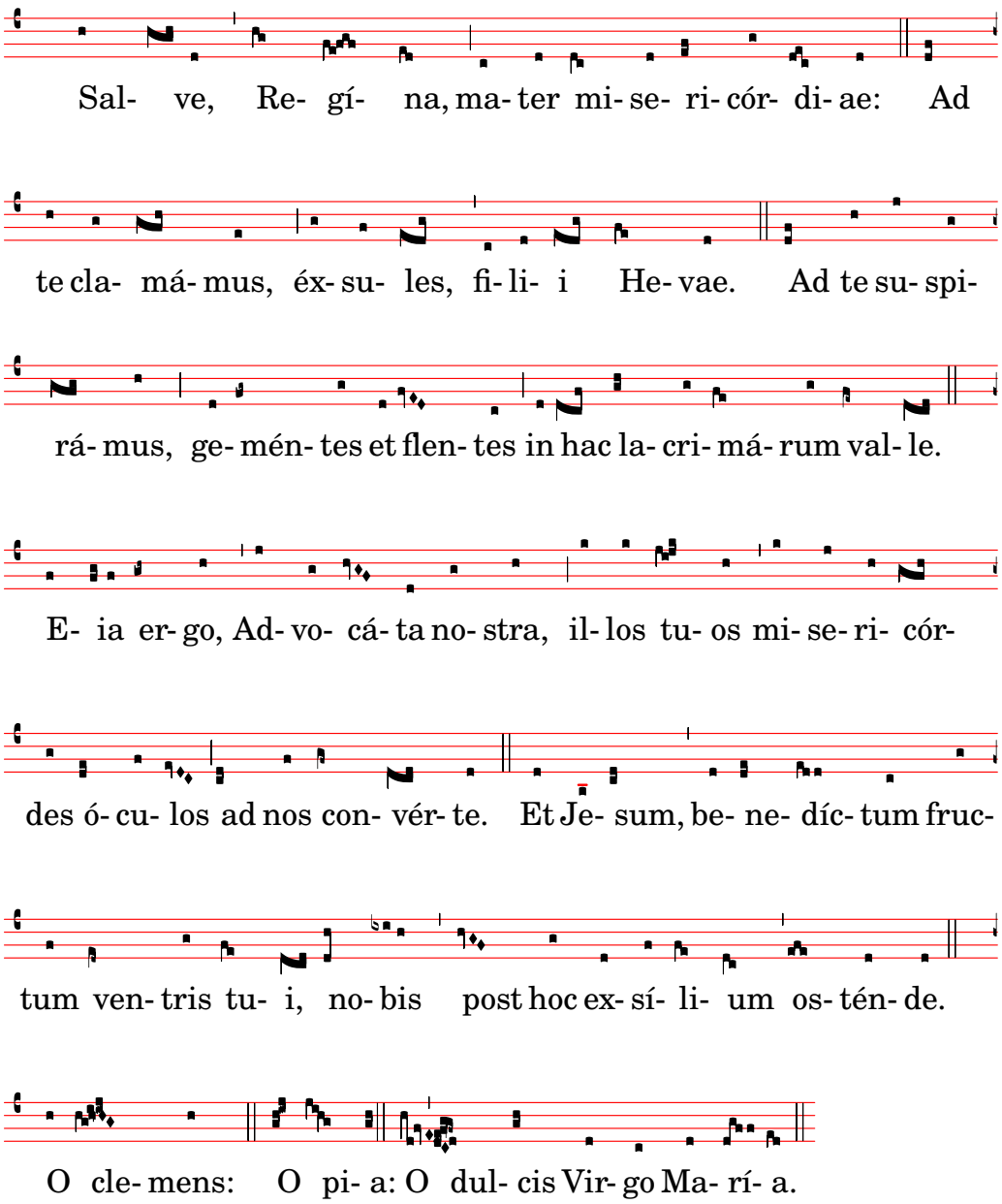
Vous trouverez ici une sélection d'ouvrages de référence, d'exemples et autres ressources qui vous permettront d'étudier plus avant la notation contemporaine.

Ouvrages et articles sur la notation en musique contemporaine

- *Music Notation in the Twentieth Century: A Practical Guidebook* par Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* par Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* par Alfred Blatter [Schirmer, 2de ed. 1997]

Partitions et exemples

2.9 Notations anciennes



La gestion par LilyPond des formes de notation ancienne inclut des fonctionnalités spécifiques à la notation mensurale, au chant grégorien et à la notation de style kievien. Ces fonctionnalités sont accessibles en modifiant les propriétés de style des objets graphiques tels que tête de note ou silence, ou bien grâce aux contextes prédéfinis à cet effet.

De nombreux objets graphiques – « grobs » dans le jargon de LilyPond – disposent d'une propriété `style`. Manipuler cette propriété permet d'adapter l'aspect typographique des *grobs* à une forme de notation particulière, ce qui évite la création de nouveaux concepts de notation. Voir à ce sujet

- [Têtes de note anciennes], page 475,
- [Altérations et armures anciennes], page 477,
- [Silences anciens], page 476,
- [Clefs anciennes], page 473,
- [Clefs grégoriennes], page 480,
- [Crochets anciens], page 475,
- [Métriques anciennes], page 474.

D'autres aspects de la notation ancienne ne peuvent pas être gérés aussi simplement qu'en jouant sur les propriétés d'un style appliqué à un objet graphique ou en lui ajoutant des articulations. Certains concepts sont spécifiques à la notation ancienne :

- [Guidons], page 471,
- [Divisions], page 481,
- [Ligatures], page 470.

Voir aussi

Glossaire musicologique : Section "custos" dans *Glossaire*, Section "ligature" dans *Glossaire*, Section "mensural notation" dans *Glossaire*.

Manuel de notation : [Altérations et armures anciennes], page 477, [Clefs grégoriennes], page 480, [Crochets anciens], page 475, [Divisions], page 481, [Guidons], page 471, [Ligatures], page 470, [Métriques anciennes], page 474, [Silences anciens], page 476, [Têtes de note anciennes], page 475.

2.9.1 Formes de notation ancienne prises en charge

En matière de chant grégorien, LilyPond dispose de trois différents styles :

- *Editio Vaticana* constitue un style à part entière dédié au chant grégorien, le plus approchant possible des éditions de Solesmes, éditeur officiel du Vatican depuis 1904. LilyPond dispose de tous les signes de notation propres à ce style, y compris les ligatures, custodes et certaines spécificités comme le quilisma et l'oriscus.
- *Editio Medicaea* dispose d'un certain nombre de spécificités des éditions qui faisaient autorité avant Solesmes. On le connaît aussi sous le nom de Ratisbone. Ce qui le distingue le plus du style *Vaticana* réside dans les clefs, en forme de barres obliques, et les têtes de note, plus carrées et régulières.
- Le style *Hufnagel* (« clou de fer à cheval ») ou *gothique* imite le style des manuscrits médiévaux d'Allemagne et d'Europe centrale. Il tire son nom de l'allure des notes (en virgule ou *virga*) qui ressemblent à des têtes de clou.

LilyPond dispose de trois styles imitant les manuscrits du Bas Moyen Âge et de la Renaissance ainsi que les premières impressions de musique mesurée.

- Le style *Mensural* est celui qui se rapproche le plus des manuscrits de la période allant de la fin du Moyen Âge au début de la Renaissance, avec ses petites têtes de note en forme de losange étroit et ses silences comme dessinés à main levée.
- Le style *Neomensural* est une version moderne et stylisée du style mensural : les têtes de note sont un peu plus galbées et les silences plus rectilignes. Ce style est tout à fait approprié à l'incipit d'une transcription de musique ancienne.

- Le style *Petrucci* tire son nom du fameux graveur vénitien Ottaviano Petrucci (1466-1539), premier imprimeur à utiliser des caractères amovibles pour la musique dans son édition du *Harmonice musices odhecaton* en 1501. Les têtes de notes de ce style sont plus larges que pour les autres styles mensuraux.

Bien qu'il ne soient pas complets, les styles *Baroque* et *Classical* diffèrent du style par défaut par quelques détails – certaines têtes de note pour le *Baroque* et le soupir pour le *Classical*.

Seul le style mensural dispose de signes alternatifs couvrant tous les aspects de la notation. Ainsi, les silences et les crochets sont absents du style grégorien puisqu'ils ne sont pas utilisés dans la notation du plain-chant ; le style Petrucci ne dispose en propre d'aucun crochet ni d'altération.

Chacun des éléments de notation peut donc être modifié de manière indépendante jusqu'à, pourquoi pas, utiliser dans une même partition des crochet en *Mensural*, des têtes de note de *Petrucci*, des silences du *Classical* et des clefs du style *Vaticana*.

Voir aussi

Glossaire musicologique : Section "flag" dans *Glossaire*, Section "mensural notation" dans *Glossaire*.

2.9.2 Considérations communes aux musiques anciennes

Contextes prédéfinis

LilyPond dispose, en matière de musique ancienne et de grégorien, de contextes prédéfinis. Ceux-ci contiennent tout ce qui est nécessaire à la gestion d'une voix ou d'une portée selon le style adopté. Si tout cela vous dépasse et que vous désirez plonger dans le vif du sujet sans trop vous préoccuper d'ajuster des contextes, consultez les pages dédiées aux contextes prédéfinis. Ils vous permettront d'adapter vos contextes de voix et de portée, et vous n'aurez plus qu'à saisir les notes dans un contexte `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice` ou `MensuralStaff`. Vous trouverez des détails sur ces contextes aux rubriques

- [Contextes du chant grégorien], page 479,
- [Contextes de musique mensurale], page 472.

Voir aussi

Glossaire musicologique : Section "mensural notation" dans *Glossaire*.
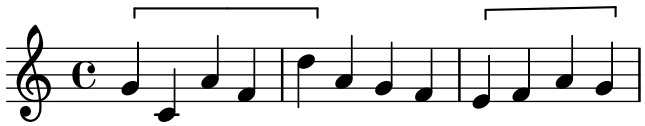
Manuel de notation : [Contextes du chant grégorien], page 479, [Contextes de musique mensurale], page 472.

Ligatures

Une ligature est un symbole graphique qui représente un groupe d'au moins deux notes distinctes. Les ligatures ont commencé à apparaître dans les manuscrits de chant grégorien, pour indiquer des suites ascendantes ou descendantes de notes.

Les ligatures s'indiquent par un bornage entre `\[` et `\]`. Certains styles de ligature peuvent demander un complément de syntaxe spécifique. Par défaut, le graveur `LigatureBracket` place un simple crochet au dessus de la ligature :

```
\relative {
  \[ g' c, a' f d' \]
  a g f
  \[ e f a g \]
}
```



Deux autres styles de ligature sont accessibles : vatican pour le grégorien et mensural pour la musique ancienne (seules sont disponibles les ligatures mensurales blanches, avec quelques limitations). Selon le style de ligature désiré, il faut remplacer, dans le contexte `Voice` approprié, le graveur `Ligature_bracket_engraver` par le graveur de ligature qui convient – voir les rubriques [Ligatures mensurales], page 478, et [Neumes et ligatures grégoriennes], page 483, à ce sujet.

Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

Manuel de notation : [Ligatures mensurales], page 478, [Neumes et ligatures grégoriennes], page 483.

Problèmes connus et avertissements

La gestion de l’espacement spécifique aux ligatures n’est à ce jour pas implémentée. En conséquence, les ligatures sont trop espacées les unes des autres et les sauts de ligne mal ajustés.

Les paroles ne s’alignent pas de manière satisfaisante en présence de ligatures.

Les altérations ne pouvant être imprimées à l’intérieur d’une ligature, il faut les rassembler et les imprimer juste avant.

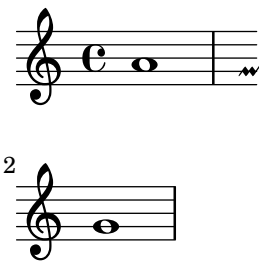
La syntaxe utilisée correspond à l’ancienne convention de préfixage `\[ expression musicale\]`. Pour des raisons d’uniformité, nous opterons probablement pour le style en suffixe (postfix) `note\[ ... note\]`.

Guidons

Un guidon — *custos*, pluriel *custodes* en latin — est un symbole qui apparaît à la fin d’une portée. Il montre la hauteur de la ou des premières notes de la portée suivante, donnant une indication judicieuse à l’exécutant.

Les guidons étaient couramment utilisés jusqu’au XVII^{e} siècle. De nos jours, on les retrouve uniquement dans quelques formes particulières de notation telles que les éditions contemporaines de chant grégorien comme les *editio vaticana*. Différents glyphes existent selon le style de notation.

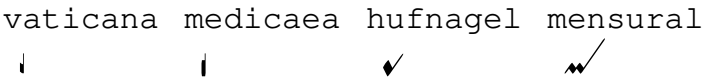
L’impression de guidons s’obtient en affectant, dans un bloc `\layout`, le Section “Custos_engraver” dans *Référence des propriétés internes* au contexte `Staff`, comme le montre l’exemple suivant.



Le glyphe du guidon est déterminé par la propriété `style`. Les styles disponibles sont `vaticana`, `medicaea`, `hufnagel` et `mensural`.

```
\new Lyrics \lyricmode {
  \markup { \column {
    \typewriter "vaticana "
    \line { " " \musicglyph "custodes.vaticana.u0" }
```

```
  } }
  \markup { \column {
    \typewriter "medicaea "
    \line { " " \musicglyph "custodes.medicaea.u0" }
  }}
  \markup { \column {
    \typewriter "hufnagel "
    \line { " " \musicglyph "custodes.hufnagel.u0" }
  }}
  \markup { \column {
    \typewriter "mensural "
    \line { " " \musicglyph "custodes.mensural.u0" }
  }}
}
```



Voir aussi

Glossaire musicologique : Section “custos” dans *Glossaire*.

Référence des propriétés internes : Section “Custos” dans *Référence des propriétés internes*.
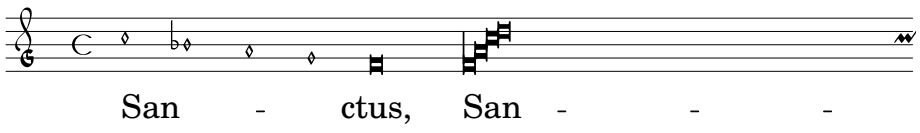
Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.
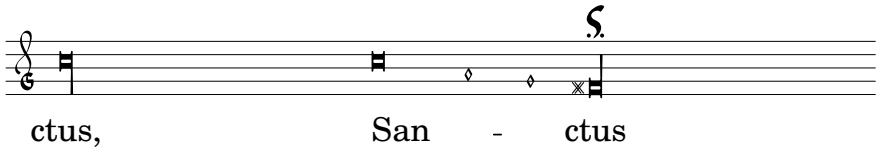
2.9.3 Typographie de musique ancienne

Contextes de musique mensurale

Les contextes `MensuralVoice` et `MensuralStaff` permettent de graver des chants dans le style mesuré. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant comme ci-après :

```
\score {
  <<
    \new MensuralVoice = "discantus" \relative {
      \hide Score.BarNumber {
        c''1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c\breve d\melismaEnd \]
        c\longa
        c\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```





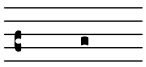
Voir aussi

Glossaire musicologique : Section “mensural notation” dans *Glossaire*.

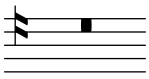
Clefs anciennes

Les clefs dédiées à la musique ancienne sont disponibles à l’aide de la commande `\clef`. Certaines de ces clés utilisent le même glyphe, attaché à l’une ou l’autre des lignes de la portée. Le chiffre porté en suffixe permet alors de les différencier, en partant de la ligne inférieure.

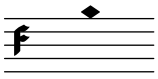
```
\override NoteHead.style = #'vaticana.punctum
\clef "vaticana-do1"
c'1
```
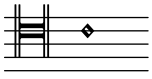


```
\override NoteHead.style = #'medicaea.punctum
\clef "medicaea-do3"
c'1
```



```
\override NoteHead.style = #'hufnagel.punctum
\clef "hufnagel-fa2"
c'1
```



```
\override NoteHead.style = #'neomensural
\clef "neomensural-c4"
c'1
```



Un glyphe de clef peut tout à fait arbitrairement être positionné sur une ligne particulière, comme indiqué à la rubrique [Clefs], page 18. Une liste exhaustive des différentes clefs est disponible à l’annexe Section A.10 [Styles de clef], page 740.

Voir aussi

Glossaire musicologique : Section “clef” dans *Glossaire*, Section “mensural notation” dans *Glossaire*.

Manuel de notation : [Clefs], page 18, [Clefs grégoriennes], page 480.

Fichiers d’initialisation : `scm/parser-clef.scm`.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Clef” dans *Référence des propriétés internes*, Section “Clef_engraver” dans *Référence des propriétés internes*, Section “clef-interface” dans *Référence des propriétés internes*, Section “ClefModifier” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La clef de sol mensurale est calquée sur celle de Petrucci.

Métriques anciennes

Les chiffrages de métrique mensurale sont partiellement pris en charge. Les glyphes ne font que représenter des métriques particulières. En d'autres termes, pour obtenir le glyphe correspondant à une métrique mensurale particulière à l'aide de la commande `\time n/m`, vous devez choisir la paire `(n,m)` parmi les valeurs suivantes :

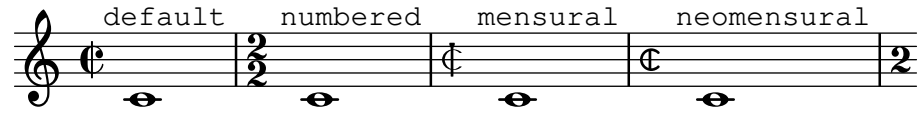
\time 4/4 \time 2/2 \time 6/4 \time 6/8

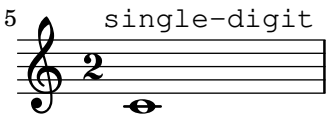
\time 3/2 \time 3/4 \time 9/4 \time 9/8

\time 4/8 \time 2/4

La propriété `style` de l'objet `TimeSignature` permet d'accéder aux indicateurs de métrique anciens. Les styles `neomensural` et `mensural` sont disponibles. Vous avez vu ci-dessus le style `neomensural`, particulièrement utilisé pour l'incipit des transcriptions. Le style `mensural` imite l'aspect de certaines éditions du XVI^{e} siècle.

Voici les différences entre les styles :





La rubrique [Métrique], page 69, expose les principes généraux sur l'utilisation des indications de métrique.

Voir aussi

Glossaire musicologique : Section "mensural notation" dans *Glossaire*.

Manuel de notation : [Métrique], page 69.

Problèmes connus et avertissements

Les équivalences de durées de note ne sont pas modifiées par un changement de métrique. Par exemple, l'équivalence une brève pour trois semi-brèves (*tempus perfectum*) doit s'effectuer à la main en entrant :

```
breveTP = #(ly:make-duration -1 0 3/2)
...
{ c\breveTP f1 }
```

Ce qui définira `breveTP` à 3/2 fois 2 = 3 fois une ronde.

Les symboles `mensural68alt` et `neomensural68alt` – alternatives à la métrique 6/8 – ne sont pas accessibles par la commande `\time`. Utilisez alors un `\markup {\musicglyph "timesig.mensural68alt" }`.

Têtes de note anciennes

Pour de la musique ancienne, vous disposez de plusieurs styles de tête de note, en plus du style par défaut `default`. Vous pouvez affecter à la propriété `style` de l'objet `NoteHead` les valeurs `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` ou `semipetrucci`.

Le style `baroque` diffère du style `default` par

- la disponibilité de la `maxima`, et
- la `\breve` qui sera carrée et non pas ovoïde.

Les styles `neomensural`, `mensural` et `petrucci` diffèrent du `baroque` par ceci :
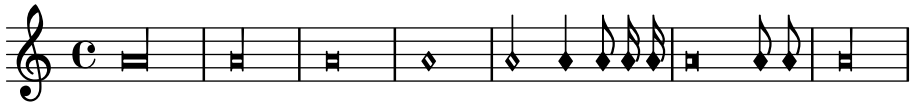
- les notes de durée inférieure ou égale à une ronde sont en forme de losange, et
- les hampes sont centrées sur la tête.

Le style `blackpetrucci` permet d'obtenir, en notation mensurale blanche, des têtes noircies. Cependant, et dans la mesure où le style de tête n'influence en rien le nombre des crochets, une *semiminima* devrait alors se noter `a8*2` plutôt que `a4`, de telle sorte qu'elle ne se confonde pas avec une *minima*. Le multiplicateur peut varier, pour indiquer par exemple un triolet.

Le style `semipetrucci` permet de partiellement noircir certaines têtes, comme la brève, la longue et la maxime.

L'exemple suivant illustre le style `petrucci`.

```
\compressEmptyMeasures
\autoBeamOff
\override NoteHead.style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```



La rubrique Section 1.1.4 [Têtes de note], page 40, présente tous les styles de notes disponibles.

Voir aussi

Glossaire musicologique : Section “mensural notation” dans *Glossaire*, Section “note head” dans *Glossaire*.

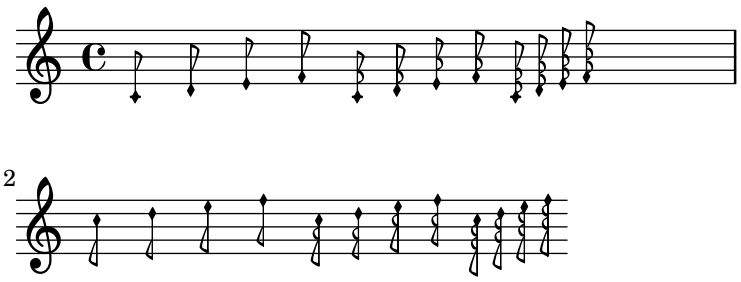
Manuel de notation : Section 1.1.4 [Têtes de note], page 40.

Crochets anciens

Le réglage de la propriété `flag-style` de l'objet hampe (`Stem`) donne accès aux crochets de style ancien. Les seuls styles actuellement pris en charge sont `default` et `mensural`.

```
\relative c' {
  \override Flag.style = #'mensural
  \override Stem.thickness = #1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8
```

```
  c'8 d e f c16 d e f c32 d e f
}
```



Notez que, pour chaque crochet mensural, l'extrémité la plus proche de la tête de note sera attachée à une ligne de la portée.

Il n'existe pas de crochet spécifique au style néomensural.

Les crochets n'existent pas en notation grégorienne.

Voir aussi

Glossaire musicologique : Section "flag" dans *Glossaire*, Section "mensural notation" dans *Glossaire*.

Problèmes connus et avertissements
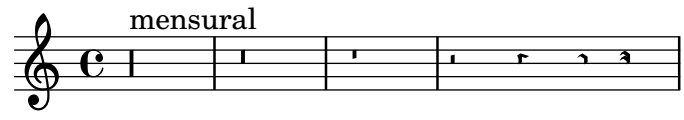
L'alignement vertical des crochets par rapport aux lignes de la portée sous-entend que les hampes se terminent toujours soit sur une ligne, soit à l'exact milieu d'un interligne. Ceci n'est pas toujours réalisable, surtout si vous faites appel à des fonctionnalités avancées de présentation de la notation classique qui, par définition, ne sont pas prévues pour être appliquées à la notation mensurale.

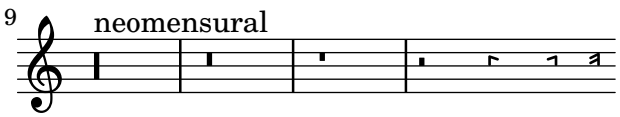
Silences anciens

La propriété `style` de l'objet `Rest` permet d'obtenir des silences de type ancien. Vous disposez des styles `mensural` et `neomensural`.

En voici une illustration.

```
\compressEmptyMeasures
\override Rest.style = #'mensural
r\longa^"mensural" r\breve r1 r2 r4 r8 r16 s \break
\override Rest.style = #'neomensural
r\longa^"neomensural" r\breve r1 r2 r4 r8 r16
```





Les styles `mensural` et `neomensural` ne disposent pas des huitième et seizième de soupir ; LilyPond utilise dans de tels cas le style par défaut.

Voir aussi

Glossaire musicologique : Section “mensural notation” dans *Glossaire*.

Manuel de notation : [Silences], page 61.

Morceaux choisis : Section “Notations anciennes” dans *Morceaux choisis*.

Problèmes connus et avertissements

En style mensural, LilyPond utilise pour un silence correspondant à une *maxima*, le même glyphe que pour un *longa* ; il faut donc les multiplier pour obtenir la durée *ad hoc*. Des silences correspondant à une *longa* ne sont pas groupés automatiquement ; utilisez en pareil cas des « notes silencieuses ».

Altérations et armures anciennes

Le style `mensural` dispose d'un dièse et d'un bémol différents du style par défaut. La notation mensurale n'utilise que très rarement le bécarre ; sont utilisés plutôt le dièse ou le bémol. Par exemple, un si bécarre en fa majeur sera indiqué par la présence d'un dièse. Si toutefois il était requis, le bécarre sera emprunté au style `vaticana`.

mensural



Pour disposer des formes anciennes d'altération, utilisez la propriété `glyph-name-alist` des objets graphiques `Accidental` et `KeySignature`, comme ceci :

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

Voir aussi

Glossaire musicologique : Section “accidental” dans *Glossaire*, Section “key signature” dans *Glossaire*, Section “mensural notation” dans *Glossaire*, Section “Pitch names” dans *Glossaire*.

Manuel de notation : [Altérations], page 6, [Altérations accidentelles automatiques], page 29, [Armure], page 22, Section 1.1 [Hauteurs], page 1.
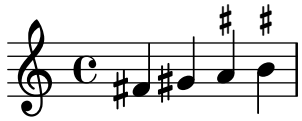
Référence des propriétés internes : Section “KeySignature” dans *Référence des propriétés internes*.

Altérations suggérées (*musica ficta*)

Dans la pratique ancienne, avant le XVII^{e} siècle, les altérations accidentelles de l'échelle modale n'étaient pas systématiquement notées et il incombait aux chanteurs, en fonction de certaines règles, de décider s'ils devaient chanter tel degré bémol, bécarre ou dièse. Cette technique est appelée *musica ficta*. Les transcriptions modernes de telles œuvres font apparaître ces altérations en surplomb de la note.

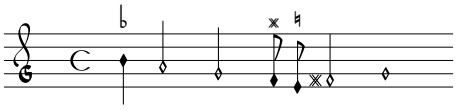
La reproduction de ces altérations suggérées est assurée par l'activation de la fonction `suggestAccidentals`.

```
\relative {
  fis' gis
  \set suggestAccidentals = ##t
  ais bis
}
```



Cette fonction considérera **toute** altération comme étant de la *musica ficta*, ce tant qu'elle n'aura pas été désactivée par un `\set suggestAccidentals = ##f`. Il est de ce fait plus pratique de recourir à une clause `\once \set suggestAccidentals = ##t`, qui peut tout à fait faire l'objet d'un raccourci :

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative
  \new MensuralVoice  {
    \once \set suggestAccidentals = ##t
    bes'4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



Voir aussi

Référence des propriétés internes : Section "Accidental_engraver" dans *Référence des propriétés internes*, Section "AccidentalSuggestion" dans *Référence des propriétés internes*.

Ligatures mensurales

Les ligatures mensurales blanches sont prises en charge, avec des limitations.

La gravure des ligatures mensurales blanches s'obtient après avoir remplacé, dans le contexte `Voice`, le `Ligature_bracket_engraver` par le `Mensural_ligature_engraver`, comme ici :

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Mensural_ligature_engraver"
  }
}
```

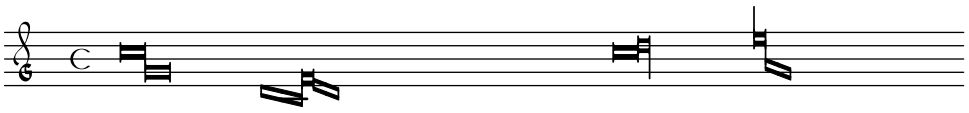
Lorsque le code ci-dessus est employé, l'aspect d'une ligature mensurale blanche est déterminé à partir des hauteurs et durées des notes qui la composent. Bien que cela demande un temps d'adaptation au nouvel utilisateur, cette méthode offre l'avantage que toute l'information musicale incluse dans la ligature est connue en interne. Ceci est non seulement important pour le rendu MIDI, mais aussi pour des questions de transcription automatisée d'une ligature.

Il se peut que deux notes consécutives puissent être représentées aussi bien par deux carrées que par un parallélogramme oblique (en forme de flexe). Par défaut, LilyPond présentera deux carrés ; l'impression d'une flexe s'obtient par affectation, pour la **deuxième** note, de la propriété `ligature-flexa`. Le réglage de la longueur d'une flexe se gère par la propriété de tête de note `flexa-width`.
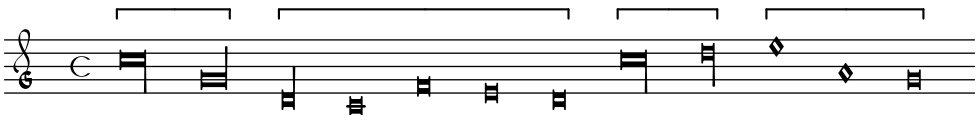
Par exemple,

```
\score {
  \relative {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c''\maxima g \]
```

```
    \[ d\longa
       \override NoteHead.ligature-flexa = ##t
       \once \override NoteHead.flexa-width = #3.2
       c\breve f e d \]
    \[ c'\maxima d\longa \]
    \[ e1 a, g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
    }
  }
}
```



Si on ne remplace pas le `Ligature_bracket_engraver` par le `Mensural_ligature_engraver`, on obtient



Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

Manuel de notation : [Ligatures], page 470, [Neumes et ligatures grégoriennes], page 483.

Problèmes connus et avertissements

L’espacement horizontal peut laisser à désirer. Les altérations peuvent se chevaucher avec les notes précédentes.

2.9.4 Typographie du chant grégorien

Si vous écrivez en notation grégorienne, le `Vaticana_ligature_engraver` se chargera de sélectionner les têtes de note appropriées ; il est donc inutile de spécifier le style à utiliser. Vous pouvez cependant spécifier par exemple le style `vaticana_punctum` pour obtenir des neumes punctums. De même, c’est le `Mensural_ligature_engraver` qui se chargera des ligatures mensurales.

Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

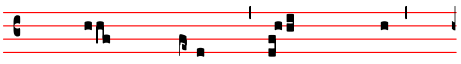
Manuel de notation : [Ligatures], page 470, [Ligatures mensurales], page 478.

Contextes du chant grégorien

Les contextes prédéfinis `VaticanaVoice` et `VaticanaStaff` permettent de graver le chant grégorien dans le style des éditions vaticanes. Ces contextes initialisent les propriétés de tous

les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant, comme ci-dessous :

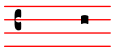
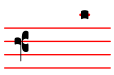
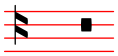
```
\include "gregorian.ly"
\score {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
  >>
}
```

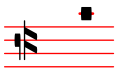
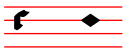




Clefs grégoriennes

Le tableau suivant présente les différentes clefs grégoriennes que vous pouvez sélectionner avec la commande `\clef`. Certaines de ces clefs utilisent le même glyphe, attaché à l'une ou l'autre des lignes de la portée. Le chiffre porté en suffixe permet alors de les différencier ; la numérotation des lignes va de bas en haut. Vous pouvez néanmoins forcer le positionnement du glyphe sur une ligne, comme expliqué à la section [Clefs], page 18. Dans la colonne exemple, la note suivant la clef est un do médium.

Description	**Clef disponible**	**Exemple**
Clef d'ut, style des éditions vaticanes	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Clef de fa, style des éditions vaticanes	`vaticana-fa1`, `vaticana-fa2`	
Clef d'ut, style Editio Medicaea	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	

Clef de fa, style Editio Medicaea	`medicaea-fa1`, `medicaea-fa2`	
Clef d'ut, style historique Hufnagel	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
Clef de fa, style historique Hufnagel	`hufnagel-fa1`, `hufnagel-fa2`	
Clef combinée ut/fa, style historique Hufnagel	`hufnagel-do-fa`	

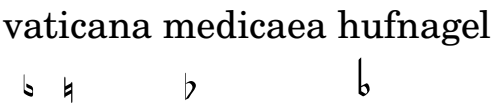
Voir aussi

Glossaire musicologique : Section "clef" dans *Glossaire*.

Manuel de notation : [Clefs], page 18.

Altérations et armures grégoriennes

LilyPond dispose d'altérations pour les trois styles grégoriens :



Vous noterez que chacun de ces styles ne comporte pas toutes les altérations. LilyPond changera de style s'il est besoin d'une altération indisponible dans le style utilisé.

Pour disposer des formes anciennes d'altération, utilisez la propriété `glyph-name-alist` des objets graphiques `Accidental` et `KeySignature`, comme ceci :

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

Voir aussi

Glossaire musicologique : Section "accidental" dans *Glossaire*, Section "key signature" dans *Glossaire*.

Manuel de notation : [Altérations], page 6, [Altérations accidentelles automatiques], page 29, [Armure], page 22, Section 1.1 [Hauteurs], page 1.

Référence des propriétés internes : Section "KeySignature" dans *Référence des propriétés internes*.

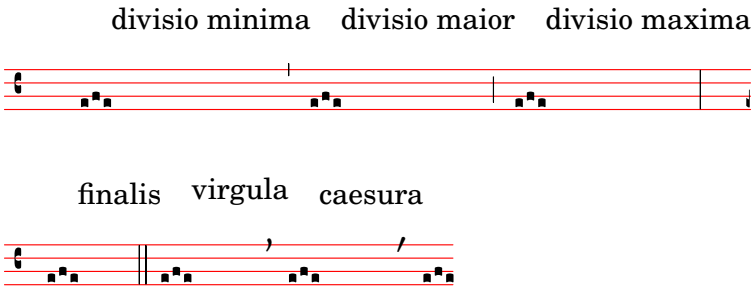
Divisions

Il n'existe pas de silence en notation grégorienne. On y parle plutôt de *divisions, pauses* ou *coupes*.

Une division – *divisio*, pluriel *divisiones* en latin – est un symbole ajouté à la portée et utilisé en chant grégorien pour séparer les phrases ou parties. *Divisio minima*, *divisio maior* et *divisio maxima* peuvent respectivement s'interpréter comme une pause courte, moyenne ou longue, à

l'image des marques de respiration — cf. [Signes de respiration], page 143. Le signe *finalis* n'est pas uniquement une marque de fin de chant ; il sert aussi à indiquer la fin de chaque partie dans une structure verset/répons.

Les divisions sont disponibles après inclusion du fichier `gregorian.ly`. Ce fichier définit les commandes `\divisioMinima`, `\divisioMaior`, `\divisioMaxima` et `\finalis`. Certaines éditions utilisent *virgula* ou *caesura* en lieu et place de *divisio minima* ; c'est pourquoi `gregorian.ly` définit aussi `\virgula` et `\caesura`.



Commandes prédéfinies

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Voir aussi

Glossaire musicologique : Section "caesura" dans *Glossaire*, Section "divisio" dans *Glossaire*.

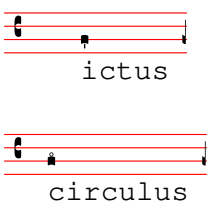
Manuel de notation : [Signes de respiration], page 143.

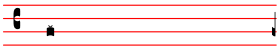
Fichiers d'initialisation : `gregorian.ly`.

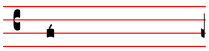
Articulations grégoriennes

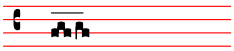
En plus des signes d'articulation standards décrits à la section [Articulations et ornements], page 127, LilyPond fournit des articulations spécifiquement destinées au style des éditions vaticanes.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript.font-family = #'typewriter
    \override TextScript.font-shape = #'upright
    \override Script.padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \episemInitium \pes b
       \flexa a b \episemFinis \flexa a \]
  }
}
```









Voir aussi

Manuel de notation : [Articulations et ornements], page 127.

Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Episema” dans *Référence des propriétés internes*, Section “Episema_engraver” dans *Référence des propriétés internes*, Section “EpisemaEvent” dans *Référence des propriétés internes*, Section “Script” dans *Référence des propriétés internes*, Section “Script_engraver” dans *Référence des propriétés internes*. Section “ScriptEvent” dans *Référence des propriétés internes*.
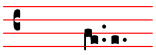
Problèmes connus et avertissements

Certaines articulations sont verticalement trop proches de leurs têtes de note.

Points d’augmentation (*morae*)

Les points d’*augmentum*, ou *morae*, s’obtiennent avec la fonction `\augmentum`. Notez que cette fonction `\augmentum` est implémentée en tant que fonction unaire plutôt que comme un préfixe de note. Par conséquent, `\augmentum \virga c` ne donnera rien de particulier. Il faut l’utiliser avec la syntaxe `\virga \augmentum c` ou `\augmentum {\virga c}`. Par ailleurs, l’expression `\augmentum {a g}` constitue une forme abrégée de `\augmentum a \augmentum g`.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Voir aussi

Manuel de notation : [Signes de respiration], page 143.

Référence des propriétés internes : Section “BreathingSign” dans *Référence des propriétés internes*.

Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.

Neumes et ligatures grégoriennes

Les neumes grégoriens, conformément au style des éditions vaticanes, sont pris en charge de façon assez limitée. Les ligatures élémentaires sont déjà disponibles, mais beaucoup de règles typographiques ne sont pas encore implémentées, notamment l’espacement horizontal des enchaînements de ligatures, l’alignement des paroles ou une gestion convenable des altérations.

La prise en charge des neumes grégoriens est activée par l’inclusion, en début de votre fichier source, du fichier d’initialisation `gregorian.ly`. Ceci aura pour effet de rendre disponible un

certain nombre de commandes dans le but de reproduire les symboles de neumes tels qu'ils apparaissent dans la notation du plain-chant.

Les têtes de note peuvent être *modifiées* ou *jointes*.

- L'aspect d'une tête de note se modifie en *préfixant* le nom d'une hauteur par l'une des commandes suivantes : `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Une ligature, autrement dit la juxtaposition de notes, s'obtient en plaçant une commande de jointure `\pes` ou `\flexa` pour marquer une ligne mélodique respectivement ascendante ou descendante, entre les notes qui la composent.

Une hauteur sans qualificatif sera considérée comme un *punctum*. Tout autre neume, y compris ceux d'une seule note d'aspect particulier comme la *virga*, sera considéré en tant que ligature et devra répondre à la syntaxe `\[...\]`.

Neumes simples

- Le *punctum* représente l'aspect standard d'une note – dans le style *Vaticana*, il s'agit d'un carré plein légèrement incurvé pour une question d'esthétique. Existe aussi le *punctum inclinatum* – carré penché qui s'obtient grâce au préfixe `\inclinatum`. Un *punctum* standard peut se modifier par le préfixe `\cavum` qui l'évidera, ou le préfixe `\linea`, qui lui adjoindra une ligne verticale de part et d'autre.
- La *virga* dispose d'une hampe sur la droite. Elle s'obtient à l'aide du modificateur `\virga`.

Ligatures

Contrairement à la majorité des autres systèmes de notation neumatique, la manière de saisir les neumes n'a rien à voir avec leur apparence typographique ; elle se concentre plutôt sur le sens musical. Ainsi, `\[ a \pes b \flexa g \]` produit un *torculus* constitué de trois *punctums*, alors que `\[ a \flexa g \pes b \]` produit un *porrectus* avec une flexe incurvée et un seul *punctum*. Il n'existe pas de commande à proprement parler qui permette de spécifier la courbe d'une flexe ; c'est la source musicale qui va le déterminer. Le fondement d'une telle approche réside dans la distinction que nous faisons entre les aspects musicaux de la source et le style de notation que nous voulons obtenir. De ce fait, la même source pourra être utilisée pour imprimer dans un autre style de notation grégorienne.

Neumes liquescents

Autre grande catégorie de notes que l'on trouve en grégorien, les neumes liquescents. Ils s'utilisent dans certaines circonstances, quand l'articulation d'une syllabe avec la suivante se fait par une « consonne liquide », sur la dernière note du groupe : M (omnis, summo), L, N, Y (ejus), NG (sanctus), W (autem, laudat). Ces consonnes ou semi-consonnes sont chantées à la hauteur correspondante, comme le seraient des voyelles, mais le chant est entravé par leur prononciation. De fait, les neumes liquescents ne sont jamais utilisés isolément (bien que rien ne l'interdise) et tombent toujours à la fin d'une ligature.

Les neumes liquescents peuvent se présenter graphiquement de deux façons différentes et relativement interchangeables : une note plus petite, ou une « bascule » verticale de la note principale. La première option s'obtient en créant un `pes` ou une `flexa` puis une modification de l'aspect de la deuxième note : `\[ a \pes \deminutum b \]`. La seconde option consiste à modifier l'aspect d'un neume sur note unique avec un `\auctum` tout en lui affectant une direction `\descendens` ou `\ascendens` : `\[ \auctum \descendens a \]`.

Signes spéciaux

Une troisième catégorie regroupe quelques signes dont la signification particulière diverge selon la source : la *quilisma*, l'*oriscus* et le *strophicus*. Ils s'obtiennent en préfixant la hauteur d'un `\quilisma`, `\oriscus` ou `\stropha`.

Il est virtuellement possible d'agglutiner autant de notes que voulu, y compris en les mélangeant avec des `\pes`, `\flexa`, `\virga`, `\inclinatum`, et de borner le tout par `\[` et `\]` pour produire une seule ligature. C'est d'ailleurs de cette manière que nous avons procédé pour générer le tableau qui suit. La création de ligatures est donc sans limite.

Notez bien que l'utilisation de ces signes en musique suit un certain nombre de règles, et que LilyPond n'effectue aucun contrôle à ce niveau. Par exemple, une *quilisma* se trouve toujours être la note intermédiaire d'une ligature ascendante et tombe habituellement sur un demi ton ; bien que cela soit tout à fait possible, mais parfaitement incorrect, rien ne vous empêche de créer une quilisma sur une seule note.

En plus des signes propres à la notation, le fichier `gregorian.ly` contient la définition des commandes `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ` et `\IIJ`, qui permettent par exemple d'indiquer dans les paroles des repères de section. Ces commandes font appel à des caractères unicode spécifiques qui ne seront reproduits que si vous utilisez une fonte qui en dispose.

Le tableau ci-dessous inventorie, bien que dans une certaine limite, les différents neumes contenus dans le second tome de l'Antiphonale Romanum (*Liber Hymnarius*) publié par l'abbaye de Solesmes en 1983. La première colonne énumère le nom des ligatures – forme normale en gras et forme liquescente en italique. La troisième colonne contient le code ayant permis de générer la ligature, se basant ici sur sol, la, si.

Neumes simples

Formes **Normale** et *Liquescente*	**Rendu**	**Code LilyPond**
Punctum		`\[ b \]`
		`\[ \cavum b \]`
		`\[ \linea b \]`
Punctum Auctum Ascendens		`\[ \auctum \ascendens b \]`
		
Punctum Auctum Descendens		`\[ \auctum \descendens b \]`
		

Punctum inclinatum

```
\[ \inclinatum b \]
```



Punctum Inclinatum Auctum

```
\[ \inclinatum \auctum b \]
```



Punctum Inclinatum Parvum

```
\[ \inclinatum \deminutum b \]
```


Virga

```
\[ \virga b' \]
```



Ligatures sur deux notes

Clivis vel Flexa

```
\[ b \flexa g \]
```



Clivis Aucta Descendens

```
\[ b \flexa \auctum \descendens g \]
```



Clivis Aucta Ascendens

```
\[ b \flexa \auctum \ascendens g \]
```



Cephalicus

```
\[ b \flexa \deminutum g \]
```



Podatus/Pes

```
\[ g \pes b \]
```



Pes Auctus Descendens

```
\[ g \pes \auctum \descendens b \]
```



Pes Auctus Ascendens

```
\[ g \pes \auctum \ascendens b \]
```



Epiphonus

```
\[ g \pes \deminutum b \]
```



Pes Initio Debilis

```
\[ \deminutum g \pes b \]
```



Pes Auctus Descendens Initio Debilis

```
\[ \deminutum g \pes \auctum
\descendens b \]
```



Ligatures sur plusieurs notes

Torculus

```
\[ a \pes b \flexa g \]
```



Torculus Auctus Descendens

```
\[ a \pes b \flexa \auctum
\descendens g \]
```



Torculus Deminutus

```
\[ a \pes b \flexa \deminutum g \]
```



Torculus Initio Debilis

```
\[ \deminutum a \pes b \flexa g \]
```



Torculus Auctus Descendens Initio Debilis

```
\[ \deminutum a \pes b \flexa \auctum
\descendens g \]
```



Torculus Deminutus Initio Debilis

```
\[ \deminutum a \pes b \flexa
\deminutum g \]
```



Porrectus

```
\[ a \flexa g \pes b \]
```



Porrectus Auctus Descendens

```
\[ a \flexa g \pes \auctum
\descendens b \]
```



Porrectus Deminutus

```
\[ a \flexa g \pes \deminutum b \]
```



Climacus

```
\[ \virga b \inclinatum a
\inclinatum g \]
```



Climacus Auctus

```
\[ \virga b \inclinatum a
\inclinatum \auctum g \]
```



Climacus Deminutus

```
\[ \virga b \inclinatum a
\inclinatum \deminutum g \]
```



Scandicus

```
\[ g \pes a \virga b \]
```



Scandicus Auctus Descendens

```
\[ g \pes a \pes
\auctum \descendens b \]
```



Scandicus Deminutus

```
\[ g \pes a \pes \deminutum b \]
```



Signes spéciaux

Quilisma

```
\[ g \pes \quilisma a \pes b \]
```



Quilisma Pes Auctus Descendens



```
\[ g \quilisma g \pes \auctum
\descendens b \]
```

Oriscus


```
\[ \oriscus b \]
```

Pes Quassus



```
\[ \oriscus g \pes \virga b \]
```

Pes Quassus Auctus Descendens



```
\[ \oriscus g \pes \auctum
\descendens b \]
```

Salicus



```
\[ g \oriscus a \pes \virga b \]
```

Salicus Auctus Descendens



```
\[ g \oriscus a \pes \auctum
\descendens b \]
```

(Apo)stropha



```
\[ \stropha b \]
```

Stropha Aucta



```
\[ \stropha \auctum b \]
```

Bistropha



```
\[ \stropha b \stropha b \]
```

Tristropha



```
\[ \stropha b \stropha b
\stropha b \]
```

Trigonus



```
\[ \stropha b \stropha b
\stropha a \]
```

Commandes prédéfinies

LilyPond dispose des préfixes suivants : `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Les préfixes de note peuvent s'agglutiner, modulo quelques restrictions. Par exemple, on peut appliquer un `\descendens` ou un `\ascendens` à une note, mais pas les deux simultanément à une même note.

Deux notes adjacentes peuvent être reliées grâce aux commandes `\pes` ou `\flexa` pour marquer une ligne mélodique respectivement ascendante ou descendante.

Utilisez la fonction musicale unaire `\augmentum` pour ajouter des points d'augmentum.

Voir aussi

Glossaire musicologique : Section "ligature" dans *Glossaire*.

Manuel de notation : [Ligatures], page 470, [Ligatures mensurales], page 478.

Problèmes connus et avertissements

Lorsqu'un `\augmentum` apparaît dans une ligature en fin de portée, son placement vertical peut être erroné. Pour y remédier, ajoutez un silence invisible, `s8` par exemple, comme dernière note de cette portée.

L'`\augmentum` devrait être implémenté en tant que préfixe plutôt qu'en tant que fonction unaire, afin qu'`\augmentum` puisse s'intégrer avec d'autres préfixes dans n'importe quel ordre.
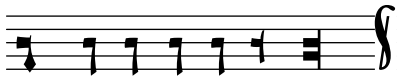
2.9.5 Typographie de notation kiévienne

Contextes de notation kiévienne

Tout comme pour les notations grégorienne et mensurale, les contextes prédéfinis `KievanVoice` et `KievanStaff` permettent de générer une partition en notation carrée. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant, comme ci-dessous :

```
% Font settings for Cyrillic
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Linux Libertine O,serif"
   ))
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        c4 c c c c2 b\longa
        \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Го -- спо -- ди по -- ми -- луй.
    }
  >>
}
```



Господи помилуй.

Voir aussi

Glossaire musicologique : Section “notation kiévienne” dans *Glossaire*.
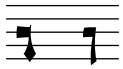
Problèmes connus et avertissements

LilyPond prend en charge la notation kiévienne du style synodal, correspondant au corpus du Saint Synode russe des années 1910, récemment réédité par les éditions du patriarcat de Moscou. LilyPond ne prend pas en charge les formes plus anciennes et moins répandues de notation kiévienne que l’on trouvait en Galicie pour noter le plain-chant ruthène.

Clefs kiéviennes

La notation kiévienne n’utilise qu’une seule clef – la clef « Tse-fa-ut » – qui indique la position du do :

```
\clef "kievan-do"
\kievanOn
c'
```



Voir aussi

Glossaire musicologique : Section “clef” dans *Glossaire*, Section “notation kiévienne” dans *Glossaire*.
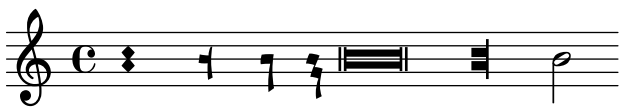
Manuel de notation : [Clefs], page 18.

Notes kiéviennes

La notation kiévienne requiert l’utilisation d’un style de tête de note particulier et la désactivation des hampes et crochets classiques. La fonction `\kievanOn` se charge d’affecter les propriétés adéquates aux têtes de note, hampes et crochets. Un simple `\kievanOff` permet de retrouver le comportement par défaut de LilyPond.

En notation kiévienne, la note finale d’une pièce apparaît souvent sous la forme d’une `\longa`. L’indication d’un récitatif – plusieurs syllabes sont chantées sur une même hauteur – s’effectue à l’aide d’une `\breve`. Voici ce à quoi ressemblent les différentes notes kiéviennes :

```
\autoBeamOff
\cadenzaOn
\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2
```



Voir aussi

Glossaire musicologique : Section “notation kiévienne” dans *Glossaire*, Section “tête de note” dans *Glossaire*.

Manuel de notation : Section A.9 [Styles de tête de note], page 740.
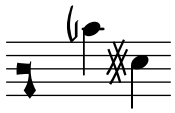
Problèmes connus et avertissements

LilyPond détermine automatiquement l'orientation des hampes. Il est cependant d'usage, en notation carrée, que les hampes des différentes notes d'un même mélisme aillent toutes dans le même sens ; il faudra donc en pareil cas définir manuellement la propriété `direction` de l'objet `Stem`.

Altérations kiéviennes

Le style d'altération propre à la notation kiévienne est accessible au travers de la propriété `glyph-name-alist` de l'objet `Accidental`. Le style `kievan` dispose d'un dièse et d'un bémol, tous deux différents du style par défaut ; il n'y a pas de bécarre en notation kiévienne. Bien que le dièse ne soit pas utilisé en notation synodale, on peut le trouver dans certains manuscrits plus anciens.

```
\clef "kievan-do"
\override Accidental.glyph-name-alist =
 #alteration-kievan-glyph-name-alist
bes' dis'
```



Voir aussi
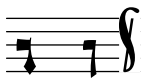
Glossaire musicologique : Section “altération” dans *Glossaire*, Section “notation kiévienne” dans *Glossaire*.

Manuel de notation : [Altérations], page 6, [Altérations accidentelles automatiques], page 29, Section A.8 [La fonte Emmentaler], page 718.

Barre de mesure kiévienne

Les pièces en notation kiévienne sont habituellement terminées par une décoration qui fait office de barre finale. Elle s'obtient à l'aide d'un `\bar "k"`.

```
\kievanOn
\clef "kievan-do"
c' \bar "k"
```



Voir aussi

Manuel de notation : [Barres de mesure], page 102, Section A.8 [La fonte Emmentaler], page 718.

Mélismes kiéviens

Les notes formant un mélisme kiévien sont habituellement rapprochées les unes des autres, les mélismes étant espacés les uns des autres. Ceci permet au chantre d'identifier aisément les structures mélodiques d'un chant *Znamenny*. Les mélismes sont traites par LilyPond comme des ligatures dont l'espacement est géré par le `Kievan_ligature_engraver`.

Le `Kievan_ligature_engraver` est activé par défaut pour les contextes `KievanVoice` et `KievanStaff`. Pour les autres contextes, il s'active au sein d'un bloc `layout` dans lequel est désactivé le `Ligature_bracket_engraver`.

```
\layout {
```
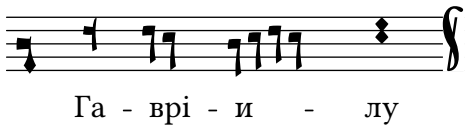
```
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Kievan_ligature_engraver"
  }
}
```

L'espacement des notes d'une ligature kiévienne se gère à l'aide de la propriété `padding` de `KievanLigature`.

Voici comment générer des ligatures en notation kiévienne :

```
% Font settings for Cyrillic
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Linux Libertine O,serif"
   ))
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        e2 \[ e4( d4 ) \] \[ c4( d e  d ) \] e1 \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Га -- вpi -- и -- лу
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

Manuel de notation : [Ligatures], page 470, [Ligatures mensurales], page 478, [Neumes et ligatures grégoriennes], page 483.

Problèmes connus et avertissements

L'espacement des ligatures n'est pas des meilleurs.

2.9.6 Réédition de musique ancienne

Travailler sur de la musique ancienne requiert bien souvent des tâches particulières et qui s'éloignent fortement de la notation moderne pour laquelle LilyPond est conçu. Nous allons aborder, au fil des paragraphes qui suivent, un certain nombre de cas particuliers et vous proposer des suggestions, voire des solutions aux problèmes que vous ne manquerez pas de rencontrer. Ceci inclut entre autres :
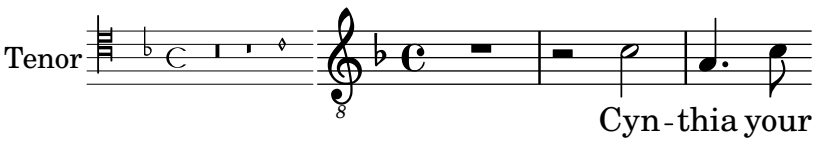
- comment réaliser un incipit, autrement dit un court extrait montrant ce à quoi ressemblait l'original, en introduction à la transcription d'une œuvre médiévale ;

- comment obtenir une présentation *Mensurstriche* comme on peut le voir dans nombre de transcriptions de musique polyphonique ;
- comment transcrire du grégorien en notation moderne ;
- comment obtenir à la fois une reproduction en notation ancienne et une édition en notation moderne à partir d'une même source.

Des incipits

Il est d'usage, lorsque l'on transcrit de la musique ancienne en notation moderne, d'indiquer aussi comment apparaissaient les silences ou notes initiaux dans la version originale, y compris la clef. Ceci s'appelle un *incipit*. La commande `\incipit` utilise le `indent` de la portée principale pour déterminer la place occupée par l'incipit, et `incipit-width` pour déterminer la longueur de la portée d'incipit.

```
\score {
  \new Staff <<
    \new Voice = Tenor {
      \set Staff.instrumentName = "Tenor"
      \override Staff.InstrumentName.self-alignment-X = #RIGHT
      \incipit { \clef "mensural-c4" \key f \major r\breve r1 c'1 }
      \clef "treble_8"
      \key f \major
      R1 r2 c'2 |
      a4. c'8
    }
    \new Lyrics \lyricsto Tenor { Cyn -- thia your }
  >>
  \layout
  {
    indent = 5\cm
    incipit-width = 3\cm
  }
}
```



Problèmes connus et avertissements

La propriété `instrumentName` doit se placer au sein de la musique de l'incipit à produire. Lorsqu'il n'y a pas de nom d'instrument, il faut cependant le définir avec `\set Staff.instrumentName = ""`.

Voir aussi

Morceaux choisis : Section “Notations anciennes” dans *Morceaux choisis*.

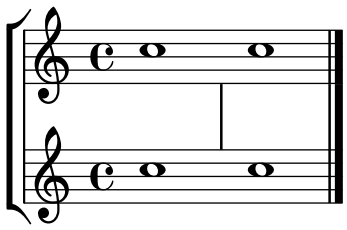
Mise en forme de la musique mensurale

Mensurstriche, pour « lignes de mensuration », est le terme consacré lorsque les barres de mesure apparaissent uniquement entre les portées d'un système. Cette présentation permet de préserver l'aspect rythmique de l'original – par exemple sans couper une syncope par l'apparition d'une barre – tout en procurant l'aide que peuvent constituer les barres de mesure.

En musique mensurale, les barres de mesure ne traversent pas les portées. Pour obtenir ce résultat avec un `StaffGroup` plutôt qu'en utilisant un `ChoirStaff`, il faudra masquer les portions de barre qui recouvrent les portées à l'aide d'un `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Transcription de chant grégorien

Une transcription d'un chant grégorien en notation moderne s'obtient grâce à quelques simples artifices.

Hampes. La gravure des hampes s'annule en supprimant le graveur `Stem_engraver` du contexte de voix :

```
\layout {
  ...
  \context {
    \Voice
      \remove "Stem_engraver"
  }
}
```

Temps. En matière de chant non mesuré, plusieurs alternatives s'offrent à vous.

La suppression du `Time_signature_engraver` du contexte `Staff` ne produit aucun effet négatif. Une alternative serait de rendre la métrique transparente, ce qui par contre préservera l'espace qu'elle occupe.

Dans de nombreux cas, une clause `\set Score.timing = ##f` donne de bons résultats. On pourrait aussi utiliser `\cadenzaOn` et `\cadenzaOff`.

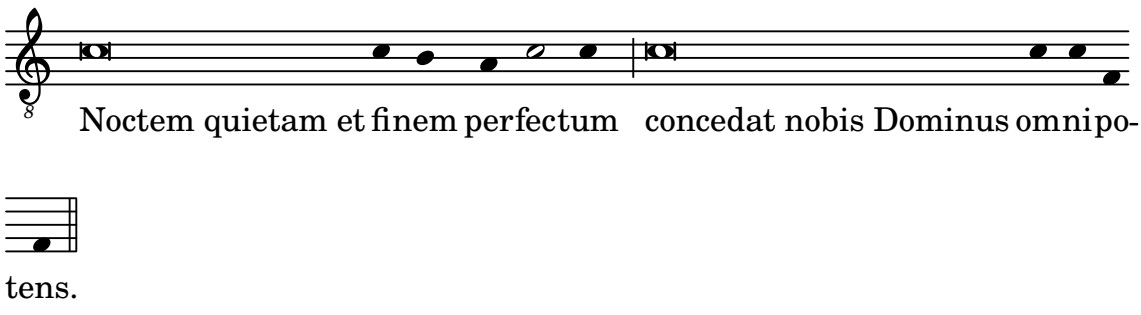
Rien de plus radical que de supprimer du contexte `Staff` le `Bar_engraver` pour ne pas voir de barre de mesure. Là aussi, une clause `\hide BarLine` vous permettra d'en afficher une au besoin.

Dans de nombreuses transcriptions, le récitatif fait apparaître une brève au lieu de la répétition d'une même note. Le texte psalmodié se présente alors sous la forme d'une unique syllabe alignée à gauche :

```
\include "gregorian.ly"
```

```
chant = \relative {
  \clef "G_8"
  c'\breve c4 b4 a c2 c4  \divisioMaior
  c\breve c4 c f, f \finalis
}

verba = \lyricmode {
  \once \override LyricText.self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText.self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}
\score {
  \new Staff <<
  \new Voice = "melody" \chant
  \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Voice
      \remove "Stem_engraver"
    }
  }
}
```



Ceci fonctionne bien tant que le texte ne risque pas de déborder de la ligne. Si tel était le cas, on pourrait plutôt ajouter des notes masquées comme ci-dessous.

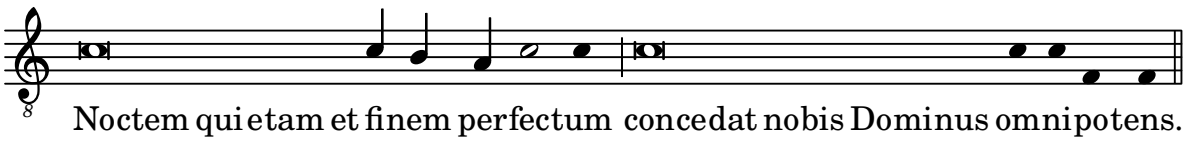
Certaines transcriptions laissent néanmoins apparaître occasionnellement des hampes, notamment pour indiquer la transition entre un récitatif monodique et une phrase mélodique. Il suffit en pareil cas d'utiliser plutôt `\hide Stem` ou `\override Stem.length = #0` puis, en cas de besoin, recourir à une clause `\once \override Stem.transparent = ##f`.

```
\include "gregorian.ly"
chant = \relative {
  \clef "G_8"
  \set Score.timing = ##f
  \hide Stem
  c'\breve \hide NoteHead  c c c c c
```

```
  \undo \hide NoteHead
  \undo \hide Stem \stemUp c4 b4 a
  \hide Stem c2 c4  \divisioMaior
  c\breve \hide NoteHead c c c c c c c
  \undo \hide NoteHead c4 c f, f \finalis
}

verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics \lyricsto "melody" \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \hide BarLine
    }
  }
}
```



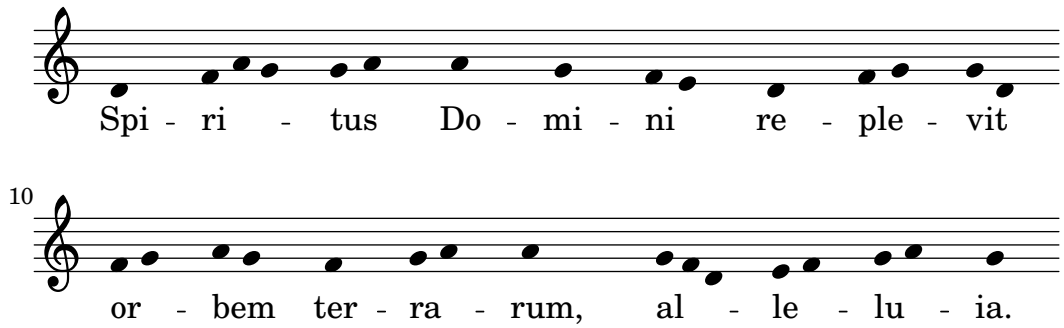
Autre situation courante, la transcription de chant neumatique contenant des mélismes, autrement dit, une psalmodie dans laquelle le nombre de syllabes varie selon les notes. Vous pourriez alors avoir envie d'indiquer clairement le découpage des groupes de syllabes ainsi que les subdivisions d'un mélisme. Le moyen pour y parvenir consiste à utiliser une métrique fixe, mettons `\time 1/4`, et de faire en sorte que chaque syllabe ou groupe de notes tienne dans une mesure, à l'aide de triolets ou de durées inférieures. Tant que les barres de mesure et autres éléments rythmiques restent transparents, et que l'espacement en regard des barres est accru, la représentation en notation moderne devrait être tout à fait satisfaisante.

Pour une répartition plus homogène de syllabes de longueur différente – telles que « -ri » et « -rum » – selon les groupes de note, une solution consiste à figer la propriété `'X-extent` de l'objet `LyricText`. Ceci s'avère moins fastidieux que d'ajouter des syllabes sous forme de *markup*. Des ajustements supplémentaires peuvent se réaliser avec des « notes silencieuses » (`s`).

```
spiritus = \relative {
  \time 1/4
  \override Lyrics.LyricText.X-extent  = #'(0 . 3)
  d'4 \tuplet 3/2 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4  s
  \tuplet 3/2 { g8 f d } e f g a g4
}

spirLyr = \lyricmode {
```

```
  Spi -- ri -- _ _ tus  _ Do -- mi -- ni  _ re -- ple -- _ vit _
  or -- _ bem _  ter -- ra -- _ rum, al -- _ _ le -- _ lu
  -- _ ia.
}
\score {
  \new Staff <<
    \new Voice = "chant" \spiritus
    \new Lyrics = "one" \lyricsto "chant" \spirLyr
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine.X-extent = #'(-1 . 1)
      \hide Stem
      \hide Beam
      \hide BarLine
      \hide TupletNumber
    }
  }
}
```



Éditions ancienne et moderne à partir d'une même source

Recours aux balises pour produire une partition ancienne et moderne à partir de la même source

Grâce aux balises (*tags*), il est possible d'utiliser une même source pour produire une partition de musique mensurale et moderne. Dans cet exemple est créée la fonction `menrest` qui permettra de positionner les silence comme dans la version originale, tout en respectant leur position sur une portée standard. Les balises permettent d'adapter les barres en fin de pièce ; elles peuvent aussi gérer d'autres différenciations selon les besoins, comme des « mesures de silence » (R1, R\breve, etc.) en notation moderne mais des silences normaux (r1, r\breve, etc.) en notation ancienne. L'action de convertir de la musique mensurale en version moderne est communément appelée « transcription ».

```
menrest = #(define-music-function (note)
  (ly:music?)
#{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
#})

MensStyle = {
```

```
  \autoBeamOff
  \override NoteHead.style = #'petrucci
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
}

finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
  \once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

  \breathe
}

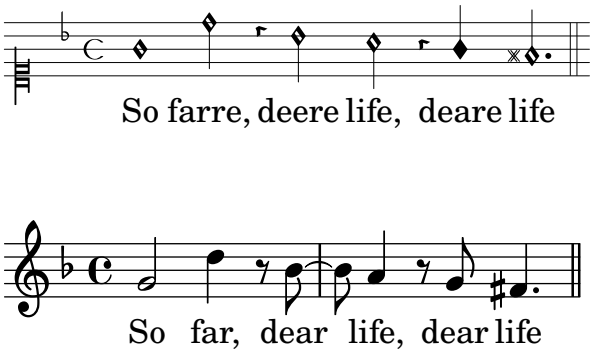
Music = \relative c'' {
  \set Score.tempoHideNote = ##t
  \key f \major
  \time 4/4
  g1 d'2 \menrest bes4 bes2 a2 r4 g4 fis2.
  \tag #'mens { \finalis }
  \tag #'mod { \bar "||" }
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {
  \keepWithTag #'mens {
    <<
      \new MensuralStaff
      {
        \new MensuralVoice = Cantus
          \clef "mensural-c1" \MensStyle \Music
      }
      \new Lyrics \lyricsto Cantus \MenLyr
    >>
  }
}

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff
      {
        \new Voice = Sop \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver" }
        {
          \shiftDurations #1 #0 { \autoBeamOff \Music }
```

```
        }
      }
      \new Lyrics \lyricsto Sop \ModLyr
    >>
  }
}
```



2.10 Musiques du monde

Ce chapitre a pour objet la notation des musiques traditionnelles autres qu'occidentales.

2.10.1 Noms des notes et altérations non-occidentaux

Nous allons voir ici comment saisir et imprimer des partitions dans d'autres formes que la musique occidentale, que les anglophones appellent aussi *Common practice period*.

Extension des systèmes de notation et d'accordage

Les formes de notation propres à la musique classique traditionnelle sont employées dans toutes sortes de musique autres que le « classique ». Nous en avons déjà parlé dans le chapitre Section 1.1.1 [Écriture des hauteurs de note], page 1, et plus particulièrement à la rubrique [Nom des notes dans d'autres langues], page 8.

De nombreuses musiques autres qu'occidentales – et même certaines formes de musique traditionnelle occidentales – ont cependant recours à des systèmes de notation alternatifs ou étendus, qui ne s'intègrent pas forcément dans notre système standard.

Dans certains cas où la notation standard est utilisée, ces différences de hauteur seront implicites. Par exemple, la musique arabe est reproduite en notation standard et utilise des quarts de ton, l'altération réelle dépendant du contexte. Elle utilise traditionnellement la dénomination italienne, étendue dans le fichier `arabic.ly` par un certain nombre de macros – voir Section 2.10.2 [Musique arabe], page 501, pour plus de détails.

D'autres, par contre, font appel à une notation étendue, voire toute particulière. La *musique classique turque*, ou musique ottomane, utilise des formes mélodiques appelées *makamlar*, dans laquelle les tons sont divisés en neuf intervalles. Du point de vue actuel des pratiques de notation, il est possible d'utiliser les notes occidentales (do, ré, mi. . .) auxquelles on ajoutera l'altération spécifique à la musique turque. Ces différentes altérations sont définies dans le fichier `turkish-makam.ly`. Pour de plus amples informations, reportez-vous à la rubrique Section 2.10.3 [Musique classique turque], page 506.

Pour savoir où se trouvent les fichiers `hel-arabic.ly` et `makam.ly` sur votre système, reportez vous au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*.

Voir aussi

Glossaire musicologique : Section “Common Practice Period” dans *Glossaire*, Section “makamlar” dans *Glossaire*.

Manuel d’initiation : Section “Autres sources de documentation” dans *Manuel d’initiation*.

Manuel de notation : Section 1.1.1 [Écriture des hauteurs de note], page 1, Section 2.10.2 [Musique arabe], page 501, Section 2.10.3 [Musique classique turque], page 506, [Nom des notes dans d’autres langues], page 8.

2.10.2 Musique arabe

Ce chapitre souligne les questions propres à la notation de la musique arabe.

Références pour la musique arabe

Jusqu’à nos jours, la musique arabe a principalement été transmise comme une tradition orale. Lorsqu’elle était transcrite, c’était en général sous forme de canevas sur lequel le rôle des interprètes était d’improviser substantiellement. La notation occidentale, cependant, est de plus en plus utilisée, avec quelques variations, pour transmettre et préserver la musique arabe.

Certains éléments de notation musicale occidentale, tels que les transcriptions d’accords ou de parties indépendantes, ne sont pas nécessaires pour retranscrire les pièces arabes les plus traditionnelles. Il y a cependant quelques besoins spécifiques, tels que des intervalles se trouvant entre le demi-ton et le ton qui s’ajoutent aux intervalles mineurs ou majeurs utilisés dans la musique occidentale. Il est également nécessaire de regrouper et de noter un grand nombre de maqams (modes) différents qui font partie de la musique arabe.

En général, la notation de la musique arabe n’essaie pas d’indiquer précisément les micro-intervalles intervenant dans la pratique musicale.

Plusieurs particularités propres à la musique arabe sont traitées ailleurs :

- Les noms des notes et altérations (y compris les quarts de tons) peuvent être adaptés comme l’explique Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.
- Les armures peuvent également être adaptées comme expliqué dans [Armure], page 22.
- Des métriques complexes peuvent nécessiter de grouper les notes manuellement, comme décrit dans [Barres de ligature manuelles], page 99.
- Les *Takasim*, qui sont des improvisations rythmiquement libres, peuvent être écrites en omettant les barres de mesures, de la façon indiquée dans [Musique sans métrique], page 78.

Voir aussi

Manuel de notation : [Armure], page 22, [Barres de ligature manuelles], page 99, Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.
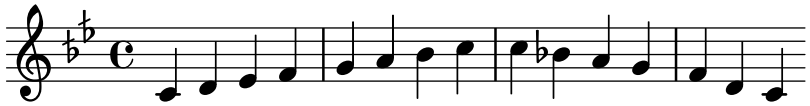
Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*.

Noms des notes en arabe

Les noms de note les plus traditionnels en arabe peuvent être très longs et ne conviennent pas à l’écriture de la musique.
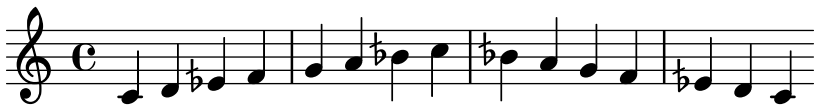
Les noms de note anglais sont accessibles à travers le fichier `hel-aravic.ly`. Voici comment écrire une gamme arabe *rast* :

```
\include "hel-arabic.ly"
\relative {
\key c \rast
  c' d edb f | g a bdb c | c bb a g | f d c
}
```



On peut aussi utiliser les noms italiens (`do, re, mi, fa, sol, la, si`) à l'aide du fichier `arabic.ly`. Par exemple, voici comment on peut écrire la gamme arabe *rast* :

```
\include "arabic.ly"
\relative {
  do' re misb fa | sol la sisb do | sisb la sol fa | misb re do
}
```
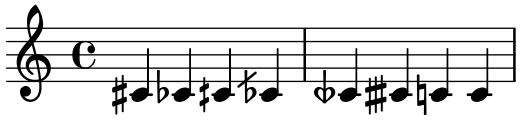


Le « rast » est une gamme heptatonique qui utilise des quarts de ton et est considéré comme la gamme central et la plus importante du « maqamat arabe ». Pour une liste exhaustive des gammes arabes disponibles, consultez les fichiers `hel-arabic.ly` ou `arabic.ly` tous deux inclus avec LilyPond.

L'utilisation des standards occidentaux pour noter la musique non occidentale est abordée dans Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500 ; consulter aussi [Nom des notes dans d'autres langues], page 8,

Le symbole indiquant un demi-bémol ne correspond pas au symbole utilisé dans la notation arabe. Si le symbole particulier du demi-bémol arabe doit absolument être utilisé, il est possible de s'en approcher en faisant précéder la note par la commande `\dwn` définie dans le fichier `arabic.ly`. Cette méthode ne peut toutefois pas être utilisée pour modifier l'aspect du demi-bémol dans l'armure.

```
\include "arabic.ly"
\relative {
  \set Staff.extraNatural = ##f
  dod' dob dosd \dwn dob dobsb dodsd do do
}
```



Voir aussi

Manuel de notation : Section 3.3.1 [Insertion de fichiers LilyPond], page 541, [Nom des notes dans d'autres langues], page 8, Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*.

Armures arabes

Outre les armures mineures et majeures, les armures définies dans les fichiers `arabic.ly` et `hel-arabic` déterminent un certain nombre de groupes de maqams.

En général, un maqam utilise l'armure de son groupe ou d'un groupe voisin et diverses altérations accidentelles sont indiquées tout au long de la musique. Les maqams arabes autorisent peu de modulations en raison de la nature des instruments arabes.

Par exemple, pour indiquer l'armure d'une pièce en maqam muhayer :

```
\key re \bayati
```

Ici, *re* est la tonalité par défaut du muayer maqam, et *bayati* le nom du maqam de base dans le groupe.

Alors que l'armure correspond à un groupe, il est courant que le titre mentionne un maqam en particulier. Ainsi, dans cet exemple, le titre devrait faire apparaître le nom du maqam muhayer.

D'autres maqams du même groupe bayati, comme l'explique le tableau ci-dessous (*bayati*, *hussaini*, *saba*, et *ushaq*), peuvent être indiqués de la même manière. Ils sont autant de variations du maqam de base, le plus courant, du groupe (en l'occurrence, *bayati*). En général, c'est dans les tétracordes supérieurs que ces modes apparentés diffèrent, ou dans certains détails de disposition qui ne changent pas fondamentalement leur nature.

Certains maqams ne sont qu'une modulation de leur maqam de base. Ainsi, dans ce même groupe de *bayati*, du maqam *nawa*, dont la modulation est indiquée entre parenthèses dans le tableau. Les maqams arabes n'admettent que des modulations limitées, en raison de la nature des instruments de musique arabes. Le *nawa* peut être indiqué comme suit :

```
\key sol \bayati
```

En musique arabe, le terme utilisé pour désigner un groupe maqam, tel que *bayati*, est également lui-même un maqam, généralement le plus important dans le groupe ; on peut le considérer comme un maqam de base.

Voici une suggestion de groupement qui relie les maqams les plus courants à leur armure :

groupe maqam	**Armure**	**Tonique**	**Autres maqams dans le groupe (tonique)**
ajam	major	sib	jaharka (fa)
bayati	bayati	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	kurd	re	shahnaz, shad arban (sol), hijazkar (do)
irakien	iraq	sisb	-
kurde	kurd	re	kurde hijazkar (do)
nahawand	minor	do	busalik (re), farah faza (sol)
nakriz	mineur	do	nawa athar, hisar (re)
rast	rast	do	mahur, yakah (sol)
sikah	sikah	misb	huzam

Morceaux choisis

Armures inhabituelles

La commande `\key` détermine la propriété `keyAlterations` d'un contexte `Staff`. Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété.

Il s'agit en l'occurrence de définir une liste :
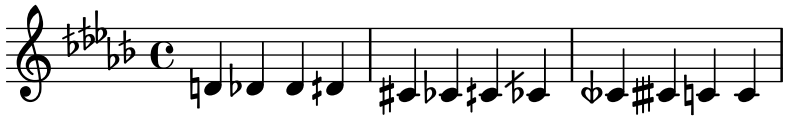
```
\set Staff.keyAlterations =
  #`(((octave . pas) . altération) ((octave . pas) . altération)
  @dots{})
```

dans laquelle, et pour chaque élément, `octave` spécifie l'octave (0 pour celle allant du do médium au si supérieur), `pas` la note dans cette octave (0 pour do et 6 pour si), et `altération` sera `,SHARP` ou `,FLAT` ou `,DOUBLE-SHARP`, etc. (attention à la virgule en préfixe).

Une formulation abrégée – `(pas . altération)` – signifie que l'altération de l'élément en question sera valide quelle que soit l'octave. En ce qui concerne les gammes microtonales dans lesquelles un « dièse » n'est pas d'un centième, `altération` se réfère à un deux-centième de ton entier.

```
\include "arabic.ly"
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
```

```
                                (1 . ,SEMI-FLAT)
                                (2 . ,FLAT)
                                (5 . ,FLAT)
                                (6 . ,SEMI-FLAT))
%\set Staff.extraNatural = ##f
  re reb \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dodsd do do |
}
```



Voir aussi

Glossaire musicologique : Section “maqam” dans *Glossaire*, Section “bayati” dans *Glossaire*, Section “rast” dans *Glossaire*, Section “sikah” dans *Glossaire*, Section “iraq” dans *Glossaire*, Section “kurd” dans *Glossaire*.

Manuel d’initiation : Section “Hauteurs et armure” dans *Manuel d’initiation*.

Manuel de notation : [Armure], page 22.

Fichiers d’initialisation : `ly/arabic.ly`, `ly/hel-arabic.ly`.

Référence des propriétés internes : Section “KeySignature” dans *Référence des propriétés internes*.

Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*, Section “Hauteurs” dans *Morceaux choisis*.

Métriques arabes

Quelques formes de musique classique arabes et turques telles que *Semai* utilisent des métriques inhabituelles comme le 10/8. Ceci peut impliquer une manière de grouper les notes fort différente de la musique écrite existante, où les notes ne sont pas groupées par temps mais d’une façon difficile à reproduire automatiquement. Il est possible d’y remédier en désactivant la ligature automatique et en groupant les notes manuellement. Lorsque l’enjeu n’est pas de reproduire exactement un texte existant, il est toujours possible d’ajuster le comportement de ligature automatique ou d’utiliser des chiffres de mesure composés.

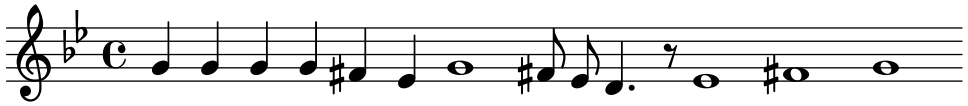
Morceaux choisis

Improvisation en musique arabe

Lorsque les improvisations ou *taqasim* sont temporairement libres, la métrique peut ne pas apparaître, auquel cas on utilisera un `\cadenzaOn`. Les altérations accidentelles devront alors être répétées en raison de l’absence de barre de mesure. Voici comment pourrait débuter une improvisation de *hijaz*.

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Voir aussi

Glossaire musicologique : Section “semai” dans *Glossaire*, Section “taqasim” dans *Glossaire*.

Manuel de notation : [Altérations accidentelles automatiques], page 29, [Barres de ligature automatiques], page 87, [Barres de ligature manuelles], page 99, [Définition des règles de ligature automatique], page 90, [Métrique], page 69, [Musique sans métrique], page 78.

Fichiers d’initialisation : `ly/arabic.ly`, `ly/hel-arabic.ly`.
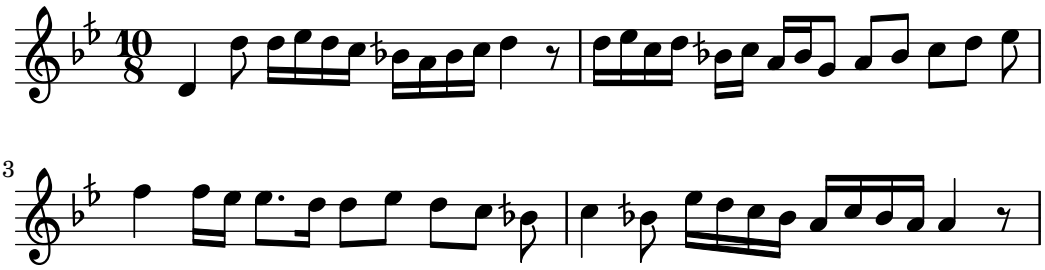
Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*.

Exemple de musique arabe

Voici un modèle qui utilise également le début d’un *semai* turc courant dans l’éducation musicale arabe, pour illustrer quelques unes des particularités de la notation musicale arabe, comme des intervalles intermédiaires et des modes inhabituels traités dans ce chapitre.

```
\include "arabic.ly"
\score {
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
  \relative {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re'4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re  [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
}
```



Voir aussi

Fichiers d’initialisation : `ly/arabic.ly`, `ly/hel-arabic.ly`.

Morceaux choisis : Section “Musiques du mondes” dans *Morceaux choisis*.

Lectures complémentaires pour la musique arabe

Si tout le monde s’accorde à apparenter les maqams d’après leur tétracorde inférieur, parfois transposé, les méthodes de classification varient dans certains détails. Les sources ne sont pas entièrement cohérentes (parfois dans un même texte) quant à la manière d’indiquer l’armure

de certains maqams. Il est courant, cependant, d'utiliser une armure par groupe plutôt qu'une armure différente pour chaque maqam.

- *La musique des Arabes* par Habib Hassan Touma [Amadeus Press, 1996], contient une étude des maqams et leur méthode de classification.
- Il existe également de nombreux sites web qui expliquent les maqams, dont quelques uns s'accompagnent d'exemples audio :
  - `https://www.maqamworld.com/`
  - `https://www.turath.org/`
- Des méthodes de luth arabe, l'oud, par les auteurs suivants, contiennent des exemples de compositions principalement turques et arabes,
  - Charbel Rouhana
  - George Farah
  - Ibrahim Ali Darwish Al-masri

2.10.3 Musique classique turque

Ce chapitre met en évidence des questions propres à la notation de la musique classique turque.

Références pour la musique classique turque

La musique classique turque s'est développée dans l'Empire Ottoman à peu près à la même période que la musique classique en Europe, et a continué jusqu'aux XX^{e} et XXI^{e} siècles comme une tradition vibrante et distincte avec sa propre théorie, ses propres formes et styles d'interprétation. Parmi ses caractéristiques remarquables, se trouve l'usage de micro-intervalles fondés sur des « commas » d'un neuvième de ton, dont sont dérivées les formes mélodiques *makam* (pluriel *makamlar*).

Quelques questions relatives à la musique classique turque sont traitées dans d'autres chapitres. Les noms de notes et altérations sont mentionnés dans Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Noms de note en turc

La musique classique turque attribue traditionnellement un nom unique à chaque hauteur, et du fait de la division du ton en neuf parts, les makamlar emploient une échelle de hauteurs complètement différente des gammes et modes d'occident :
koma de 1/9 de ton entier, *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (un ton entier) et *artık ikili* (12/9 ou 13/9 de ton).

D'un point de vue de notation moderne, il est pratique d'utiliser les positions occidentales des notes sur la portée (do, ré, mi. . .) avec des altérations spéciales qui haussent ou baissent les notes par intervalles de 1/9, 4/9, 5/9 et 8/9 de ton. Ces altérations sont définies dans le fichier `turkish-makam.ly`.

Pour plus d'information sur les formes de notation non-occidentales, reportez-vous au chapitre Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Voir aussi

Glossaire musicologique : Section “makam” dans *Glossaire*, Section “makamlar” dans *Glossaire*.

Manuel de notation : Section 2.10.1 [Noms des notes et altérations non-occidentaux], page 500.

Armures turques

LilyPond prend en charge plus de 200 définitions d'armures de makam – nettement moins que ce qui peut être utilisé en musique classique turque –, chaque maqam disposant de sa propre tonique et hauteur finale (appelée « karar » en turc).

Il est important de garder à l'esprit la hauteur finale de chaque makam. Voici un esmemple où *g* (sol) est la tonique par défaut et *rast* le nom du makam.

```
\key g \rast
```

Les altérations adéquates, koma bémol (*b1*) et koma dièse (`f4`), tous deux en relation avec la tonique *g*, seront affichés automatiquement.

Morceaux choisis

Exemple de makam turc

Ce canevas utilise le début d'un *saz semai* turc bien connu du répertoire aux fins d'illustrer certains éléments de la notation musicale turque.

```
% Initialize makam settings
\include "turkish-makam.ly"

\header {
    title = "Hüseyni Saz Semaisi"
    composer = "Lavtacı Andon"
}

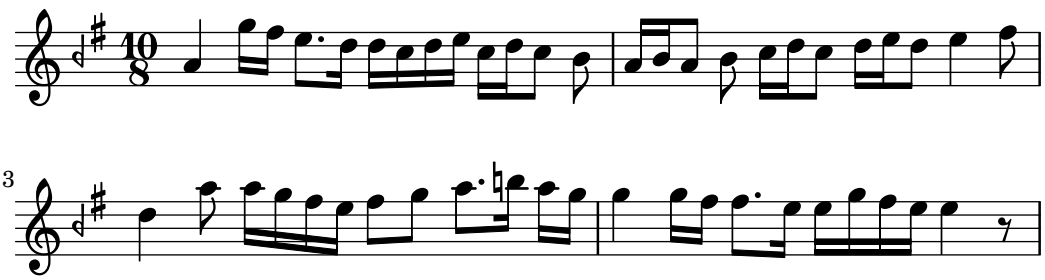
\relative {
  \set Staff.extraNatural = ##f
  \set Staff.autoBeaming = ##f

  \key a \huseyni
  \time 10/8

  a'4 g'16 [fb] e8. [d16] d [c d e] c [d c8] bfc |
  a16 [bfc a8] bfc c16 [d c8] d16 [e d8] e4 fb8 |
  d4 a'8 a16 [g fb e] fb8 [g] a8. [b16] a16 [g] |
  g4 g16 [fb] fb8. [e16] e [g fb e] e4 r8 |
}
```

Hüseyni Saz Semaisi

Lavtacı Andon



Lectures complémentaires pour la musique turque

- *Turk Musikisi Nazariyati ve Usulleri: Kudum Velveleleri* par Ismail Hakki Ozkan [(Kultur serisi, 41) (en turc) Paperback – 1986]

  contient des informations sur la théorie des makams et du usul.
- *Music of the Ottoman Court* par Walter Feldman [VWB Hardback – 1996]

  contient des informations sur l'histoire de la musique de cour ottomane.

- *Turkish Music Makam Guide* par Murat Aydemir [Pan Paperback – 2010]
  contient des informations en anglais sur les makams turcs et inclut deux disques compacts.

3 Généralités en matière d'entrée et sortie

Nous n'allons pas, dans ce chapitre, parler directement de notation, mais plutôt du contenu des fichiers source et du résultat produit par LilyPond.

3.1 Agencement du code

LilyPond traite des fichiers textuels. Ces fichiers portent par convention une extension `.ly`.

3.1.1 Structure d'une partition

Un bloc `\score` contient obligatoirement une seule expression musicale délimitée par des accolades :
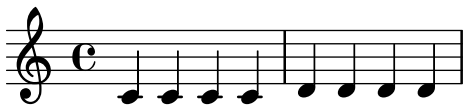
```
\score {
...
}
```

> **Note :** Il ne doit y avoir qu'**une seule** expression musicale globale dans un bloc `\score`, et elle **doit** être bornée par une paire d'accolades.

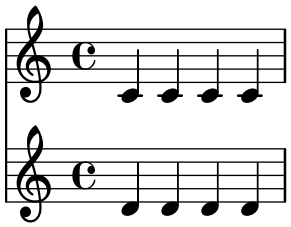
Cette unique expression musicale peut être de n'importe quelle taille et contenir d'autres expressions musicales aussi complexes soient elles. Voici quelques exemples d'expression musicale :

```
{ c'4 c' c' c' }
```

```
{
  { c'4 c' c' c' }
  { d'4 d' d' d' }
}
```



```
<<
  \new Staff { c'4 c' c' c' }
  \new Staff { d'4 d' d' d' }
>>
```



```
{
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { \flute }
      \new Staff { \hautbois }
    >>
    \new StaffGroup <<
      \new Staff { \violonI }
      \new Staff { \violonII }
```

```
    >>
  >>
}
```

Les commentaires constituent l'une des rares exceptions à cette règle immuable – voir Section 3.1.5 [Structure de fichier], page 513, pour les autres. Qu'il s'agisse d'une seule ligne ou de tout un bloc – délimité par `%{ ... %}` – un commentaire peut se placer n'importe où dans le fichier source, aussi bien à l'intérieur qu'à l'extérieur du bloc `\score`, ou encore à l'intérieur ou à l'extérieur de l'expression musicale contenue dans un bloc `\score`.

Lorsqu'un fichier ne comprend qu'un bloc `\score`, celui-ci est implicitement inclus dans un bloc `\book`. Le bloc `\book` d'un fichier source permet la production d'au moins un fichier dont le nom sera, par défaut, déduit du fichier source : le traitement de `fandangopourelephants.ly` produira donc `fandangopourelephants.pdf`.

Pour de plus amples informations à propos du bloc `\book`, lisez Section 3.1.2 [Plusieurs partitions dans un même ouvrage], page 510, Section 3.1.3 [Plusieurs éditions pour une même source], page 511, et Section 3.1.5 [Structure de fichier], page 513.

Voir aussi

Manuel d'initiation : Section "La partition est une (unique) expression musicale composée" dans *Manuel d'initiation*, Section "Les expressions musicales en clair" dans *Manuel d'initiation*, Section "Travail sur les fichiers d'entrée" dans *Manuel d'initiation*.

3.1.2 Plusieurs partitions dans un même ouvrage

Un ouvrage peut se composer de plusieurs morceaux et de texte. C'est le cas des cahiers d'exercices ou d'une partie d'orchestre avec ses différents mouvements. Chaque mouvement fait l'objet d'un bloc `\score`,

```
\score {
  ...musique...
}
```

et le texte est contenu dans un bloc `\markup`,

```
\markup {
  ...texte...
}
```

Les différents mouvements et textes qui apparaissent dans un même fichier `.ly` ne composeront en principe qu'un seul fichier résultant.

```
\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}
```

Attention cependant si vous travaillez avec lilypond-book : il vous faudra explicitement mentionner le bloc `\book`, en l'absence de quoi seul le premier `\score` ou `\markup` apparaîtra après traitement.

L'entête de chaque pièce peut se placer au sein du bloc `\score` ; le contenu du champ `piece` viendra s'imprimer avant chaque mouvement. De même, le titre de l'ouvrage peut se placer au

sein du bloc `\book`. Dans le cas contraire, le contenu du bloc `\header` placé en début de fichier sera utilisé.

```
\header {
  title = "Huit miniatures"
  composer = "Igor Stravinsky"
}
\score {
  \header { piece = "Romance" }
  ...
}
\markup {
   ...texte du second couplet...
}
\markup {
   ...texte du troisième couplet...
}
\score {
  \header { piece = "Menuet" }
  ...
}
```

Plusieurs pièces seront regroupées dans un même « chapitre » à l'aide d'un bloc `\bookpart`. Ces différents « chapitres » sont séparés par un saut de page et peuvent comporter un titre à l'instar de l'ouvrage dès lors que vous y insérez un bloc `\header`.

```
\bookpart {
  \header {
    title = "Titre de l'ouvrage"
    subtitle = "Première partie"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Deuxième partie"
  }
  \score { ... }
  ...
}
```

3.1.3 Plusieurs éditions pour une même source

Dès lors que vous inscrivez plusieurs blocs `\book` dans un même fichier `.ly`, chacun d'eux donnera lieu à un résultat indépendant. Lorsqu'aucun bloc `\book` n'est spécifié dans le fichier source, LilyPond considère que l'intégralité du fichier constitue un bloc `\book` unique, comme indiqué à la rubrique Section 3.1.5 [Structure de fichier], page 513.

LilyPond fait en sorte, lorsque plusieurs fichiers sont produits à partir d'une même source, qu'aucun résultat d'un bloc `\book` n'écrase celui qui a été généré pour un bloc `\book` précédent.

Dans les faits, et si le nom du fichier produit est repris de sa source – comportement par défaut –, un suffixe lui sera ajouté pour chaque `\book`. Il s'agit en principe d'un pseudo numéro de version. Ainsi, le fichier `huitminiatures.ly` qui contiendrait

```
\book {
```

```
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
```

générera

```
huitminiatures.pdf,
huitminiatures-1.pdf
huitminiatures-2.pdf.
```

3.1.4 Nom des fichiers de sortie

LilyPond vous permet de prendre le contrôle dans la dénomination des fichiers que vous voulez générer, quel que soit le moteur de rendu utilisé.

Nous avons vu dans la rubrique précédente que LilyPond évite les conflits de nom des fichiers qu'il génère à partir d'une même source. Vous pouvez même définir vous-même le suffixe qui sera appliqué à chacun des blocs `\book`. Ainsi, en reprenant l'exemple ci-avant, vous obtiendrez les fichiers

```
huitminiatures-Romance.pdf
huitminiatures-Menuet.pdf
huitminiatures-Nocturne.pdf
```

en ajoutant simplement une déclaration `\bookOutputSuffix` au sein de chaque bloc `\book`.

```
\book {
  \bookOutputSuffix "Romance"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Menuet"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \paper { ... }
}
```

La déclaration `\bookOutputName` vous permet de définir vous-même le nom du fichier généré pour un bloc `\book` :

```
\book {
  \bookOutputName "Romance"
  \score { ... }
  \paper { ... }
}
\book {
```

```
  \bookOutputName "Menuet"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
  \paper { ... }
}
```

Le traitement de ce fichier produira :

```
Romance.pdf
Menuet.pdf
Nocturne.pdf
```

3.1.5 Structure de fichier

Un fichier `.ly` peut contenir un certain nombre d'expressions de haut niveau. Les expressions de haut niveau sont les suivantes :

- Une définition de sortie, comme `\paper`, `\midi` et `\layout`. Ces définitions, lorsqu'elles se trouvent à un niveau supérieur, s'appliqueront à l'intégralité de l'ouvrage. Si l'une de ces expression apparaît à plusieurs reprises à un niveau supérieur, les différents contenus seront combinés, à ceci près qu'en cas de déclarations conflictuelles, la dernière aura préséance. Des informations complémentaires sont disponibles à la rubrique Section 4.2.1 [Le bloc `\layout`], page 583.
- Une expression Scheme pure, telle que `#(set-default-paper-size "a7" 'landscape)` ou `#(ly:set-option 'point-and-click #f)`.
- Un bloc `\header`, dont le contenu sera valide pour tout le fichier. Il comporte en général les valeurs par défaut des champs de titrage, tels le titre ou l'auteur entre autres, communs à tous les blocs `\book` inclus dans le fichier – voir [Généralités en matière de titrages], page 515.
- Un bloc `\score` pour la partition. Cette partition sera assemblée avec les autres partitions se trouvant au même niveau pour composer le `\book`. Vous pouvez modifier ce comportement à l'aide de la variable `toplevel-score-handler` placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation `../scm/lily.scm`, et les réglages par défaut dans le fichier `../ly/declarations-init.ly`.
- Un bloc `\book` permet de regrouper naturellement plusieurs mouvements – autrement dit plusieurs blocs `\score` – dans un même document. Lorsqu'il y a plusieurs `\score`, LilyPond génère un seul fichier dans lequel les mouvements sont mis les uns à la suite des autres, ce pour chacun des blocs `\book` rencontrés. La seule raison qui peut vous demander d'expliciter plusieurs blocs `\book` dans un fichier `.ly` est lorsque vous avez besoin de générer différents documents à partir d'une même source. La présence explicite d'un bloc `\book` est aussi nécessaire lorsque vous travaillez sur un document lilypond-book qui reprendrait plusieurs `\score` ou `\markup` dans un même extrait. Vous pouvez modifier ce comportement à l'aide de la variable `toplevel-book-handler` placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation `../scm/lily.scm`.
- Un bloc `\bookpart`. Un ouvrage peut se découper en plusieurs parties à l'aide de blocs `\bookpart`, aussi bien pour alléger le travail de l'algorithme de calcul des sauts de page, que si les réglages du bloc `\paper` diffèrent d'une partie à l'autre.
- Une expression musicale telle que

  ```
  { c'4 d' e'2 }
  ```

Ce bout de code sera placé dans un `\score` et intégré à l'ouvrage en même temps que tous les autres `\score` ou expressions musicales. En d'autres termes, un fichier qui ne contiendrait que cette simple expression musicale sera traduit en

```
\book {
  \score {
    \new Staff {
      \new Voice {
        { c'4 d' e'2 }
      }
    }
    \layout { }
  }
  \paper { }
  \header { }
}
```

Vous pouvez modifier ce comportement à l'aide de la variable `toplevel-music-handler` placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation `../scm/lily.scm`.

- Du texte sous forme de *markup* comme les paroles d'un couplet

```
\markup {
   2.  Le première ligne du deuxième couplet.
}
```

De tels *markups* seront imprimés là où ils apparaissent, avant, après ou entre les expressions musicales.

- Une variable, ou identificateur, telle que

```
toto = { c4 d e d }
```

Vous pourrez la réutiliser plus loin dans votre fichier en saisissant simplement `\toto`. Le nom des identificateurs ne doit pas comporter de chiffre (ASCII), de succession de caractères souligné (`_`) ou de tiret ni aucune espace. Tous les autres caractères Unicode sont permis, aussi bien latins, grecs, chinois que cyrilliques. Les souligné ou tiret isolés sont autorisés. Autrement dit, les variables `CorIII` ou $\alpha\beta\gamma$`XII` sont valides.

Toute combinaison de caractères est permise dès lors que le nom de la variable est borné par des guillemets informatiques. En pareil cas, les antislashes et guillemets doivent être « échappés » par un antislash. Sont donc valides : `"toto tutu"`, `"a-b-c"` et `"Cor 3"`.

Voici trois éléments que vous pouvez placer à un niveau supérieur :

```
\layout {
  % pas en pleine largeur
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

Vous pouvez placer, n'importe où dans votre fichier, les instructions suivantes :

- `\version`
- `\include`

- `\sourcefilename`
- `\sourcefileline`
- Une ligne de commentaire, introduite par le signe `%`.
- Un bloc de commentaire, délimité par `%{ ... %}`.

Vous pouvez insérer des espaces dans votre fichier source afin de lui apporter une meilleure lisibilité. Les espaces superflus sont normalement ignorés. Notez cependant qu'il est des cas où l'espace est requis pour éviter tout risque d'erreur :

- Autour d'une accolade, qu'elle soit ouvrante ou fermante ;
- Après chaque commande ou variable, autrement dit tout élément qui commence par un \ ;
- Après tout élément qui sera interprété comme une expression Scheme, autrement dit tout élément qui commence par un `#` ;
- Pour séparer les éléments d'une expression Scheme ;
- En mode parole – `lyricmode` – avant et après les commandes `\override` et `\set`.

Voir aussi

Manuel d'initiation : Section "Organisation des fichiers LilyPond" dans *Manuel d'initiation*.

Manuel de notation : [Généralités en matière de titrages], page 515, Section 4.2.1 [Le bloc `\layout`], page 583.

3.2 Titres et entêtes

La plupart de la musique qui est éditée comporte un titre et le nom de son compositeur ; certains ouvrages dispensent beaucoup plus d'informations.

3.2.1 Création de titres et entête ou pied de page

Généralités en matière de titrages

Chaque bloc `\book` apparaissant dans un même fichier source résultera en un fichier indépendant, comme indiqué à la rubrique Section 3.1.5 [Structure de fichier], page 513. Chacun de ces fichiers résultants comporte trois endroits où placer des titrages : le **titrage de l'ouvrage** au début de chaque recueil (*book*), les **titrages de partie** au début de chaque partie (*bookpart*) et les **titrages de morceau** avant chaque pièce (*score*).

La valeur des champs de titrage `title` (le titre) et `composer` (le compositeur) se définissent dans des blocs `\header` – la syntaxe appropriée et la liste des différents champs disponibles par défaut sont à la section [Mise en forme par défaut des titrages subalternes], page 518. Les titrages d'un ouvrage, de ses parties ou des morceaux qu'il contient peuvent tous comporter les même champs bien que, par défaut, le titrage d'un morceau se limite à `piece` et `opus`.

Les blocs `\header` peuvent se placer à quatre endroits différents qui formeront une hiérarchie descendante :

- En tête du fichier source, avant même tout bloc `\book`, `\bookpart` ou `\score` ;
- Au sein d'un bloc `\book` et en dehors de tout bloc `\bookpart` ou `\score` qu'il contient ;
- Au sein d'un bloc `\bookpart` et en dehors de tout bloc `\score` qu'il contient ;
- Au sein d'un bloc `\score`.

La valeur des différents champs sera filtrée en respectant cette hiérarchie ; les valeurs persisteront à moins d'être écrasées par une autre valeur à un niveau inférieur. Ainsi :

- Le titre d'un ouvrage découle des champs définis en tête de fichier source, modifiés par les champs définis au sein du bloc `\book`. Les champs résultants serviront à affecter un titre de recueil à l'ouvrage, si tant est que quoi que ce soit génère une page au début de cet ouvrage, avant la première partie – un simple saut de page forcé (`\pageBreak`) suffit.

- Le titre d'une partie découle des champs définis en tête du fichier source, modifiés par les champs définis au sein du bloc `\book` puis par ceux définis au sein du bloc `\bookpart`. Les valeurs qui en résulteront permettront d'imprimer les titrages de partie pour cette partie.
- Le titre d'un morceau découle des champs définis en tête du fichier source, modifiés par les champs définis au sein du bloc `\book` puis par ceux définis au sein du bloc `\bookpart`, et enfin par ceux définis au sein du bloc `\score`. Les valeurs qui en résulteront permettront d'imprimer les titrages de morceau pour ce morceau. Notez toutefois que, pour un morceau, seuls les champs `piece` et `opus` seront imprimés, à moins d'avoir valorisé à `#t` la variable `print-all-headers` dans la section `\paper`.

Nul n'est besoin de fournir un bloc `\header` à chacun des quatre niveaux ; on peut se passer aussi bien de l'un d'eux que de tous. Dans la même veine, un fichier source simpliste peut ne pas mentionner de bloc `\book` ou `\bookpart` qui seront alors créés implicitement.

Lorsque l'ouvrage ne comporte qu'un seul morceau, le bloc `\header` devrait prendre place en tête de fichier, de telle sorte que soit produit un titrage de partie qui met à disposition tous les champs de titrage.

Lorsque l'ouvrage comporte plusieurs morceaux, différents arrangements du bloc `\header` permettent d'obtenir différents styles de publication musicale. Par exemple, si la publication comprend plusieurs pièces du même compositeur, un bloc `\header` placé en tête de fichier définira le titre de l'ouvrage et le compositeur, que l'on complètera par un bloc `\header` dans chaque bloc `\score` pour définir les champs `piece` et `opus`, comme ici :

```
\header {
  title = "SUITE I."
  composer = "J. S. Bach."
}
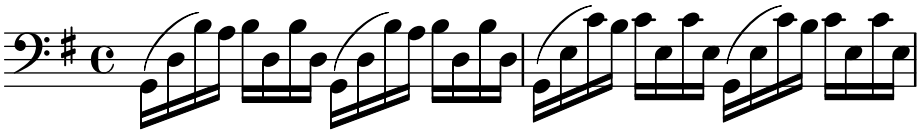
\score {
  \header {
    piece = "Prélude."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \repeat unfold 2 { g,16( d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16( e' c') b c e, c' e, } |
  }
}

\score {
  \header {
    piece = "Allemande."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b(c d e) fis( g a b) |
  }
}
```
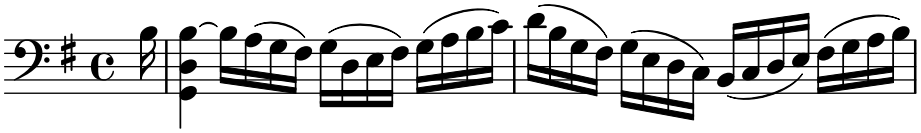
SUITE I.

J. S. Bach.

Prélude.



Allemande.



Des agencements plus élaborés sont aussi réalisables. Par exemple, les champs appartenant au titrage principal d'un ouvrage peuvent se reporter dans chaque bloc `\score`, certains étant modifiés voire supprimés manuellement :

```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Pas de mention spéciale par défaut pour cet ouvrage
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Pas de sous-titre pour ce morceau
      subtitle = ##f
    }
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
  \score {
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOCI"
      opus = "BWV 846"
      % Pas de sous-titre pour ce morceau
      subtitle = ##f
    }
    \new PianoStaff <<
```

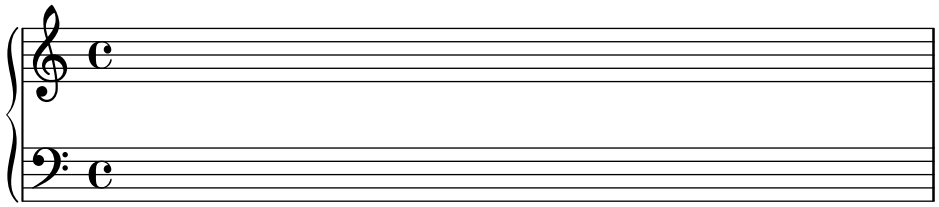
```
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
}
```

DAS WOHLTEMPERIRTE CLAVIER

TEIL I

PRAELUDIUM I

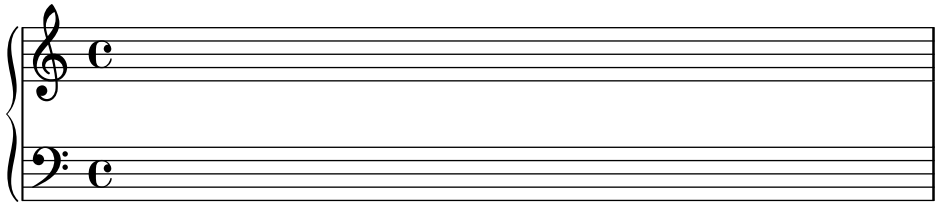
BWV 846



FUGA I

A 4 VOCI

BWV 846



Voir aussi

Manuel de notation : [Mise en forme par défaut des titrages subalternes], page 518, [Mise en forme personnalisée des titrages], page 523, Section 3.1.5 [Structure de fichier], page 513.

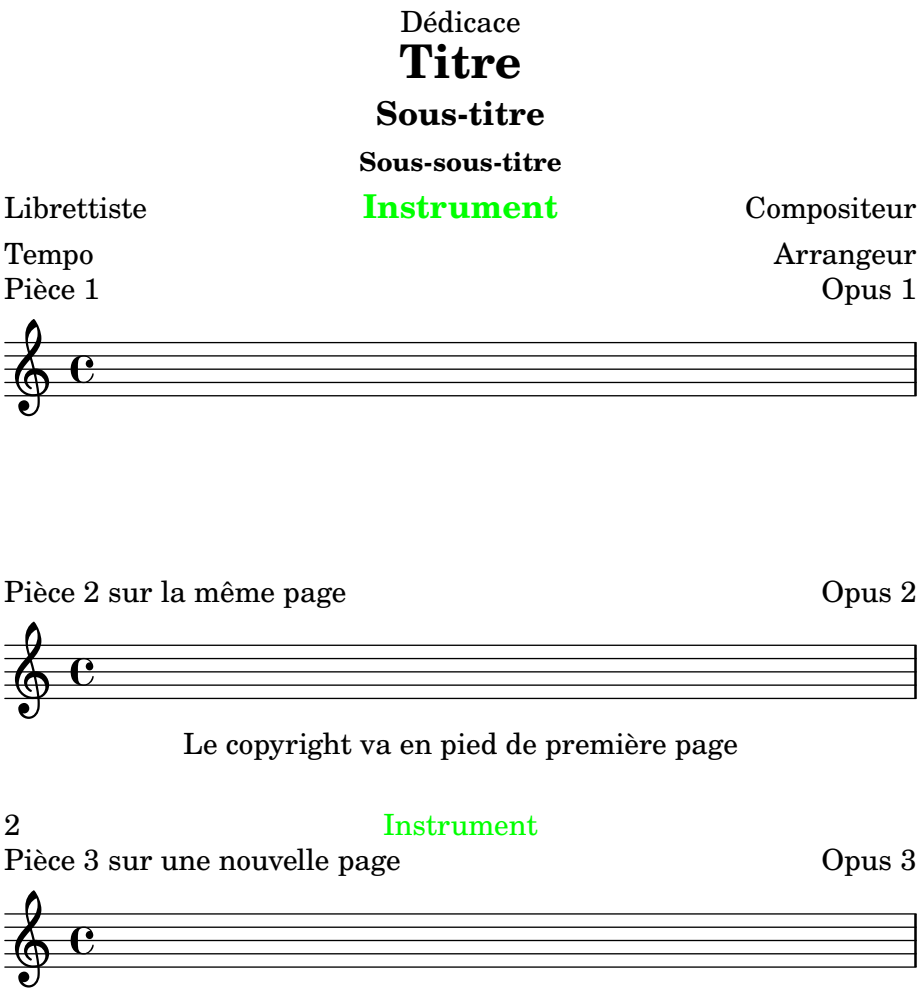
Mise en forme par défaut des titrages subalternes

Voici les différentes variables imprimables attachées au bloc `\header` :

```
\book {
  \header {
    % Les champs suivants sont centrés
    dedication = "Dédicace"
    title = "Titre"
    subtitle = "Sous-titre"
    subsubtitle = "Sous-sous-titre"
    % Les champs suivants sont répartis sur une même ligne, et
    % le champ "instrument" apparaîtra sur les pages suivantes
    instrument = \markup \with-color #green "Instrument"
    poet = "Librettiste"
    composer = "Compositeur"
```

```
    % Les champs suivants sont en opposition sur la même ligne
    meter = "Tempo"
    arranger = "Arrangeur"

    % Les champs suivants sont centrés en bas de page
    tagline = "Le « tagline » ou mention spéciale va en pied de dernière page"
    copyright = "Le copyright va en pied de première page"
  }
  \score {
    { s1 }
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 1"
      opus = "Opus 1"
    }
  }
  \score {
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 2 sur la même page"
      opus = "Opus 2"
    }
    { s1 }
  }
  \pageBreak
  \score {
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 3 sur une nouvelle page"
      opus = "Opus 3"
    }
    { s1 }
  }
}
```



Quelques précisions :

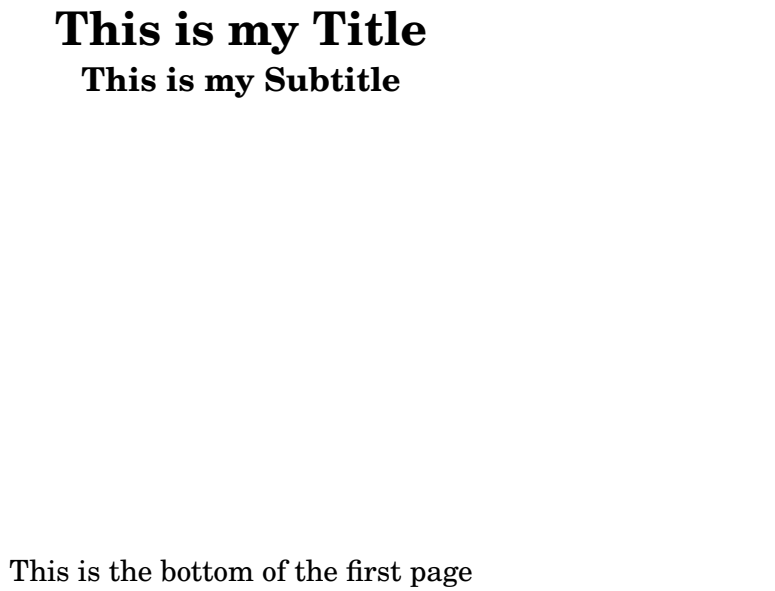
- Le nom de l'instrument sera répété en tête de chaque page.
- Seuls seront imprimés les champs `piece` et `opus` inclus dans un bloc `\score` dès lors que la variable `print-all-headers` reste désactivée (valeur à `##f`).
- Les champs d'un bloc `\header` qui n'auront pas été alimentés seront remplacés par un *markup* `\null` de façon à ne pas gaspiller d'espace.
- Par défaut, `scoreTitleMarkup` place les champs `piece` et `opus` de part et d'autre sur une même ligne.

Les possibilités de modifier la mise en forme par défaut sont abordées à la rubrique [Mise en forme personnalisée des titrages], page 523.
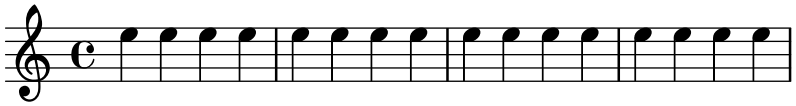
Un bloc `\book` qui commencerait directement par un bloc `\bookpart` ne verra pas ses titrages apparaître puisqu'il n'y a aucune page où imprimer le titre. Si toutefois le titre de l'ouvrage est requis, le bloc `\book` devra commencer par un *markup* ou une commande `\pageBreak`.

La variable `breakbefore` activée dans un bloc `\header` situé dans un bloc `\score` force le saut de page avant le morceau contenu dans ce `\score`. Vous pourrez ainsi séparer le titre principal de la musique.

```
\book {
  \header {
    title = "This is my Title"
    subtitle = "This is my Subtitle"
    copyright = "This is the bottom of the first page"
  }
  \score {
    \header {
      piece = "This is the Music"
      breakbefore = ##t
    }
    \repeat unfold 4 { e'' e'' e'' e'' }
  }
}
```



2
This is the Music



Music engraving by LilyPond 2.23.2—www.lilypond.org

Voir aussi

Manuel d'initiation : Section "Organisation des fichiers LilyPond" dans *Manuel d'initiation*.

Manuel de notation : [Mise en forme personnalisée des titrages], page 523, Section 3.1.5 [Structure de fichier], page 513.

Fichiers d'initialisation : `ly/titling-init.ly`.

Mise en forme par défaut des entête et pied de page

Les entête et pied – *header* et *footer* – sont des lignes de textes qui apparaissent en haut et en bas de chaque page, indépendamment du texte de l'ouvrage. Ils sont contrôlés par les variables suivantes, attachées au bloc `\paper` :

- `oddHeaderMarkup` – entête de page impaire
- `evenHeaderMarkup` – entête de page paire
- `oddFooterMarkup` – pied de page impaire
- `evenFooterMarkup` – pied de page paire
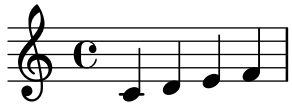
Ces variables *markup* n'accèdent qu'au contenu des champs du bloc `\header` principal, celui qui s'appliquera à tous les blocs `\score` du document. Ces variables sont définies dans le fichier `ly/titling-init.ly`, et de manière suivante par défaut :

- les numéros sont placés en haut à gauche (si pair) ou à droite (si impair) de chaque page à compter de la deuxième ;
- le contenu du champ `instrument` est centré en haut de chaque page à compter de la deuxième ;
- le texte du `copyright` est centré au bas de la première page ;
- le `tagline` – mention spéciale – se place au bas de la dernière page, ou bien sous le `copyright` s'il n'y a qu'une seule page.

Le texte de la mention spéciale par défaut se modifie en alimentant le champ `tagline` au niveau du bloc `\header` principal.

```
\book {
  \header {
    tagline = "... la notation musicale pour Tous"
  }
  \score {
    \relative {
      c'4 d e f
```

```
    }
  }
}
```



... la notation musicale pour Tous

Pour supprimer le `tagline` par défaut, il suffit de lui assigner la valeur `##f`.

3.2.2 Titrages personnalisés

Mise en forme personnalisée des champs de titrage

Toutes les commandes de mise en forme d'un `\markup` permettent de personnaliser le texte des entête, pied de page et éléments de titrage contenus dans un bloc `\header`.

```
\score {
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
  { s1 }
}
```

PRAELUDIUM I *BWV 846*



Voir aussi

Manuel de notation : Section 1.8.2 [Mise en forme du texte], page 263.

Mise en forme personnalisée des titrages

L'utilisation de commandes `\markup` au sein d'un bloc `\header` permet de modifier aisément l'apparence du texte, mais n'influence en rien le positionnement précis des éléments de titrage. L'accès au positionnement des champs de titrage est géré par les deux variables suivantes, attachées au bloc `\paper` :

- `bookTitleMarkup`
- `scoreTitleMarkup`

Le positionnement des titres, avec les valeurs par défaut de ces variables `\markup`, est illustré à la rubrique [Mise en forme par défaut des titrages subalternes], page 518.

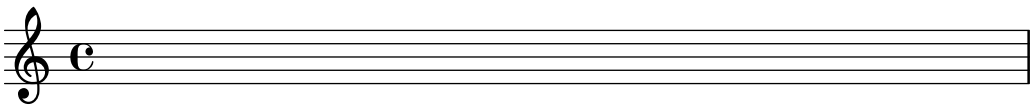
Voici les réglages par défaut de `scoreTitleMarkup`, tels que définis dans le fichier `ly/titling-init.ly` :

```
scoreTitleMarkup = \markup { \column {
  \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
}
```

Ceci aura donc pour effet de positionner les champs `piece` et `opus` sur la même ligne, en opposition :

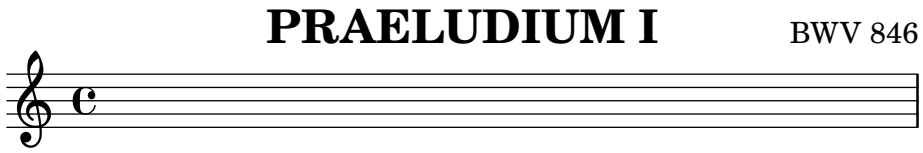
```
\score {
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
  { s1 }
}
```

PRAELUDIUM I BWV 846



Voici comment redéfinir le `scoreTitleMarkup` de telle sorte que le champ `piece`, dont nous modifions la taille et la graisse, se place au centre de cette ligne :
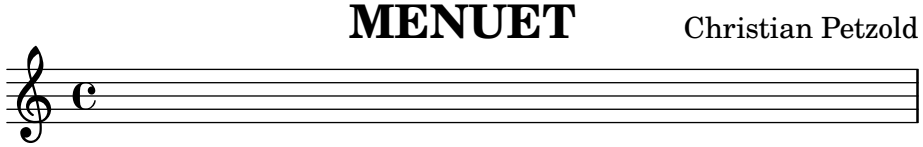
```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "PRAELUDIUM I"
      opus = "BWV 846"
    }
    { s1 }
  }
}
```
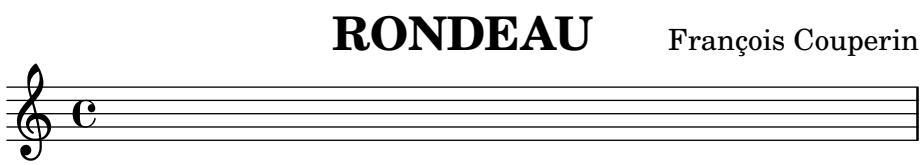


Les champs normalement absents du `\header` d'un bloc `\score` seront toutefois imprimés dès lors que vous aurez activé l'instruction `print-all-headers` au sein du bloc `\paper`. Le principal inconvénient de cette fonction réside dans le fait que les champs dévolus au titrage des parties devront être supprimés dans chacun des blocs `\score` de votre fichier source – voir [Généralités en matière de titrages], page 515.

Afin d'éviter ce désagrément, ajoutez le champ que vous désirez voir apparaître à la définition de `scoreTitleMarkup`. Nous allons, dans l'exemple suivant, ajouter au `scoreTitleMarkup` le champ `composer`, normalement associé au `bookTitleMarkup` ; chaque `\score` pourra alors mentionner un compositeur différent.
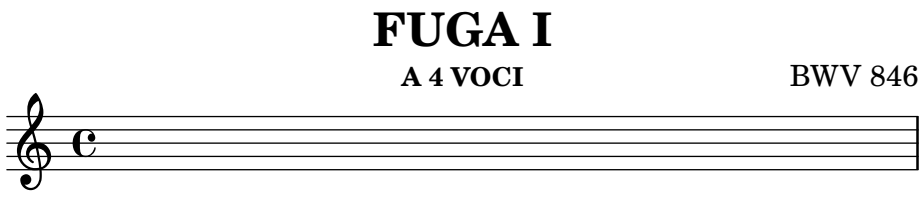
```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:composer
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "MENUET"
      composer = "Christian Petzold"
    }
    { s1 }
  }
  \score {
    \header {
      piece = "RONDEAU"
      composer = "François Couperin"
    }
    { s1 }
  }
}
```





Rien ne vous empêche de créer votre propre champ personnalisé, puis d'y faire référence dans la définition du *markup*.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP)
        \dir-column {
          \center-align \fontsize #-1 \bold
            \fromproperty #'header:mycustomtext %% User-defined field
          \center-align \fontsize #4 \bold
            \fromproperty #'header:piece
        }
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "FUGA I"
      mycustomtext = "A 4 VOCI" %% User-defined field
      opus = "BWV 846"
    }
    { s1 }
  }
}
```



Voir aussi

Manuel de notation : [Généralités en matière de titrages], page 515.

Mise en forme personnalisée des entête et pied de page

L'utilisation de commandes `\markup` au sein d'un bloc `\header` permet de modifier aisément l'apparence du texte, mais n'influence en rien le positionnement précis des entête et pied de page. L'accès au positionnement des champs concernés est géré par les quatre variables suivantes, attachées au bloc `\paper` :

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

L'instruction `\on-the-fly` au sein d'un `\markup` permet d'ajouter, lorsque certaines conditions sont respectées, des éléments au texte des entête et pied de page définis dans le bloc `\paper`. En voici la syntaxe :

```
variable = \markup {
  ...
  \on-the-fly \procédure markup
  ...
}
```

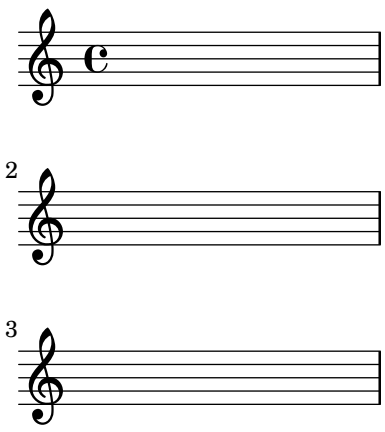
La *procédure* est appelée à chaque fois que la commande `\markup` où elle apparaît est évaluée. La *procédure* effectuera un test de conformité particulier et interprètera, autrement dit imprimera l'argument *markup* si et seulement si cette condition est remplie.

LilyPond dispose d'ores et déjà d'un certain nombre de procédures :

Nom de la procédure	**Condition testée**
print-page-number-check-first	il faut imprimer ce numéro de page.
create-page-number-stencil	print-page-numbers est vrai.
print-all-headers	print-all-headers est vrai.
first-page	c'est la première page du *book*.
not-first-page	ce n'est pas la première page du *book*.
(on-page nombre)	numéro de page = nombre
last-page	c'est la dernière page du *book*.
part-first-page	c'est la première page de la partie.
not-part-first-page	ce n'est pas la première page de la partie.
part-last-page	c'est la dernière page de la partie.
not-single-page	cette partie fait plus d'une page.

L'exemple suivant illustre la manière de centrer son numéro au bas de chaque page. Il nous faut tout d'abord annuler les définitions de `oddHeaderMarkup` et `evenHeaderMarkup` à l'aide d'un *markup* `\null`. Nous redéfinissons ensuite `oddFooterMarkup` pour qu'il contienne le numéro de page, centré. Enfin, nous appliquons le même paramétrage au `\oddFooterMarkup`.

```
\book {
  \paper {
    print-page-number = ##t
    print-first-page-number = ##t
    oddHeaderMarkup = \markup \null
    evenHeaderMarkup = \markup \null
    oddFooterMarkup = \markup {
      \fill-line {
        \on-the-fly \print-page-number-check-first
        \fromproperty #'page:page-number-string
      }
    }
    evenFooterMarkup = \oddFooterMarkup
  }
  \score {
    \new Staff { s1 \break s1 \break s1 }
  }
}
```



1

Plusieurs conditions `\on-the-fly` mentionnées l'une à la suite de l'autre se cumulent. Ainsi, par exemple,

```
\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }
```

teste si la sortie tient sur une page unique.

Voir aussi

Manuel de notation : [Généralités en matière de titrages], page 515, [Mise en forme par défaut des titrages subalternes], page 518.

fichiers d'initialisation : `../ly/titling-init.ly`.

3.2.3 Création des métadonnées des fichiers de sortie

En plus de s'imprimer sur la partition, les variables du bloc `\header` permettent de générer les métadonnées des fichiers de sortie. Dans le cas d'un fichier PDF, ces métadonnées pourront être affichées par le lecteur en tant que propriétés du document. Quel que soit le type de fichier de sortie, seules seront analysées les variables déterminées dans le `\header` du bloc définissant le fichier à générer, ainsi que celles des blocs hiérarchiquement supérieurs. Pour les fichiers PDF, seules les définitions du `\header` en dehors ou au niveau d'un bloc `\book` affecteront les métadonnées des documents PDF ; pour les fichiers MIDI seront utilisées les définitions jusqu'au niveau `\score`.

Par exemple, affecter « Symphony I » à la propriété `title` dans le bloc `\header` donnera aussi ce titre au document PDF et à la séquence MIDI.

```
\header {
  title = "Symphony I"
}
```

Lorsque le titre imprimé diffère de celui affiché en tant que propriété du PDF, devra être renseignée la propriété `pdftitle`.

```
\header {
  title = "Symphony I"
  pdftitle = "Symphony I by Beethoven"
}
```

Les variables `title`, `subject`, `keywords`, `subtitle`, `composer`, `arranger`, `poet`, `author` et `copyright` initialisent toutes les propriétés PDF, qu'il suffit de préfixer d'un « pdf » pour affecter aux propriétés PDF une valeur divergente de la sortie imprimable.

La propriété PDF `Creator` prend automatiquement la valeur « LilyPond » additionnée du numéro de version ; les valeurs de `CreationDate` et `ModDate` sont définies à la date et l'heure courantes – `ModDate` peut être écrasé par la variable de `\header` `moddate` (ou `pdfmoddate`) pour un horodatage PDF valide.

La variable `title` détermine aussi le nom de la séquence MIDI. L'utilisation de la variable `miditítle` permet d'attribuer à la séquence MIDI un nom différent de celui attribué au fichier imprimable.

3.2.4 Notes de bas de page

Les notes de bas de page sont utiles dans bien des situations. Dans tous les cas, un « appel de note » vient se placer en référence dans le texte ou la musique, et le « texte de la note » est reporté au bas de la page, isolé de la musique par un trait horizontal. L'apparence de ce séparateur peut se modifier à l'aide de la variable de papier `footnote-separator-markup` – voir [Variables supplémentaires d'entête et *markup*], page 582.

Selon qu'elle est référencée dans une expression musicale ou dans du texte indépendant, une note de bas de page sera créée suivant une procédure différente.

Notes de bas de page dans une expression musicale

Généralités sur l'annotation de musique

Il existe deux catégories d'annotation concernant une expression musicale :

Les annotations événementielles
: se rattachent à des événements particuliers, comme une note individuelle, un élément d'interprétation (doigté, accent ou nuance) ou des événements postérieurs (liaison, ligature manuelle). Une note de bas de page événementielle se libelle généralement sous la forme :

```
[position] \footnote [marque] décalage annotation musique
```

Les annotations temporelles
: se rapportent à un point particulier du déroulement d'un contexte musical. Certaines commandes, telles `\time` et `\clef`, ne reposent pas sur un événement pour la création de l'objet métrique ou clef. Il en va de même pour un accord : sa hampe ou ses crochets ne sont créés qu'à la fin d'un moment (plus exactement au travers de l'un des événements note qui le composent). Il n'est pas possible de connaître assurément lequel des événements note d'un accord est plus particulièrement à l'origine de la hampe ou du crochet. Il est donc plus aisé, pour de tels éléments, d'utiliser des annotations temporelles.

  Une annotation temporelle permet d'annoter des objets de rendu sans se référer à un événement. Elle se libelle généralement sous la forme :

```
\footnote [marque] décalage annotation [Contexte.]nom-grob
```

Les arguments, quelle que soit la catégorie d'annotation, peuvent se définir ainsi :

position
: Lorsque la commande `\footnote` s'applique à un élément d'interprétation ou un événement rattaché, et uniquement dans ces cas, elle doit être précédée d'un indicateur de positionnement (`-`, `_` ou `^`) de façon à rattacher *musique* (avec sa marque) à la note ou au silence qui précède.

marque
: Un *markup* ou une chaîne de caractères identifiant l'annotation tant au niveau de l'appel que de la note qui apparaîtra au bas de la page. L'absence de cet élément – ou une valeur de `\default` – incrémentera automatiquement le compteur. Ce compteur est réinitialisé à chaque page comportant une annotation.

décalage Une paire de nombres – '`#(2 . 1)`' par exemple – spécifiant le décalage de la marque, en abscisse et en ordonnée, par rapport au point de référence. Des valeurs positives décalent vers la droite ou le haut, des valeurs négatives vers la gauche ou le bas ; des valeurs à zéro centrent la marque sur le point de référence. Le décalage s'exprime en espace de portée.

Contexte Le contexte auquel appartient l'objet à annoter. Cet argument peut être omis dès lors qu'il s'agit d'un contexte de bas niveau tel que `Voice`.

nom-grob Le type d'objet à annoter – '`Flag`' par exemple. Lorsque cet élément est spécifié, c'est l'objet en question qui servira de point de référence, même s'il trouve son origine non pas directement dans une expression musicale mais dans tout objet du type spécifié intervenant à cet instant précis de la partition.

annotation
un *markup* ou une chaîne de caractères qui sera reporté au bas de la page.
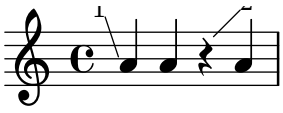
musique l'élément qui fait l'objet du commentaire, qu'il s'agisse d'un événement musical, de l'un des constituants d'un accord ou d'un événement rattaché.

Notes de bas de page événementielles

Ce type de note de bas de page s'attache à un objet de rendu généré directement par l'événement correspondant à *musique*. Il répond à la syntaxe :

```
\footnote [marque] décalage annotation musique
```

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "Une note" a4
    a4
    \footnote #'(2 . 2) "Un silence" r4
    a4
  }
}
```

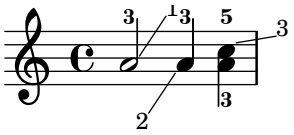


[1]Une note
[2]Un silence

Un accord *dans son intégralité* ne peut pas faire l'objet d'une note de bas de page événementielle : un accord, même s'il ne contient qu'une seule et unique note, ne génère aucun événement en propre. Une des notes *au sein* de l'accord peut toutefois se voir attribuer une annotation :

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(2 . 3) "Résultat non probant" <a-3>2
```

```
    <\footnote #'(-2 . -3) "Résultat probant" a-3>4
    <a-3 \footnote #'(3 . 1/2) "Tout aussi probant" c-5>4
  }
}
```

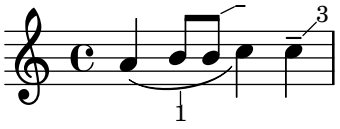


[1]Résultat non probant
[2]Résultat probant
[3]Tout aussi probant

Lorsque l'annotation concerne un événement postérieur ou une articulation, la commande `\footnote` **doit** être précédée d'un indicateur de position (`-`, `_` ou `^`) et suivie de l'événement postérieur ou l'articulation comme argument *musique*. Dans ce cas, la commande `\footnote` peut se considérer comme une copie de son dernier argument auquel on attache une annotation. La syntaxe consacrée est :

position `\footnote [`*marque*`]` *décalage annotation musique*

```
\book {
  \header { tagline = ##f }
  \relative {
    a'4_\footnote #'(0 . -1) "Une liaison arbitrairement en dessous" (
    b8^\footnote #'(1 . 0.5) "Une ligature manuelle forcée en haut" [
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
}
```



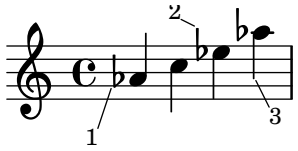
[1]Une liaison arbitrairement en dessous
[2]Une ligature manuelle forcée en haut
[3]Tenuto

Notes de bas de page temporelles

Lorsque la note de bas de page se réfère à un objet de rendu résultant d'un événement – `Accidental` ou `Stem` découlent d'un `NoteHead` –, l'argument *nom-grob* de l'objet en question est requis après le texte de l'annotation, en lieu et place de *musique* :

```
\book {
  \header { tagline = ##f }
```
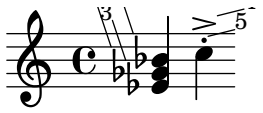
```
  \relative c'' {
    \footnote #'(-1 . -3) "Un bémol" Accidental
    aes4 c
    \footnote #'(-1 . 0.5) "Un autre bémol" Accidental
    ees
    \footnote #'(1 . -2) "Une hampe" Stem
    aes
  }
}
```



[1]Un bémol
[2]Un autre bémol
[3]Une hampe

Notez bien que, lorsque *nom-grob* est spécifié, tous les objets de ce type qui se trouvent à ce même instant se verront attacher une annotation :

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "Un bémol" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Une articulation" Script
    c'->-.
  }
}
```
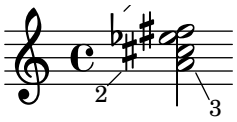


[1]Un bémol
[2]Un bémol
[3]Un bémol
[4]Une articulation
[5]Une articulation

Une note incluse dans un accord peut individuellement se voir attribuer une annotation événementielle. Une tête de note (`NoteHead`) est le *seul* objet directement généré par un constituant d'accord ; elle peut donc être affectée d'une annotation événementielle. Tous les autres objets constituant un accord sont générés indirectement. La commande `\footnote` ne dispose pas d'une syntaxe permettant de spécifier *à la fois* un type d'objet *et* un événement particulier

auquel s'attacher. De tels objets pourront toutefois faire l'objet d'une annotation temporelle, préfixée d'un `\single` afin d'annoter l'événement directement consécutif :

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    < \footnote #'(1 . -2) "Un la" a
      \single \footnote #'(-1 . -1) "Un dièse" Accidental
      cis
      \single \footnote #'(0.5 . 0.5) "Un bémol" Accidental
      ees fis
    >2
  }
}
```



[1]Un bémol
[2]Un dièse
[3]Un la

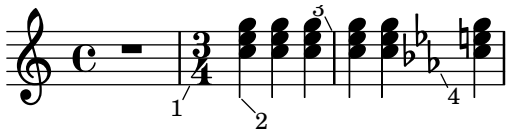
Note : Lorsque plusieurs notes de bas de page se rapportent à un même empilement vertical comme ci-dessus, elles sont numérotées et apparaîtront selon l'ordre vertical des éléments présentés, autrement dit celui positionné le plus haut en premier, non dans leur ordre d'apparition dans le fichier source.

Les objets de rendu tels que changement de clef ou d'armure tirent leur origine dans la modification d'une propriété plutôt que d'un véritable événement. D'autres, comme les barres ou numéros de mesure, dépendent directement de la temporisation. C'est la raison pour laquelle de tels objets doivent s'annoter en fonction de leur survenance au fil de la musique. Les notes de bas de page temporelles sont la solution à privilégier lorsqu'il s'agit d'annoter les hampes ou ligatures affectant des accords : bien qu'une telle fonctionnalité puisse s'appliquer à l'un des événements constituant l'accord, rien ne laisse présager lequel serait le plus approprié.

En matière de note de bas de page temporelle, l'objet de rendu considéré doit toujours être mentionné explicitement, ainsi que le contexte si l'objet est créé dans un autre contexte que celui du plus bas niveau.

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Changement de métrique" Staff.TimeSignature
    \time 3/4
    \footnote #'(1 . -1) "Hampe de l'accord" Stem
    <c e g>4 q q
    \footnote #'(-0.5 . 1) "Barre de mesure" Staff.BarLine
```
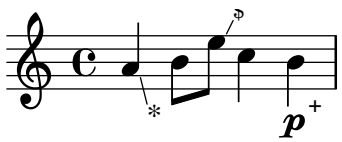
```
    q q
    \footnote #'(0.5 . -1) "Changement d'armure" Staff.KeySignature
    \key c \minor
    q
  }
}
```



[1]Changement de métrique
[2]Hampe de l'accord
[3]Barre de mesure
[4]Changement d'armure

Les appels de note peuvent être personnalisés, et le trait reliant l'objet à l'appel supprimé :

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "* La première note" }
    a'4 b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
      \markup { \super "$" \italic " La deuxième note" }
    e c4
    \once \override Score.FootnoteItem #'annotation-line = ##f
    b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
      \markup { \super "+" \italic " Éditorial" } \p
  }
}
```



* *La première note*
$ *La deuxième note*
+ *Éditorial*

D'autres exemples de personnalisation des appels de note sont donnés à la rubrique [Notes de bas de page dans du texte indépendant], page 534.

Notes de bas de page dans du texte indépendant

De telles notes de bas de page affectent les *markup* extérieurs aux expressions musicales. Il n'est pas nécessaire en pareil cas d'indiquer un point de référence par un trait ; l'appel de note vient juste s'accoler au *markup* qui fait l'objet de l'annotation. Les appels de note peuvent être gérés

automatiquement, auquel cas ils seront numériques, ou bien manuellement en fournissant un indicateur particulier.

Les notes de bas de page concernant du texte indépendant se gèrent différemment selon qu'elles sont automatiques ou manuelles.

Notes de bas de page automatiques dans du texte

La syntaxe consacrée dans le cas d'une gestion automatique des appels de note est :

```
\markup { ... \footnote texte annotation ... }
```

Ses les éléments sont :
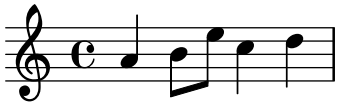
texte le *markup* ou la chaîne de caractères sur lequel porte l'annotation ;

annotation
un *markup* ou une chaîne de caractères constituant le texte de l'annotation qui sera reportée en bas de page.

Par exemple :

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple"
    \footnote "tune" \italic " By me"
    "is shown below.  It is a"
    \footnote "recent" \italic " Aug 2012"
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune is shown below. It is a recent composition.



By me
Aug 2012

Notes de bas de page personnalisées dans du texte

La syntaxe consacrée dans le cas d'une gestion personnalisée des appels de note est :

```
\markup { ... \footnote appel annotation ... }
```
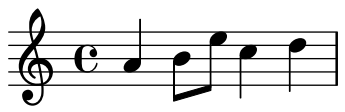
Ses les éléments sont :

appel un *markup* ou une chaîne de caractères représentant l'appel de note affecté à ce point de référence. Notez bien que cette marque ne sera **pas** reproduite automatiquement avant le texte proprement dit de l'annotation.

annotation
un *markup* ou une chaîne de caractères constituant le texte de l'annotation qui sera reportée en bas de page, précédé de l'*appel*.

N'importe quel caractère simple tel que `*` ou `+` peut s'utiliser en tant qu'appel de note, comme nous l'avons vu à la rubrique [Notes de bas de page dans une expression musicale], page 529. D'autres caractères particuliers sont accessibles sous forme de raccourci – voir la rubrique [Équivalents ASCII], page 550 :

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote "*" \italic "* By me"
    "is shown below.  It is a recent"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Aug 2012"
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```
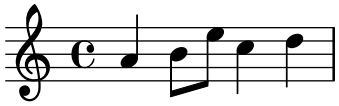


* *By me*
† *Aug 2012*

Un appel de note peut aussi se libeller sous la forme d'un point de code unicode – voir la rubrique [Unicode], page 550 :

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple tune"
```

```
    \footnote \super \char##x00a7 \concat {
      \super \char##x00a7 \italic " By me"
    }
    "is shown below.  It is a recent"
    \footnote \super \char##x00b6 \concat {
      \super \char##x00b6 \italic " Aug 2012"
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune [§] is shown below. It is a recent [¶] composition.



[§] *By me*
[¶] *Aug 2012*

Voir aussi

Manuel d'initiation : Section "Objets et interfaces" dans *Manuel d'initiation*.

Manuel de notation : [Commentaires textuels], page 256, [Équivalents ASCII], page 550, [Indications textuelles], page 259, [Info-bulle], page 248, Section A.13 [Liste des caractères spéciaux], page 803, [Unicode], page 550.

Référence des propriétés internes : Section "FootnoteEvent" dans *Référence des propriétés internes*, Section "FootnoteItem" dans *Référence des propriétés internes*, Section "FootnoteSpanner" dans *Référence des propriétés internes*, Section "Footnote_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les notes de bas de page ne peuvent que s'empiler l'une au-dessus de l'autre ; elles ne seront jamais présentées à la queue leu leu.

Silences multimesures, ligatures automatiques et paroles ne peuvent se voir affecter de note de bas de page.

Les notes de bas de page peuvent générer des chevauchements quand elles sont trop nombreuses sur une même page.

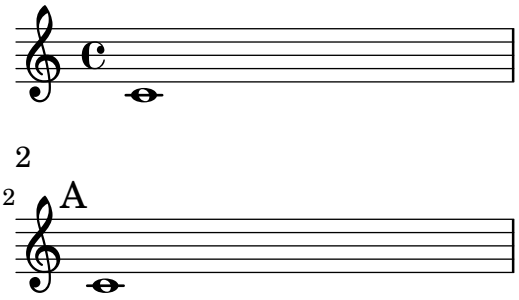
3.2.5 Référencement des numéros de page

LilyPond vous permet, à l'aide de la commande `\label`, d'insérer des points de référence dans un ouvrage, aussi bien en dehors qu'au fil de la musique. Ce point de référence pourra être

ensuite repris à l'intérieur d'un *markup* ; vous pourrez même y ajouter le numéro de page grâce à la commande de *markup* `\page-ref`.

```
\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    {
      c'1
      \pageBreak \mark A \label #'markA
      c'1
    }
  }

  \markup { Le premier mouvement débute à la page \page-ref #'firstScore "0" "?" }
  \markup { Le repère A est à la page \page-ref #'markA "0" "?" }
}
```



Le premier mouvement débute à la page 1
Le repère A est à la page 2

L'instruction `\page-ref` prend trois arguments :

1. le point de référence, sous la forme d'un symbole Scheme, comme par exemple `#'firstScore`,
2. un « emporte-pièce » afin d'estimer la longueur totale du *markup*, et
3. un texte de remplacement au cas où la référence ne serait pas retrouvée.

La présence de l'emporte-pièce est rendue nécessaire par le fait que les *markups* sont générés avant que les sauts de page ne soient positionnés. Bien que le numéro de page en question ne soit pas encore déterminé, LilyPond doit connaître les dimensions de ce *markup*. Vous pouvez, lorsque l'ouvrage contiendra plus de dix pages, stipuler un emporte-pièce sur deux caractères – soit `"00"`.

Commandes prédéfinies

`\label`, `\page-ref`.

3.2.6 Table des matières

La commande `\markuplist \table-of-contents` vous permettra de générer une table des matières. Les éléments qui la composeront sont créés par la commande `\tocItem`, insérée indépendamment ou au sein d'une expression musicale.

`\markuplist \table-of-contents`

```
\pageBreak

\tocItem \markup "Première partie"
\score {
  {
    c'4  % ...
    \tocItem \markup "Passage spécifique de la première partie"
    d'4  % ...
  }
}

\tocItem \markup "Seconde partie"
\score {
  {
    e'4 % ...
    \tocItem actI \markup "Acte I"
    f'4 % ...
    \tocItem actI.sceneI \markup "Scène 1"
    g'4 % ...
    \tocItem actI.sceneI.recitativo \markup "Récit."
    a'4 % ...
  }
}
```

Un libellé peut, facultativement, être associé à un élément particulier ou de façon hiérarchique dans une liste de libellés existants, terminant alors par le libellé de cet élément. Ce dernier cas permet de marquer l'élément comme « enfant » d'éléments précédemment libellés et ainsi laisser transparaître la structure de la partition dans la table des matières.

Les *markups* dévolus à la mise en forme de la table des matières se définissent dans le bloc `\paper`. LilyPond dispose de trois *markups* prédéfinis :

- `tocTitleMarkup`

  Utilisé pour mettre en forme le titre de la table des matières.

  ```
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table of Contents" \null }
    \null
  }
  ```

- `tocItemMarkup`

  Utilisé pour mettre en forme les éléments au sein de la table des matières.

  ```
  tocItemMarkup = \markup \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
  ```

- `tocFormatMarkup`

  Utilisé pour mettre en forme les entrées de premier niveau dans la table des matières si tant est qu'existent plusieurs niveaux hiérarchiques. Il s'agit ici d'une procédure, comme abordé dans Section "Construction d'un markup en Scheme" dans *Extension de LilyPond*.

  ```
  tocFormatMarkup = #make-bold-markup
  ```

- `tocIndentMarkup`

  Utilisé pour définir comment sera mise en évidence la hiérarchie. Ce *markup* sera imprimé zéro, une ou plusieurs fois selon le niveau de chacune des entrées.

  ```
  tocIndentMarkup = \markup \hspace #4
  ```

Toutes ces variables sont adaptables.

Voici comment, par exemple, franciser le titre :

```
\paper {
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
```

L'exemple suivant illustre la manière de modifier la taille des éléments de la table des matières :

```
tocItemMarkup = \markup \large \fill-line {
  \fromproperty #'toc:text \fromproperty #'toc:page
}
```

Notez bien la manière de référencer le libellé et le numéro de page dans la définition de `tocItemMarkup`.

Grâce à la commande `\tocItemWithDotsMarkup`, l'élément et son numéro de page peuvent se rejoindre par une ligne pointillée :

```
\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}
```

Table of Contents

Allegro 1
Largo 1

Au-delà de ces mécanismes de mise en forme, il est posssible de définir d'autres commandes et *markups* afin de construire une table plus élaborée. Dans l'exemple qui suit, nous créons un nouveau style d'élément dans le but de mentionner les actes et scènes dans la table des matières d'un opéra :

Commençons par définir une nouvelle variable de type `markup` – appelée `tocActMarkup` – au sein du bloc `\paper`.

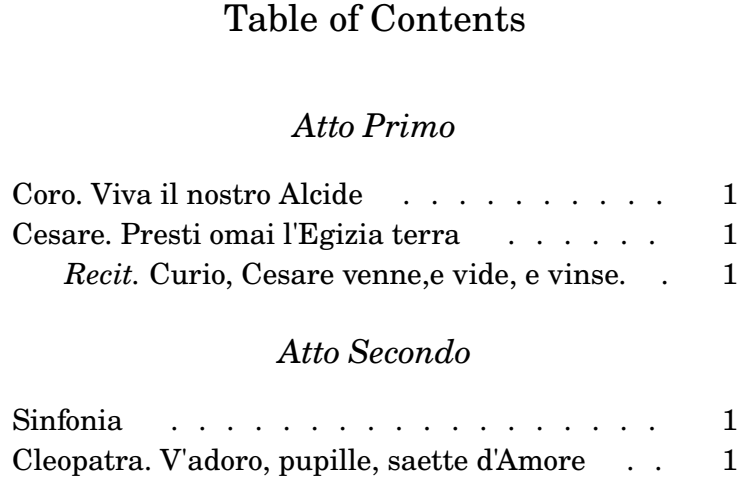
```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}
```

Créons ensuite une fonction musicale (`tocAct`) utilisant la nouvelle définition de *markup* `tocActMarkup`, tout en lui autorisant de définir un libellé pour chaque acte.

```
tocAct =
```

```
  #(define-music-function (label text) (symbol? markup?)
+     (add-toc-item! 'tocActMarkup label text))
```

Dans un fichier LilyPond, l'utilisation de cette définition personnalisée, avec quelques adaptations aux réglages par défaut, pourrait ressembler à ceci :



Cet exemple illustre par ailleurs l'utilisation de la commande `\fill-with-pattern` dans le cadre d'une table des matières.

Commandes prédéfinies

`\table-of-contents`, `\tocItem`, `tocItemMarkup`, `tocTitleMarkup`, `tocFormatMarkup`, `tocIndentMarkup`.

Voir aussi

Fichiers d'initialisation : `../ly/toc-init.ly`.

3.3 Travail sur des fichiers texte

3.3.1 Insertion de fichiers LilyPond

Lorsqu'un projet prend de l'importance en volume, il est judicieux de le scinder en plusieurs fichiers, auxquels vous ferez référence avec un simple

```
\include "autrefichier.ly"
```

Une ligne `\include "autrefichier.ly"` dans un fichier revient à recopier intégralement le contenu de `autrefichier.ly` à l'endroit même ou est placée l'instruction `\include`. Vous pouvez par exemple écrire un fichier individuel par instrument, puis les regrouper pour former le fichier « conducteur ». Les différentes variables définies dans les fichiers séparés seront normalement reprises et utilisables dans le fichier formant le conducteur. Les sections balisées dans les fichiers individuels peuvent être réutilisées en différents endroit de la partition, comme expliqué à la rubrique Section 3.3.2 [Différentes éditions à partir d'une même source], page 543.

Lorsque le fichier auquel il est fait référence se trouve dans le même répertoire, donner seulement son nom en argument à la commande `\include` suffit. S'il se trouve ailleurs, vous devrez indiquer le chemin d'accès, absolu ou relatif, en respectant toutefois la syntaxe UNIX – autrement dit, le séparateur de répertoire est une oblique normale `/` et non l'oblique inverse `\` de DOS ou Windows. Par exemple, si le fichier `truc.ly` se trouve dans le répertoire supérieur au répertoire de travail, la ligne devra être

```
\include "../truc.ly"
```

ou bien, si les fichiers correspondant aux parties d'orchestre se trouvent dans le sous-répertoire `parties` relativement au répertoire courant, vous devrez mentionner

```
\include "parties/VI.ly"
\include "parties/VII.ly"
etc.
```

Les fichiers à inclure peuvent eux-mêmes contenir des instructions `\include`. Ces instructions `\include` de second niveau seront, par défaut, interprétées par rapport à leur situation dans l'arborescence. Tel sera, par exemple, le cas d'une biblothèque générale libA créée pour utiliser des sous-fichiers à l'aide d'inclusions dans un fichier catalogue, comme ici :

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

puis le fichier catalogue, `libA.ly`, qui contient

```
\include "A1.ly"
\include "A2.ly"
...
```

Tout fichier `.ly` peut désormais consulter l'intégralité de cette bibliothèque grâce à un simple

```
\include "~/libA/libA.ly"
```

Vous pouvez toutefois influer sur ce comportement de manière globale à l'aide de l'option `-drelative-includes=#f` en ligne de commande ou en ajoutant une clause `#(ly:set-option 'relative-includes #f)` en tête du fichier principal. En pareil cas, le chemin à suivre pour chacune des commandes `\include` sera pris relativement au fichier principal. Selon l'endroit où `relative-includes` est valorisé à `#t` ou `#f`, la commande `\include` permettra d'incorporer des fichiers contenus dans l'arborescence du répertoire principal et des fichiers situés ailleurs.

Vous pouvez inclure des fichiers dont vous spécifierez le chemin d'accès sur la ligne de commande au moment de lancer la compilation. L'appel à ces fichiers ne mentionnera alors que leur nom. Par exemple, si vous voulez compiler avec cette méthode le fichier `principal.ly` qui inclut des fichiers situés dans le sous-répertoire `parties`, placez vous dans le répertoire contenant `principal.ly`, puis tapez

```
lilypond --include=parties principal.ly
```

tout en ayant bien dans `principal.ly`

```
\include "VI.ly"
\include "VII.ly"
 etc.
```

Lorsqu'un fichier est voué à être inclus dans nombre de partitions, vous pouvez le placer dans le répertoire de LilyPond `../ly`. Attention : ce répertoire varie selon votre installation, comme indiqué au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*. Ce fichier sera inclus dès lors que vous fournirez uniquement son nom en argument à la fonction `\include`. C'est par exemple le cas du fichier de définition particulier `gregorian.ly`.

Au moment où vous lancez LilyPond, un certain nombre de fichiers se retrouvent inclus par défaut ; il suffit d'activer le mode verbeux en faisant `lilypond --verbose` pour s'en rendre compte. Vous verrez ainsi défiler, en plus de nombreuses informations, le nom d'un certain nombre de fichiers et de chemins d'accès. Les fichiers les plus important sont mentionnés au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*. Si vous venez à les modifier, rappelez-vous qu'ils seront écrasés à l'installation d'une nouvelle version de LilyPond.

Vous trouverez quelques exemples simples d'utilisation de la commande `\include` au chapitre Section "Conducteurs et parties" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "Conducteurs et parties" dans *Manuel d'initiation*.

Problèmes connus et avertissements

Lorsque vous incluez un fichier qui porte le même nom que l'un des fichiers d'initialisation de LilyPond, le fichier de la distribution de LilyPond aura préséance.

3.3.2 Différentes éditions à partir d'une même source

Plusieurs méthodes permettent de générer différentes versions d'une partition à partir d'une même source. Les variables – ou identificateurs – sont sûrement le moyen le plus simple de combiner de différentes manières des passages relativement longs, alors que les balises permettront de sélectionner de courts fragments selon leur utilisation.
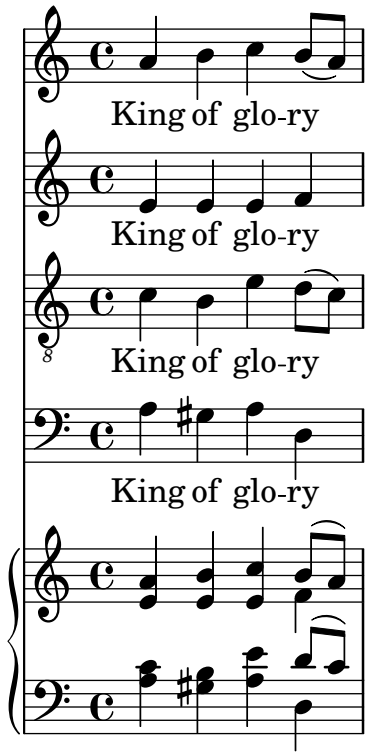
Quelle que soit la méthode utilisée, séparer la notation de la structure de la partition vous donnera plus de liberté dans l'agencement de l'ouvrage final, puisque vous ne reviendrez pas sur la musique qui le compose.

Utilisation de variables

Un fragment musical identifié par une variable est réutilisable à divers endroits de la partition, comme nous l'avons vu à la rubrique Section "Organisation du code source avec des variables" dans *Manuel d'initiation*. Par exemple, une partition pour chœur *a cappella* comporte souvent une réduction pour piano reprenant toutes les voix ; il s'agit de la même musique, et vous ne devrez donc la saisir qu'une seule fois. D'autre part, la musique issue de deux variables peut se combiner sur une seule portée, comme nous l'avons vu à la rubrique [Regroupement automatique de parties], page 189. Prenons l'exemple suivant :

```
sopranoMusic = \relative { a'4 b c b8( a) }
altoMusic = \relative { e'4 e e f }
tenorMusic = \relative { c'4 b e d8( c) }
bassMusic = \relative { a4 gis a d, }
allLyrics = \lyricmode { King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenor" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Bass" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "RH" {
      \set Staff.printPartCombineTexts = ##f
```

```
        \partCombine \sopranoMusic \altoMusic
      }
      \new Staff = "LH" {
        \set Staff.printPartCombineTexts = ##f
        \clef "bass"
        \partCombine \tenorMusic \bassMusic
      }
    >>
  >>
```



Générer une partition chorale ou la réduction pour piano ne requiert que de modifier la structure des éléments, sans aucunement toucher à la musique.

Dans le cas d'une partition relativement longue, vous pouvez isoler la définition des différentes variables dans des fichiers séparés que vous rappellerez ensuite, comme indiqué à la rubrique Section 3.3.1 [Insertion de fichiers LilyPond], page 541.

Utilisation de balises

La commande `\tag #'partieA` affecte à une expression musicale le nom *partieA*. Les expressions ainsi balisées pourront être filtrées par la suite, à l'aide de `\keepWithTag #'nom` ou `\removeWithTag #'nom`. Ces filtres fonctionnent de la manière suivante :

Filtre	**Résultat**

Musique balisée précédée de
`\keepWithTag #'nom`
ou
`\keepWithTag #'(nom1 nom2...)`
	Musique non balisée et musique balisée par l'un des noms de balise fournis seront incluses ; la musique balisée autrement est exclue.

Musique balisée précédée de `\removeWithTag #'`*nom* ou `\removeWithTag #'(`*nom1 nom2...*`)`	Musique non balisée et fragments appelés autrement que par l'un des noms fournis seront inclus ; la musique balisée par l'un des noms mentionnés est exclue.

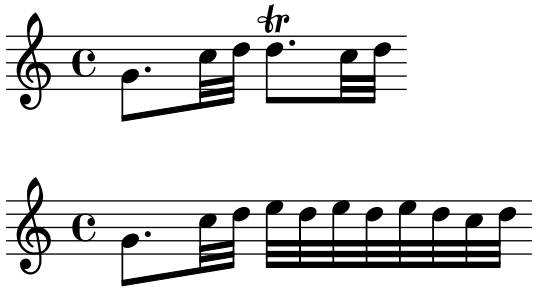
Musique balisée non précédée de
`\keepWithTag`
ou
`\removeWithTag`
	Musique balisée et non balisée seront incluses.

L'argument des commandes `\tag`, `\keepWithTag` et `\removeWithTag` doit être un symbole ou une liste de symboles (tel que `#'conducteur` ou `#'(violonI violonII)`), suivi d'une expression musicale. Si, *et seulement si* les symboles sont des indentifiants LilyPond valides (caractères alphabétiques uniquement, sans chiffre, souligné ou tiret) qui ne peuvent se confondre avec des notes, le `#'` peut s'omettre et, pour raccourcir, une liste de symbole peut utiliser le point en séparateur – autrement dit, `\tag #'(violinI violinII)` peut s'écrire `\tag violinI.violinII`. Ceci s'applique aussi bien pour `\keepWithTag` que pour `\removeWithTag`. Les commandes de balisage sont des fonctions musicales ; elles ne peuvent donc s'utiliser pour filtrer des éléments qui ne sont pas des expressions musicales tels que des blocs `\book` ou `\score`.

Dans l'exemple qui suit, nous obtenons deux versions du même extrait, l'une pour le conducteur, l'autre pour l'instrumentiste qui, elle, comportera les ornements développés.

```
music = \relative {
  g'8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }

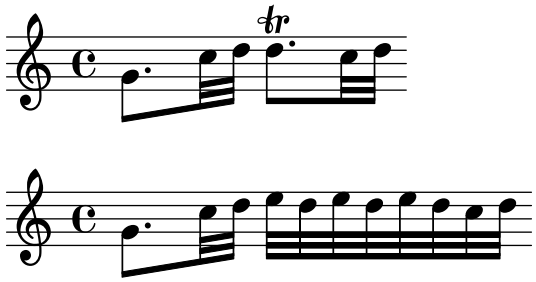
\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}
```



Il est parfois plus aisé d'exclure des fragments :

```
music = \relative {
  g'8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
 }
```

```
\score {
  \removeWithTag #'expand
  \music
}
\score {
  \removeWithTag #'trills
  \music
}
```



Ce principe de filtrage peut s'appliquer aux articulations, textes, etc. Il suffit de positionner

```
-\tag #ma-balise
```

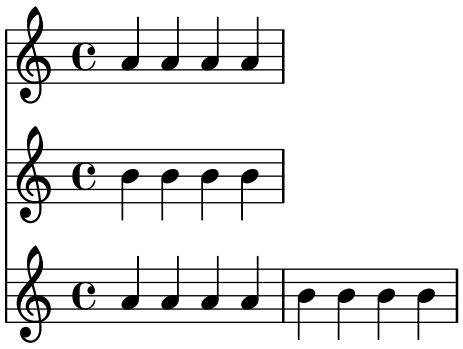
avant l'articulation ou le texte, comme ici :

```
c1-\tag #'doigt ^4
c1-\tag #'gaffe ^"Attention !"
```

Ceci définira une note avec une indication conditionnelle de doigté ou un texte.

Vous pouvez baliser différemment la même expression musicale en saisissant plusieurs `\tag` ou bien en combinant plusieurs balises dans une liste :
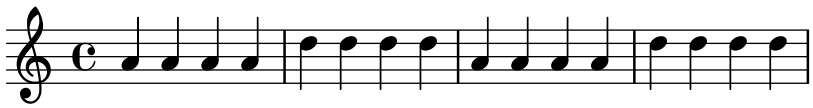
```
music = \relative c'' {
  \tag #'a \tag #'both { a4 a a a }
  \tag #'(b both) { b4 b b b }
}
<<
\keepWithTag #'a \music
\keepWithTag #'b \music
\keepWithTag #'both \music
>>
```



L'application concomitante de plusieurs filtres `\removeWithTag` à la même expression musicale permet d'exclure plusieurs balisages. Une liste fournie en argument à un unique `\removeWithTag` produira le même effet :

```
music = \relative c'' {
```
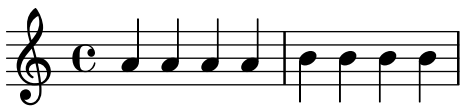
```
  \tag #'A { a4 a a a }
  \tag #'B { b4 b b b }
  \tag #'C { c4 c c c }
  \tag #'D { d4 d d d }
}
\new Voice {
  \removeWithTag #'B
  \removeWithTag #'C
  \music
  \removeWithTag #'(B C)
  \music
}
```



L'application de plus d'un filtre `\keepWithTag` à la même expression musicale aboutit à l'exclusion de **tous** les balisages. En effet, si le premier filtre exclut tous les autres balisages, l'application du second exclura les effets du premier. L'utilisation d'une unique commande `\keepWithTag` avec une liste de balises est en pareil cas des plus pertinente : seront exclus tous les fragments non concernés par l'une quelconque des balises mentionnées.

```
music = \relative c'' {
  \tag #'violinI { a4 a a a }
  \tag #'violinII { b4 b b b }
  \tag #'viola { c4 c c c }
  \tag #'cello { d4 d d d }
}

\new Staff {
  \keepWithTag #'(violinI violinII)
  \music
}
```



imprimera les `\tag`s *violinI* et *violinII*, mais ni *viola* ni *cello*.

Bien que `\keepWithTag` soit efficace pour gérer *un* jeu d'alternatives, le rejet de musique filtrée par des balises *étrangères* se révèle problématique lorsque les `\tag` sont utilisés à plusieurs fins. Des « groupements de balises » peuvent alors être déclarés :

```
\tagGroup #'(violinI violinII viola cello)
```

Les différents filtres appartiennent désormais tous à un seul regroupement. Notez bien qu'une balise ne saurait être membre de plusieurs regroupements.
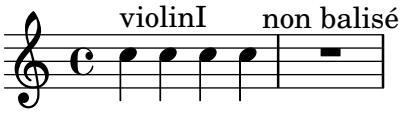
```
\keepWithTag #'violinI ...
```

ne prendra désormais en compte que la musique concernée par la balise `violinI` du groupe de filtres : tout élément de la musique qui serait balisé par l'un des autres filtres de ce jeu sera rejeté.

```
music = \relative {
  \tagGroup #'(violinI violinII viola cello)
```

```
  \tag #'violinI { c''4^"violinI" c c c }
  \tag #'violinII { a2 a }
  \tag #'viola { e8 e e2. }
  \tag #'cello { d'2 d4 d }
  R1^"non balisé"
}

\new Voice {
  \keepWithTag #'violinI
  \music
}
```
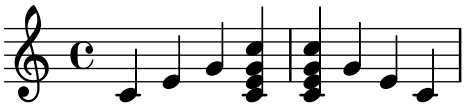


Dans le cadre de la commande `\keepWithTag`, seules les balises du regroupement mentionnées dans la commande seront visibles.

Il peut arriver que vous ayez besoin de raccorder quelque chose en un point particulier d'une expression musicale. Les commandes `\pushToTag` et `\appendToTag` permettent d'insérer du matériau, qu'il soit antérieur ou postérieur, à des `éléments` d'une construction musicale existante. La musique séquentielle ou simultanée comporte assurément des `éléments` :

```
music = { \tag #'here { \tag #'here <<c''>> } }

{
  \pushToTag #'here c'
  \pushToTag #'here e'
  \pushToTag #'here g' \music
  \appendToTag #'here c'
  \appendToTag #'here e'
  \appendToTag #'here g' \music
}
```



Ces deux instructions sont affectées d'une balise, le matériau à raccorder à chaque instance de la balise, et l'expression balisée.

Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section 3.3.1 [Insertion de fichiers LilyPond], page 541, [Regroupement automatique de parties], page 189.

Problèmes connus et avertissements

L'application d'un `\relative` à une expression musicale obtenue par filtrage à l'aide de `\keepWithTag` ou `\removeWithTag` peut générer des changements d'octave, puisque seules les hauteurs récupérées dans ce filtre seront prises en considération. Une instruction `\relative` qui précède les commandes `\keepWithTag` ou `\removeWithTag` permet d'éviter ce risque, dans la mesure où elle viendra « recaler » ces hauteurs récupérées.

Globalisation des réglages

Vous pouvez regrouper dans un fichier indépendant vos réglages personnels que vous inclurez au besoin :

```
lilypond -dinclude-settings=MES_REGLAGES.ly MA_PARTITION.ly
```

Vous pouvez ainsi stocker dans un fichier séparé vos réglages en matière de format de papier, de fontes utilisées ou vos définitions particulières. Selon le fichier de réglages que vous mentionnerez, vous obtiendrez facilement différentes éditions à partir d'une même source quelle qu'elle soit.

Cette technique peut s'utiliser en combinaison avec des feuilles de styles, comme indiqué au chapitre Section "Feuilles de style" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Feuilles de style" dans *Manuel d'initiation*, Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section 3.3.1 [Insertion de fichiers LilyPond], page 541.

3.3.3 Caractères spéciaux

Codage du texte

LilyPond utilise le jeu de caractères défini par le consortium Unicode et la norme ISO/CEI 10646. Chaque caractère est identifié par un nom unique et associé à un point de code, ce qui permet dans l'absolu de couvrir tous les langages. Unicode permet de coder tous les caractères utilisés par toutes les langues écrites du monde. LilyPond utilise le codage UTF-8 (UTF pour *Unicode Transformation Format*) qui permet de représenter les caractères latins sur un octet et les autres sur une longueur allant jusqu'à quatre octets.
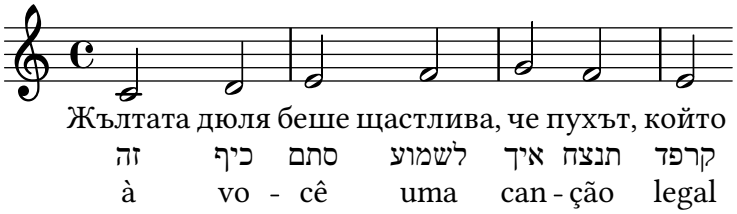
L'apparence réelle des caractères est déterminée par les glyphes ou graphèmes tels que définis dans les différentes polices disponibles. Une police, ou une fonte, définit la mise en correspondance d'un sous-ensemble de points de code unicode en glyphes. LilyPond recourt à la bibliothèque Pango pour assurer le rendu des textes multilingues.

LilyPond n'effectue aucune conversion d'encodage que ce soit. Ceci implique donc que tout texte – un titre, des paroles ou même une instruction musicale – comportant des caractères non ASCII soit codé en UTF-8. Le plus sûr moyen de saisir du texte de la sorte consiste à utiliser un éditeur supportant l'unicode et à enregistrer vos fichier en UTF-8. C'est le cas pour la plupart des éditeurs actuels, que ce soit vim, Emacs, jEdit et Gedit. Tous les systèmes Windows postérieurs à NT utilisent Unicode en natif ; même Notepad est capable d'éditer et sauvegarder un fichier en UTF-8 – sans parler de l'excellente alternative qu'est BabelPad.

La compilation d'un fichier LilyPond comportant des caractères non ASCII qui n'aurait pas été enregistré dans l'encodage UTF-8 vous renverra l'erreur

```
FT_Get_Glyph_Name () erreur : invalid argument
```

Voici un exemple utilisant du texte en cyrillique, en hébreux et en portugais.



Unicode

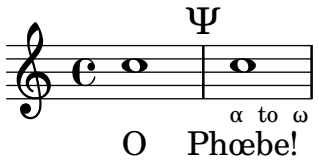
Lorsque vous avez besoin d'un caractère dont vous connaissez le point de code mais que votre éditeur ne permet pas de saisir directement, vous pouvez utiliser les instructions `\char ##xhhhh` ou `\char #dddd` au sein d'un bloc `\markup` – `hhhh` et `dddd` correspondant respectivement à la valeur hexadécimale ou décimale. Même s'il est inutile de saisir les zéros superflus, il est de bon ton de stipuler les quatre caractères formant la représentation hexadécimale. Évitez cependant l'encodage UTF-8 d'un point de code après un `\char` ; les encodages UTF-8 comprennent un bit supplémentaire indiquant le nombre d'octets. Une table de correspondance entre les codes Unicode et le nom des caractères ainsi que leur code hexadécimal est disponible sur le site du consortium Unicode, `http://www.unicode.org/`.

Par exemple, `\char ##x03BE` et `\char #958` correspondent tous deux au caractère unicode U+03BE, dénommé « Greek Small Letter Xi ».

Quel que soit le point de code spécifié de cette manière, il ne vous sera alors pas nécessaire d'enregistrer votre fichier en UTF-8. Vous devrez toutefois disposer d'une fonte contenant ce caractère qui soit accessible à LilyPond.

L'exemple suivant illustre la manière d'insérer un caractère sous sa forme hexadécimale, à la fois dans un repère, dans une articulation, dans des paroles et dans du texte indépendant.

```
\score {
  \relative {
    c''1 \mark \markup { \char ##x03A8 }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2021" \char ##x00A9 }
```



Copyright 2008--2021 ©

Le signe *copyright* dans le champ de titrage consacré s'inscrit de la manière suivante :

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

Équivalents ASCII

Dès lors que vous aurez inclus la liste de leur équivalent ASCII, LilyPond reconnaîtra un certain nombre de caractères spéciaux :

```
\paper {
  #(include-special-characters)
}

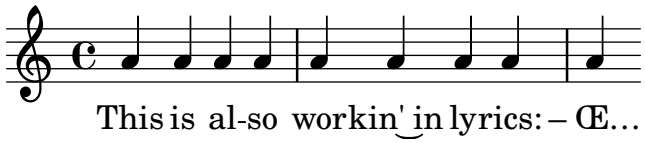
\markup "&flqq; &ndash; &OE;uvre incomplète&hellip; &frqq;"

\score {
  \new Staff { \repeat unfold 9 a'4 }
```

```
  \addlyrics {
    This is al -- so wor -- kin'~in ly -- rics: &ndash;_&OE;&hellip;
  }
}

\markup \column {
  "The replacement can be disabled:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète… »



The replacement can be disabled:
– Œ …
– &OE; …

L'extension de cette liste est possible aussi bien de manière globale :

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

qu'en un point particulier de votre source :

```
\markup \replace #'(("100" . "hundred")
                    ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

Ces alias ne pourront plus, quant à eux, faire l'objet d'un remplacement.

Voir aussi

Manuel de notation : Section A.13 [Liste des caractères spéciaux], page 803.

Fichiers d'initialisation : `ly/text-replacements.ly`.

3.4 Contrôle des sorties

3.4.1 Extraction de fragments musicaux

LilyPond permet d'extraire des fragments d'une partition, une fois definis explicitement le ou les emplacements de la musique concernés au sein du bloc `\layout` du fichier source, grâce à la fonction `clip-regions`, puis en lançant `lilypond` avec l'option `-dclip-systems`.

```
\layout {
  clip-regions
  = #(list
      (cons
       (make-rhythmic-location 5 1 2)
       (make-rhythmic-location 7 3 4)))
}
```

L'exemple ci-dessus permet d'extraire un seul fragment *débutant* après une blanche dans la cinquième mesure (`5 1 2`) et *finissant* après trois noires dans la septième mesure (`7 3 4`).

D'autres fragments seront extraits dès lors que d'autres paires de `make-rhythmic-location` auront été ajoutées à la liste de `clip-regions` présente dans le bloc `\layout`.

Chaque fragment sera généré individuellement sous la forme d'un fichier `EPS`, converti en `PDF` ou `PNG` selon le format que vous aurez stipulé. La musique extraite est rendue comme si elle avait été littéralement « découpée » dans la partition. Par voie de conséquence, un fragment dépassant une ligne fera l'objet d'autant de fichiers séparés que de lignes de la partition complète.

Voir aussi

Manuel de notation : Section 4.2.1 [Le bloc `\layout`], page 583.

Manuel d'utilisation : Voir Section "Utilisation en ligne de commande" dans *Utilisation des programmes*.

3.4.2 Ignorer des passages de la partition

Dans un travail de transcription ou de recopie de la musique, ce qui vous intéresse plus particulièrement se situe à la fin, là même où vous en êtes dans la notation. Dans le but de gagner du temps dans le processus de correction, vous pouvez « escamoter » le reste et ne générer que les dernières mesures en définissant une variable +particulière en début de fichier, comme ceci :

```
showLastLength = R1*5
\score { ... }
```
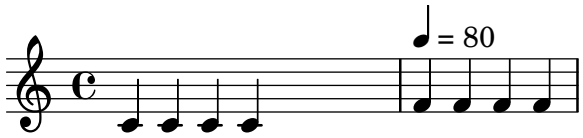
Ceci aura pour effet de ne générer que les cinq dernières mesures – si tant est que le morceau soit à 4/4 – de tous les `\score` de votre fichier. Dans le cas d'un œuvre conséquente, cette pratique s'avère fort utile puisqu'elle évite de tout générer. Vous pourriez aussi être amené à retravailler le début d'une œuvre, pour y ajouter une partie par exemple, auquel cas c'est la propriété `showFirstLength` que vous utiliserez.

Vous pouvez contrôler très finement les parties à escamoter, grâce au commutateur `Score.skipTypesetting` : lorsqu'il est activé, aucune gravure n'est réalisée. En tant que propriété du contexte `+Score`, il affecte toutes les voix et portées – voir +[Score – le père de tous les contextes], page 629.

Ce commutateur agit aussi sur la sortie MIDI. Notez bien que tous les événements seront escamotés, y compris les changements de tempo ou d'instrument qui interviendraient avant que `skipTypesetting` ne +soit désactivé.

```
\relative c' {
  c4 c c c
  \set Score.skipTypesetting = ##t
```

```
  d4 d d d
  \tempo 4 = 80
  e4 e e e
  \set Score.skipTypesetting = ##f
  f4 f f f
}
```



Commandes prédéfinies

`showLastLength`, `showFirstLength`.

Voir aussi

Manuel de notation : Section 5.1 [Contextes d'interprétation], page 628, Section A.17 [Liste des propriétés de contexte], page 813, [Score – le père de tous les contextes], page 629.

3.4.3 Formats de sortie alternatifs

En matière de partition imprimable, LilyPond génère par défaut des documents au format PostScript (PS) et Portable Document Format (PDF). Vous pouvez aussi obtenir des documents au format Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) ou Portable Network Graphics (PNG) dès lors que vous aurez lancé LilyPond en ligne de commande avec l'option *ad hoc* – voir Voir Section "Utilisation en ligne de commande" dans *Utilisation des programmes* à ce sujet.

Sortie SVG

La sortie SVG peut accessoirement contenir des métadonnées pour les *grobs* (objets graphiques) tels que têtes de notes, silences, etc. Ces métadonnées peuvent correspondre aux attributs standards du format SVG comme `id` et `class`, ou bien à des attributs personnalisés. Les attributs et leur valeur se spécifient à l'aide d'une dérogation à la propriété `output-attributes` d'un *grob* par une liste associative (alist) en Scheme. Les valeurs peuvent être des nombres, chaînes ou symboles comme, par exemple :

```
{
  \once \override NoteHead.output-attributes =
  #'((id . 123)
     (class . "ceci cela")
     (data-quelconque . quelquechose))
  c
}
```

Le code ci-dessus produira la balise `<g>` (group) suivante dans le fichier SVG :

```
<g id="123" class="ceci cela" data-quelconque="quelquechose">
  ...NoteHead grob SVG elements...
</g>
```

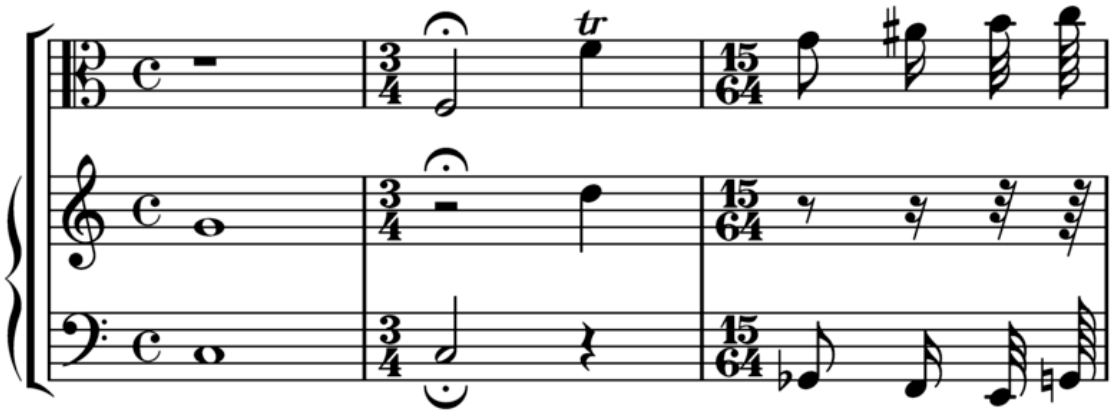
La balise `<g>` contient tous les éléments SVG d'un *grob* donné ; certains *grobs* génèrent de multiples éléments SVG. Dans la syntaxe SVG, le préfixe `data-` s'utilise pour les attributs de métadonnée personnalisée non-standard.

3.4.4 Changement des fontes musicales
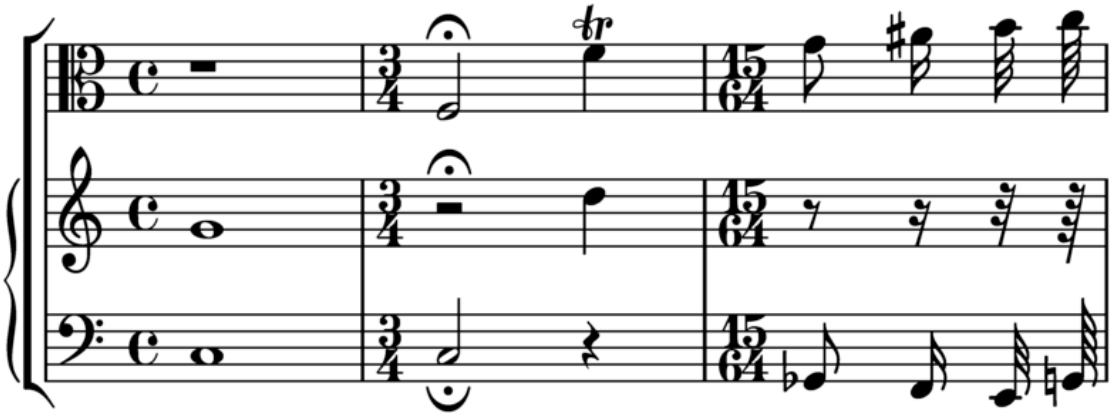
Gonville est une alternative au jeu de glyphes *Feta* inclus dans la fonte Emmentaler que LilyPond utilise par défaut. Vous pouvez la télécharger à partir de

`http://www.chiark.greenend.org.uk/~sgtatham/gonville/` (`http://www.chiark.greenend.org.uk/~sgtatham/gonville/`)

Voici quelques mesures utilisant la police Gonville :



Et ces même mesures avec les glyphes *Feta* de LilyPond :



Instructions d'installation

- Téléchargez puis extrayez les fichiers de fonte.
- Copiez les fichiers

```
gonville-11.otf
gonville-13.otf
gonville-14.otf
gonville-16.otf
gonville-18.otf
gonville-20.otf
gonville-23.otf
gonville-26.otf
gonville-brace.otf
```

dans le dossier `.../share/lilypond/current/fonts/otf` ou `.../share/lilypond/X.Y.Z/fonts/otf`.

- Si vous disposez des fichiers `gonville-*.svg` et `gonville-*.woff`, copiez les dans `.../share/lilypond/current/fonts/svg` ou `.../share/lilypond/X.Y.Z/fonts/svg`.

Pour de plus amples informations, reportez-vous à Section "Autres sources de documentation" dans *Manuel d'initiation*.

Il est à noter que les fichiers `gonville-*.otf` sont destinés aux moteurs `ps` et `eps` (pour obtenir des fichiers PDF ou PostScript) ; les fichiers `gonville-*.svg` sont destinés au moteur `svg` sans l'option `svg-woff` ; les fichiers `gonville-*.woff` sont destinés au moteur `svg` avec l'option `svg-woff`. Pour de plus amles informations, consultez Voir Section "Options avancées de lilypond" dans *Utilisation des programmes*.

La syntaxe suivante substitue aux fontes musicales (générale et accolades) les fontes Gonville.

```
\paper {
  #(define fonts
     (set-global-fonts
      #:music "gonville"
      #:brace "gonville"
    ))
}
```

En tout état de cause, tout appel à `set-global-fonts` réinitialise aussi bien les fontes musicales que les fontes textuelles. Dès lors que l'une de ces catégories n'est pas mentionnée sera utilisée la fonte par défaut y afférente.
Par ailleurs, chaque appel à `set-global-fonts` modifie les fontes du `\book` qui le suit, qu'il ait été créé explicitement ou non. Par voie de conséquence, chaque `\book` peut disposer de son propre jeu de fontes principales s'il est précédé d'un appel à `set-global-fonts`. Pour plus d'informations, voir [Choix des fontes par défaut], page 284.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : [Choix des fontes par défaut], page 284, Section A.8 [La fonte Emmentaler], page 718.

Problèmes connus et avertissements

Gonville ne permet pas de générer de la notation ancienne, et certains glyphes ajoutés depuis lors aux jeux de caractères en sont absent. Consultez le site de l'auteur pour de plus amples informations ainsi qu'à propos des conditions d'utilisation.

Autres fontes musicales

Si vous disposez d'autres fontes musicales telles que *nomfonte*`-*.otf`, *nomfonte*`-*.svg` et *nomfonte*`-*.woff`, vous pouvez les utiliser de façon comparable à Gonville.

Autrement dit, copiez les fichiers *nomfonte*`-*.otf` dans `.../share/lilypond/current/fonts/otf` ou `.../share/lilypond/X.Y.Z/fonts/otf`. Si vous disposez de fichiers *nomfonte*`-*.svg` et *nomfonte*`-*.woff`, copiez les dans `.../share/lilypond/current/fonts/svg` ou `.../share/lilypond/X.Y.Z/fonts/svg`.

Note : à ce jour, et pour fonctionner correctement, LilyPond requiert la présence des suffixes suivants dans les dossiers d'instalation : `-11`, `-13`, `-14`, `-16`, `-18`, `-20`, `-23`, `-26` et `-brace`. Par exemple, `emmentaler-11.otf`, `emmentaler-20.svg`, `emmentaler-brace.woff`, etc.

La syntaxe suivante substitue aux fontes musicales (générale et accolades) les fontes *nomfonte*.

```
\paper {
  #(define fonts
     (set-global-fonts
```

```
        #:music "nomfonte" ; fichier de fonte sans suffixe ni extension
        #:brace "nomfonte" ; fichier de fonte sans suffixe ni extension
      ))
  }
```

Note : pour les catégories `music` et `brace`, le nom du fichier de fonte se spécifie sans suffixe ni extension.

3.5 Génération de fichiers MIDI

LilyPond peut produire des fichiers conformes au standard MIDI (Musical Instrument Digital Interface), ce qui permet de vérifier le rendu à l'oreille grace à un logiciel ou un périphérique sachant interpréter le MIDI. L'écoute du rendu en MIDI permet de contrôler aisément ce que vous avez saisi : octaves et altérations erronées heurteront votre oreille avertie !

Les fichiers MIDI, contrairement aux fichiers AAC, MP3 ou Vorbis, ne contiennent pas de son et nécessitent donc le recours à un logiciel supplémentaire pour les écouter.

3.5.1 Notation prise en compte dans le MIDI

LilyPond retranscrit par défaut dans un fichier MIDI les éléments de notation suivants :

- les marques de respiration,
- les accords nommés,
- les crescendos et decrescendos s'étendant sur plusieurs notes – le volume s'ajuste linéairement entre les deux extrêmes,
- les indications de nuance, de `ppppp` à `fffff`, y compris `mp`, `mf` et `sf`,
- les microtonalités, mais *pas* sous forme d'accord ; leur rendu nécessite cependant un lecteur qui prenne en charge la modulation,
- les paroles,
- les hauteurs,
- le rythme sous forme de durée de note, y compris les n-olets,
- les articulations « simples » comme staccato, staccatissimo, accent, marcato et portato,
- les changements de tempo indiqués par un `\tempo`,
- les liaisons de tenue,
- les tremolos, excepté ceux utilisant la syntaxe « `:`[*nombre*] ».

Spatialisation, balance, expression, réverbération et chorus peuvent se contrôler à l'aide de propriétés de contexte – voir Section 3.5.8 [Propriétés de contextes et effets MIDI], page 566.

En combinaison avec le script `articulate`, d'autres éléments seront aussi reportés en MIDI :

- les appoggiatures – celles-ci prendront la moitié de la valeur, dépourvue de point, de la note qui les suit – par exemple,

  ```
  \appoggiatura c8 d2.
  ```

  le do (noté `c`) prendra la valeur d'une noire.
- les ornements (mordants, trilles et groupettos, etc.),
- rallentando, accelerando, ritardando et a tempo,
- les liaisons y compris de phrasé,
- les tenutos.

Voir Section 3.5.9 [Amélioration du rendu MIDI], page 567.

3.5.2 Notation non prise en compte dans le MIDI

Certains éléments de notation ne peuvent être retranscrits dans un fichier MIDI :

- les articulations autres que staccato, staccatissimo, accent, marcato et portato,
- les crescendos et decrescendos sur *une seule* note,
- les points d'orgue,
- la basse chiffrée,
- les glissandos,
- les chutes ou sauts,
- les accords en microtonalité,
- le rythme indiqué sous forme d'annotation, comme « swing »,
- les changements de tempo indiqués sous forme d'annotation (sans `\tempo`),
- les trémolos indiqués par la syntaxe « `:`[*nombre*] ».

3.5.3 Le bloc MIDI

LilyPond générera un fichier MIDI dès que vous ajouterez un bloc `\midi`, même vide, au sein du bloc `\score` :

```
\score {
  ...musique...
  \layout { }
  \midi { }
}
```

> **Note :** Lorsque le bloc `\score` contient uniquement un bloc `\midi` (autrement dit pas de bloc `\layout`), LilyPond produira uniquement la sortie MIDI – aucun support visuel ne sera généré.

L'extension par défaut des fichiers MIDI générés (`.midi`) peut se modifier en ligne de commande :

```
lilypond -dmidi-extension=mid MonFichier.ly
```

Une autre manière de procéder consiste à placer la ligne suivante au début de votre fichier source, avant l'ouverture de tout bloc `\book`, `\bookpart` ou `\score` – voir Section 3.1.5 [Structure de fichier], page 513 :

```
#(ly:set-option 'midi-extension "mid")
```

Voir aussi

Manuel de notation : Section 3.1.5 [Structure de fichier], page 513.

Fichiers d'initialisation : `scm/midi.scm`.

Problèmes connus et avertissements

Le standard MIDI dispose de 15 canaux plus un (le numéro 10) affecté aux percussions. Les portées sont assignées l'une après l'autre à un canal. Dans la mesure où une partition comporte plus de 15 portées, les portées au-delà de la quinzième partageront un même canal MIDI, sans toutefois l'écraser. Ceci peut entraîner des conflits au niveau des canaux en raison des propriétés MIDI, notamment l'instrument utilisé.

L'utilisation d'un bloc `midi` dans le cadre d'une musique polymétrique peut engendrer des avertissements inopinés quant au contrôle des barres de mesure. En pareil cas, il suffit de déplacer, au sein du bloc `midi`, le `Timing_translator` du contexte `Score` au contexte `Staff`.

```
\midi {
```

```
  \context {
    \Score
    \remove "Timing_translator"
  }
  \context {
     \Staff
     \consists "Timing_translator"
  }
}
```

3.5.4 Gestion des nuances en MIDI

Le volume général de la sortie MIDI peut se définir, ainsi que ses modulations, en fonction des indications de nuance et les volumes relatifs entre les différents instruments.

Les indications de nuance se traduisent automatiquement en niveau de volume dans l'amplitude disponible en MIDI alors que crescendos et diminuendos auront une progression linéaire entre les extrêmes.

Indication des nuances en MIDI

Les indications de nuance, de `ppppp` à `fffff` – y compris `mp`, `mf` et `sf` – ont des valeurs prédéfinies. Ce coefficient est alors appliqué pour corriger le volume général de façon à obtenir le niveau sonore qui sera retranscrit dans le fichier de sortie pour la nuance considérée. Nous allons, par défaut, de 0,25 pour un *ppppp* à 0,95 pour un *fffff*. Les correspondances entre nuance et fraction de volume sont répertoriées dans le fichier `scm/midi.scm`.

Morceaux choisis

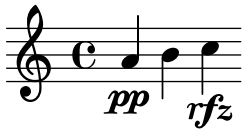
Création de nuance particulière pour la sortie MIDI

L'exemple suivant illustre la manière de créer une indication de nuance, absente de la liste par défaut, et de lui assigner une valeur spécifique utile à la sortie MIDI.

L'indication de nuance `\rfz` (*rinforzando*) se voit attribuer une valeur de `0.9`.

```
#(define (myDynamics dynamic)
    (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Voir aussi

Fichiers d'initialisation : `ly/script-init.ly`, `scm/midi.scm`.

Morceaux choisis : Section "MIDI" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Réglage du volume en MIDI

Les valeurs extrêmes du volume MIDI des nuances se contrôlent à l'aide des propriétés `midiMinimumVolume` et `midiMaximumVolume` qui agissent au niveau `Score`. Ces propriétés sont effectives dès lors qu'une nuance est indiquée ; une nuance de départ explicite est donc requise pour que le volume soit ajusté dès le début de la partition. Vous pouvez alors modifier la fraction correspondant à chaque nuance à l'aide de la formule

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * fraction
```

Voici comment ajuster les nuances tout en limitant l'amplitude du volume entre 0,2 et 0,5 :

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = "flute"
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinet"
      ... music ...
    }
  >>
  \midi {
    \context {
      \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```

La définition de l'amplitude du volume MIDI au niveau d'un contexte `Staff` – grace aux propriétés `midiMinimumVolume` et `midiMaximumVolume` – permet en quelque sorte d'égaliser un instrument MIDI.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "flute"
    \set Staff.midiMinimumVolume = #0.7
    \set Staff.midiMaximumVolume = #0.9
    ... musique ...
  }
  \midi { }
}
```

Dans le cas d'une partition à plusieurs portées et différents instruments, les volumes relatifs entre les différents instruments se gèrent individuellement :

```
\score {
```

```
  <<
    \new Staff {
      \set Staff.midiInstrument = "flute"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinet"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      ... music ...
    }
  >>
  \midi { }
}
```

La clarinette de cet exemple jouera relativement moins fort que la flûte.

En l'absence de tout réglage des propriétés de volume, LilyPond appliquera cependant un léger degré d'égalisation pour certains instruments – voir `scm/midi.scm`.

Morceaux choisis

Réglage de l'égalisation par défaut des instruments MIDI

L'égaliseur basique peut être modifié par la définition d'une nouvelle procédure Scheme `instrumentEqualizer` au sein du contexte `Score`. Cette procédure prend en unique argument le nom d'un instrument MIDI et renverra une paire de fractions correspondant aux minimum et maximum de volume alloué à cet instrument.

Dans l'exemple suivant sont réglés les volumes relatifs de la flûte et de la clarinette.
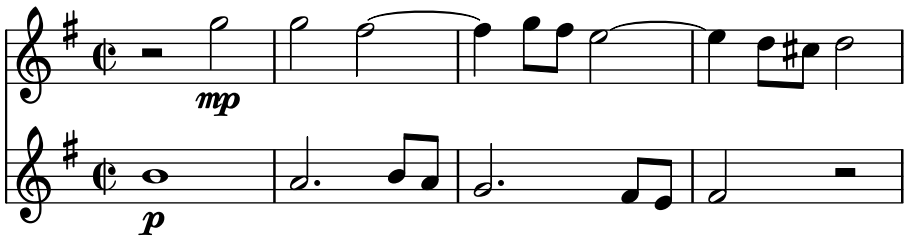
```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = "flute"
      \new Voice \relative {
        r2 g''\mp g fis~
        4 g8 fis e2~
```

```
        4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = "clarinet"
      \new Voice \relative {
        b'1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {  }
}
```



Voir aussi

Fichiers d'initialisation : `scm/midi.scm`.

Manuel de notation : Section 4.2 [Mise en forme de la partition], page 583.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les modifications apportées au volume MIDI n'interviennent que sur l'attaque d'une note, en conséquence de quoi crescendos et decrescendos n'affecteront pas le volume s'ils se produisent sur une même et unique note.

Réglage de propriétés dans le bloc MIDI

Le bloc `\midi` peut contenir des aménagements pour certains contextes, la définition de contextes particuliers ou du code permettant de déterminer la valeur de certaines propriétés.

```
\score {
  ... music ...
  \midi {
    \tempo 4 = 72
  }
}
```

Le tempo est ici réglé à 72 noires par minute. Une indication de tempo inscrite dans le bloc `\midi` ne sera pas reportée sur la partition imprimable. Cependant, tout `\tempo` mentionné dans le bloc `\score` sera répercuté dans la sortie MIDI.

Placée au sein d'un bloc `\midi`, la commande `\tempo` détermine des propriétés lors de la phase d'interprétation de la musique et dans le contexte de définition des sorties. Elle est alors considérée comme une modification de contexte.

La syntaxe permettant de définir un contexte pour le `\midi` est en tout point identique à celle que vous utilisez dans le bloc `\layout` :

```
\score {
  ... musique ...
  \midi {
    \context {
      \Voice
      \remove "Dynamic_performer"
    }
  }
}
```

Ces quelques lignes ont pour effet de supprimer l'application des nuances à la sortie MIDI. Vous aurez noté que les modules de traduction de LilyPond en matière de son s'appelent *performers* – des « interprètes ».

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section 4.2 [Mise en forme de la partition], page 583, Section 1.3 [Signes d'interprétation], page 126.

Fichiers d'initialisation : `ly/performer-init.ly`.

Morceaux choisis : Section "MIDI" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Certains lecteurs MIDI ne rendent pas correctement les changements de tempo.

Les modifications de `midiInstrument` ou autres options MIDI en début de portée peuvent se retrouver dédoublées dans la sortie MIDI.

3.5.5 Gestion des instruments MIDI

L'instrument MIDI est déterminé par la propriété `midiInstrument`, au sein d'un contexte `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "glockenspiel"
    ... music ...
  }
  \midi { }
}
```

ou

```
\score {
  \new Staff \with {midiInstrument = "cello"} {
    ... music ...
  }
  \midi { }
}
```

Lorsque l'instrument choisi ne correspond pas exactement à l'une des dénominations consacrées, LilyPond le remplacera par un piano de concert (`"acoustic grand"`) – voir Section A.6 [Instruments MIDI], page 715.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section A.6 [Instruments MIDI], page 715, Section 4.2 [Mise en forme de la partition], page 583.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Fichiers d'initialisation : `scm/midi.scm`.

Problèmes connus et avertissements

Les percussions gérées par un contexte `DrumStaff` sont affectées directement au canal 10 qui leur est réservé. Certains instruments, tels le xylophone, le marimba, le vibraphone ou les timbales, se traitent cependant comme des instruments « classiques » puisqu'ils sont capables d'émettre des hauteurs différentes ; leur notation relève donc d'un contexte `Staff` standard, et non d'un `DrumStaff` pour pouvoir être rendus correctement en MIDI. Une liste complète des percussions affectées au canal 10 (`channel 10 drum-kits`) est disponible dans le fichier `scm/midi.scm` – voir Section "Autres sources de documentation" dans *Manuel d'initiation*.

3.5.6 Gestion des répétitions en MIDI

Les reprises de toutes sortes peuvent être rendues dans le fichier MIDI. Il suffit pour cela de recourir à la fonction `\unfoldRepeats`, qui développe toutes les reprises.

```
\score {
  \unfoldRepeats {
    \repeat tremolo 8 { c'32 e' }
    \repeat percent 2 { c''8 d'' }
    \repeat volta 2 { c'4 d' e' f' }
    \alternative {
      \volta 1 { g' a' a' g' }
      \volta 2 { f' e' d' c' }
    }
  }
  \midi { }
}
```

Lorsque l'on veut utiliser `\unfoldRepeats` seulement pour le rendu MIDI, il faut établir **deux** blocs `\score` : un pour le MIDI, avec des reprises explicites, et l'autre pour la partition, avec des reprises notées sous forme de barres de reprise, de trémolo ou de symboles de pourcentage. Par exemple

```
\score {
  ... musique ...
  \layout { }
}
\score {
  \unfoldRepeats ... musique ...
  \midi { }
}
```

Dans une partition comportant plusieurs voix, le développement des reprises ne sera effectif en MIDI qu'à la condition que ces reprises soient mentionnée correctement dans **toutes** les voix.

Voir aussi

Manuel de notation : Section 1.4 [Répétitions et reprises], page 156.

3.5.7 Affectation des canaux MIDI

Lorsque LilyPond génère un fichier MIDI à partir d'une partition, chaque note contenue dans cette partition sera automatiquement assignée à un canal MIDI, celui sur lequel elle devrait être jouée quand elle est transmise à un périphérique MIDI. Chaque canal MIDI dispose d'un certain nombre de contrôles pour, par exemple, sélectionner l'instrument qui jouera les notes de ce canal ou bien demander au périphérique MIDI d'appliquer différents effets au son produit sur ce canal. En tout état de cause, chaque contrôle d'un canal MIDI ne peut se voir affecté que d'une seule valeur à la fois – celle-ci peut toutefois être modifiée pour, par exemple, changer d'instrument au milieu du morceau.

Le standard MIDI ne dispose que de 16 canaux par périphérique MIDI. Cette limite du nombre de canaux entraîne une limitation du nombre d'instruments pouvant jouer de concert.

LilyPond crée une piste MIDI séparée pour chaque portée (ou chaque instrument ou voix selon la valeur de `Score.midiChannelMapping`) ainsi que pour chaque contexte de paroles. Il n'y a pas de limite au nombre de pistes.

Afin de contourner la limitation du nombre de canaux MIDI, LilyPond dispose de différents modes d'allocation d'un canal MIDI grâce à la propriété de contexte `Score.midiChannelMapping`. Dans tous les cas, lorsque la limite au nombre de canaux est atteinte, LilyPond repart du canal 0, ce qui peut affecter des notes au mauvais instrument. Cette propriété de contexte peut prendre les valeurs suivantes :

`'staff`

Allocation d'un canal MIDI particulier à chacune des portées de la partition (option par défaut). Toutes les notes de toutes les voix d'une même portée partageront le canal MIDI affecté à la portée qui les englobe, et toutes seront encodées dans la même piste.

La limite des 16 canaux s'applique au nombre total de portées augmenté des contextes de paroles même si les paroles MIDI n'occupent pas de canal MIDI.

`'instrument`

Allocation d'un canal MIDI particulier à chaque instrument MIDI tel que spécifié dans la partition. En d'autre termes, des notes jouées par un même instrument MIDI partageront le même canal MIDI (et la même piste), même si elles proviennent de voix ou portées différentes.

Dans ce cas particulier, les contextes de paroles ne sont pas pris en compte dans la limite des 16 canaux, puisqu'ils ne sont pas assignés à un instrument MIDI, ce qui permet une meilleure allocation des canaux MIDI lorsque le nombre de portées et de contextes de paroles dépasse 16.

`'voice`

Allocation d'un canal MIDI particulier à chaque voix de la partition portant un nom unique parmi les voix de la portée considérée. Des voix appartenant à des portées différentes seront toujours affectées à des canaux MIDI différents, mais deux voix partageant une même portée partageront le même canal MIDI dès lors qu'elles porteront le même nom. Dans la mesure où `midiInstrument` et les différents contrôles d'effets MIDI sont des propriétés affectant le contexte de portée, ils ne peuvent se déterminer individuellement pour une voix. La première voix adoptera l'instrument et les effets spécifiés pour cette portée, et les voix dénommées différemment de la première se verront attribué l'instrument et le effets par défaut.

Note : l'affectation d'instruments ou d'effets différents aux differentes voix d'une même portée s'obtient dès lors que le `Staff_performer` est déplacé du contexte `Staff` au contexte `Voice` tout en maintenant le `midiChannelMapping` dans le contexte `'staff` ou en le réglant sur `'instrument`.

Par exemple, l'affectation par défaut des canaux MIDI d'une partition peut être réglée sur `'instrument` comme ceci :

```
\score {
  ...musique...
  \midi {
    \context {
      \Score
      midiChannelMapping = #'instrument
    }
  }
}
```

Morceaux choisis
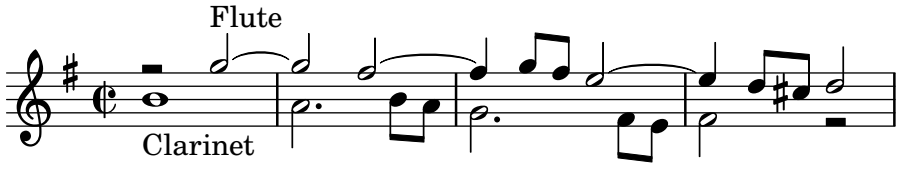
Affectation d'un canal MIDI par voix

Lorsque LilyPond génère un fichier MIDI, chaque portée sera par défaut affectée à un canal, quel que soit le nombre de voix qu'elle contient. Ceci permet d'éviter de se retrouver à court de canaux, sachant qu'il n'y en a que seize de disponibles par piste.

Le fait de déplacer le `Staff_performer` dans le contexte `Voice` permet d'affecter à chaque voix d'une même portée un canal MIDI spécifique. Dans l'exemple suivant, la même portée donnera lieu à deux canaux MIDI différents, chacun étant affecté de son propre `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = #"flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = #"clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
```

```
    \tempo 2 = 72
  }
}
```



3.5.8 Propriétés de contextes et effets MIDI

Les différentes propriétés de contexte qui suivent permettent d'appliquer différents effets MIDI aux notes contenues dans le canal MIDI associé à la portée courante, à l'instrument ou à la voix, selon la valeur affectée à la propriété de contexte `Score.midiChannelMapping` et le contexte dans lequel le `Staff_performer` réside – voir Section 3.5.7 [Affectation des canaux MIDI], page 564.

Une adaptation de ces propriétés de contexte affectera toutes les notes jouées sur ce canal dès leur modification. Certains effets pourront même s'appliquer sur des notes déjà en cours, selon l'implémentation du périphérique de sortie MIDI.

LilyPond dispose des propriétés de contexte suivantes :

`Staff.midiPanPosition`
: La spatialisation (*pan position*) contrôle le positionnement d'un canal MIDI entre les sorties stéréo droite et gauche. Cette propriété de contexte prend en argument une valeur entre -1.0 (`#LEFT`) et 1.0 (`#RIGHT`). Une valeur de -1.0 enverra toute la puissance sonore sur le haut-parleur gauche (le droit sera silencieux), une valeur de 0.0 (`#CENTER`) distribuera équitablement le son entre les haut-parleurs de gauche et de droite, et une valeur de 1.0 enverra tout le son sur le haut-parleur de droite. Des valeurs entre -1.0 et 1.0 permettent d'obtenir une répartition du son entre les sorties gauche et droite d'un équipement stéréophonique.

`Staff.midiBalance`
: La balance stéréo d'un canal MIDI. Tout comme la spatialisation, cette propriété de contexte prend en argument une valeur comprise entre -1.0 (`#LEFT`) et 1.0 (`#RIGHT`). Elle permet de faire varier le volume relatif envoyé aux deux haut-parleurs stéréo sans pour autant affecter la distribution des signaux stéréo.

`Staff.midiExpression`
: Le niveau d'expression, en tant que fraction du niveau maximum de volume disponible, à appliquer à un canal MIDI. Un périphérique MIDI combine le niveau d'expression des canaux MIDI et le niveau de nuance de la voix en cours (tel que défini par `\p` ou `\ff`) afin d'obtenir le volume total de chacune des notes de la voix. Un contrôle de l'expression permet, par exemple, d'implémenter des effets de crescendo ou decrescendo sur une note tenue, ce que LilyPond ne sait pas faire automatiquement.

  Le niveau d'expression varie entre 0.0 (sans expression, autrement dit volume à zéro) et 1.0 (volume au maximum).

`Staff.midiReverbLevel`
: Le niveau de réverbération, en tant que fraction du niveau maximum disponible, à appliquer à un canal MIDI. Cette propriété prend en argument une valeur entre 0.0 (pas d'écho) et 1.0 (effet maximal).

`Staff.midiChorusLevel`
: Le niveau de chœur, en tant que fraction du niveau maximum disponible, à appliquer à un canal MIDI. Cette propriété prend en argument une valeur entre 0.0 (pas de chorus) et 1.0 (effet maximal).

Problèmes connus et avertissements

Dans la mesure où les fichiers MIDI ne comportent effectivement aucune donnée audio, les modifications des propriétés de contexte ne se traduisent qu'en requêtes de changement des contrôles du canal MIDI lorsque ces fichiers MIDI sont joués. La manière dont un périphérique MIDI particulier, tel un synthétiseur MIDI logiciel, gérera ces requêtes incluses dans un fichier MIDI dépend complètement de l'implémentation du périphérique : certains d'entre eux pourront simplement ignorer plusieurs, voire toutes ces requêtes. Par ailleurs, la manière dont un périphérique MIDI interprète les différentes valeurs de ces contrôles (en règle générale, le standard MIDI ne fixe le comportement qu'aux valeurs extrêmes de l'amplitude disponible pour chacun des contrôles) et leur modification alors qu'une note sur un canal est tenue, dépend de l'implémentation particulière à ce périphérique.

Lors de la génération d'un fichier MIDI, LilyPond transforme simplement les valeurs fractionnaires dans l'amplitude linéaire en valeurs entières correspondantes (de 0 à 127 et sur 7 octets, ou de 0 à 32767 et sur 14 octets pour les contrôles MIDI supportant une résolution fine). Ces valeurs entières converties sont stockées telles quelles dans le fichier MIDI généré. Pour plus d'information sur la manière dont un périphérique MIDI interprète ces valeurs, se reporter à sa documentation.

3.5.9 Amélioration du rendu MIDI

Le fichier MIDI généré par LilyPond est relativement brut. Il peut toutefois être amélioré en affectant des instruments MIDI ou en réglant certaines propriétés au sein du bloc `\midi`.

Des scripts additionnels permettent d'affiner la manière dont les nuances, articulations et rythme sont rendus en MIDI : le script `articulate` et le script `swing`.

Le script `articulate`

L'utilisation du script `articulate` se fait après avoir ajouté en tête de fichier la commande `\include` appropriée :

```
\include "articulate.ly"
```

Le script créera une sortie MIDI dont les notes seront échelonées de sorte à tenir compte de toute articulation ou changement de tempo. La sortie imprimable sera toutefois modifiée en profondeur, pour refléter littéralement la sortie MIDI.

```
\score {
  \articulate <<
    ... musique ...
  >>
  \midi { }
}
```

Le script `\articulate` tient compte des abréviations telles que les trilles ou groupettos. L'intégralité des éléments traités est répertoriée dans le script lui-même – voir `ly/articulate.ly`.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section 4.2 [Mise en forme de la partition], page 583.

Fichiers d'initialisation : `ly/articulate.ly`.

Note : Dans la mesure où le script `articulate` tend à raccourcir les accords, certaines musiques, notamment pour l'orgue, paraîtront de moins bonne qualité. Les notes dépourvues d'articulation peuvent aussi se voir raccourcies ; pour pallier cet inconvénient, le recours à la fonction `\articulate` devrait ne concerner que de courts fragments, sauf à modifier les valeurs des variables contenues dans le script `articulate`.

Le script `swing`

Le script `swing` procure des fonctions additionnelles qui permettent de jouer des durées égales sur un rythme inégal. L'exemple le plus évident est l'interprétation *swinguée* que l'on trouve en jazz où des croches binaires devraient se jouer de façon ternaire. D'autres interprétations sont toutefois prises en charge.

Ce script doit faire l'objet d'une inclusion en début de fichier source :

```
\include "swing.ly"
```

Le script fournit trois commandes :

- `\tripletFeel` crée un *swing* sur une base de triolet. Elle prend deux arguments : les durées à affecter (typiquement `8` pour des croches) et l'expression musicale sur laquelle l'appliquer.
- `\applySwing` prend un argument supplémentaire avant l'expression musicale : une *liste de pondération* de *n* nombres de ratio exprimant la manière dont doivent être jouées les notes régulières. Par exemple, `#'(2 1)` indique que chaque note devrait se jouer deux fois plus longue que la suivante – en fait, `\tripletFeel` *durée* `{`*musique*`}` est un raccourci de `\applySwing` *durée* `#'(2 1) {`*musique*`}` Des croches chaloupées plus doucement s'obtiennent avec une *liste de pondération* de `#'(3 2)` ou toute autre valeur selon les goûts.

  Cette liste peut prendre plus de deux valeurs, ce qui permet d'adopter un schéma de *groove* plus long ou plus sophistiqué. Par exemple, une impression de samba sur des double-croches peut s'obtenir ainsi :

```
\score {
  \applySwing 16 #'(3 2 2 3) {
    ... music ...
  }
  \midi { }
}
```

- `\applySwingWithOffset` ajoute quant à elle un autre argument entre la *liste de pondération* et l'expression musicale : une durée de décallage, donnée sous forme d'expression `ly:make-moment`. Cette commande devrait s'utiliser lorsque l'expression musicale démarre à contre temps, avec une portion de cycle.

Note : Tout comme avec le script `articulate`, toutes les commandes seront rendues dans la partition imprimable, ce qui résulte en un espacement irrégulier. Ceci peut s'éviter en utilisant le script dans un bloc `\score` dédié à la sortie MIDI.

Une aide et des informations supplémentaires sont directement incluses dans le script – voir `ly/swing.ly`.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section 1.2 [Rythme], page 47.

Fichiers d'initialisation : `ly/swing.ly`.

Problèmes connus et avertissements

- Les constructions `\repeat` et `\repeat unfold` ne sont pas prises en considération lors de la temporisation des notes. Ceci entraînera des problèmes à moins que les durées de tous les fragments répétés soient des multiples entiers du cycle de *swing*.
- Ces fonctions sont agnostiques en matière de métrique et de mesure, raison pour laquelle des décallages doivent être fournis à l'aide de `\applySwingWithOffset` lorsque la musique démarre sur une levée.
- Les appogiatures sont ignorées et laissées telles quelles, tout comme les triolets.

3.6 Extraction d'informations musicales

En plus de générer du graphisme et du MIDI, LilyPond peut présenter l'information musicale sous forme textuelle.

3.6.1 Affichage de notation au format LilyPond

La fonction musicale `\displayLilyMusic` permet d'afficher en notation LilyPond une expression musicale. Le résultat défilera dans le terminal après avoir lancé LilyPond en ligne de commande. Par exemple,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

affichera

```
{ a,4 cis4 e4 fis4 g4 }
```

LilyPond affichera le résultat sous forme de message en console, au milieu de toutes les informations de compilation. Afin d'isoler ces messages et enregistrer le résultat de la fonction `\displayLilyMusic`, pensez à rediriger la sortie vers un fichier.

```
lilypond fichier.ly > affichage.txt
```

Vous noterez que LilyPond ne se contente pas de simplement afficher l'expression musicale, mais procède aussi à son interprétation – du fait que `\displayLilyMusic` renvoie l'expression tout en l'affichant. La simple insertion d'un `\displayLilyMusic` dans une expression musicale permet d'obtenir l'information la concernant.

Si l'instruction `\displayLilyMusic` interprète et affiche des informations sur un fragment, la faire précéder d'un `\void` aura pour effet de l'exclure du fichier résultant.

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
  c1
}
```

3.6.2 Affichage de la musique sous forme d'expression Scheme

Voir Section "Affichage d'expressions musicales" dans *Extension de LilyPond*.

3.6.3 Enregistrement d'événements musicaux dans un fichier

LilyPond vous permet de sauvegarder dans un fichier séparé, sur la base de la portée, les événements musicaux. Vous devrez pour ce faire inclure dans votre fichier maître un fichier d'initialisation spécifique :

```
\include "event-listener.ly"
```

Pour chaque portée que comporte votre partition, vous obtiendrez un fichier `NOMFICHIER-PORTÉENOMMÉE.notes` ou `NOMFICHIER-unnamed-staff.notes`. Notez bien que si plusieurs

portées ne sont pas explicitement nommées, tous leurs événements seront regroupés et mélangés dans le même fichier. Le résultat ressemblera à ceci :

```
0.000   note     57       4   p-c 2 12
0.000   dynamic  f
0.250   note     62       4   p-c 7 12
0.500   note     66       8   p-c 9 12
0.625   note     69       8   p-c 14 12
0.750   rest     4
0.750   breathe
```

Il s'agit d'un tableau dont les colonnes sont délimitées par une tabulation. Chaque ligne comporte deux champs fixes suivis d'un certain nombre de paramètres optionnels.

```
temps type ...paramètres...
```

Ces informations peuvent faire l'objet d'un retraitement par d'autres programmes, comme des scripts python, aux fins de recherche en analyse musicologique ou des expériences à partir du rendu de LilyPond.

Problèmes connus et avertissements

Tous les événements ne sont pas pris en charge par `event-listener.ly`. Il s'agit en premier lieu d'une démonstration, un « proof of concept » du potentiel de LilyPond. Si certains des éléments que vous cherchez à obtenir n'apparaissent pas, recopiez le fichier `event-listener.ly` dans votre répertoire et modifiez-le de telle sorte qu'il travaille selon vos attentes.

4 Gestion de l'espace

L'agencement général d'une partition dépend de trois facteurs interdépendants : la mise en page, les sauts de ligne et l'espacement. Les choix faits en matière d'espacement détermineront la densité de chacun des systèmes, ce qui influera sur le positionnement des sauts de ligne et, par voie de conséquence, sur le nombre de pages de la partition.

En pratique, cette procédure comporte quatre étapes. Dans un premier temps, des distances élastiques *(springs)* sont déterminées sur la base des durées. Sont alors calculées différentes possibilités de saut de ligne, chacune se voyant attribuer un « coefficient de laideur », puis est estimée la hauteur de chaque système. LilyPond opte enfin pour la combinaison entre sauts de page et de ligne qui offre la meilleure occupation de l'espace, tant horizontalement que verticalement.

Les réglages qui influencent la mise en forme se placent dans deux blocs différents. Le bloc `\paper {...}` étudié à la rubrique Section 4.1 [Mise en forme de la page], page 571, contient les réglages applicables à toutes les partitions d'une partie ou de l'intégralité d'un ouvrage – tels que format du papier, impression ou non des numéros de page, etc. Quant au bloc `\layout {...}`, qui fait l'objet de la rubrique Section 4.2 [Mise en forme de la partition], page 583, il détermine la mise en forme de la musique : le nombre de systèmes utilisés, l'espacement des regroupements de portées, etc.

> **Note :** Vous verrez au fil de ce chapitre apparaître certains termes dont la traduction vous semblera assurément erronée. Il n'en est cependant rien : certains termes techniques ont une histoire particulière selon leur langue d'origine. Ainsi le vocable anglais *Ragged* signifie en lambeau, en loques ; dans l'univers typographique, un maître français voit un alignement à gauche – il dira « au fer à gauche » – alors que son homologue anglophone constate un *ragged-right* – donc du vide à droite.

4.1 Mise en forme de la page

Nous allons examiner ici les options qui contrôlent la mise en forme des pages attachées au bloc `\paper`.

4.1.1 Le bloc \paper

Des blocs `\paper` peuvent apparaître à trois différents endroits et former ainsi une hiérarchie :

- En début de fichier source, avant même tout bloc `\book`, `\bookpart` ou `\score`.
- Au sein d'un bloc `\book` et indépendamment de tout bloc `\bookpart` ou `\score` qu'il pourrait contenir.
- Au sein d'un bloc `\bookpart`, mais en dehors de tout bloc `\score`.

Un bloc `\paper` ne doit donc en aucun cas prendre place au sein d'un bloc `\score`.

Les valeurs des différents champs seront filtrées en respectant cette hiérarchie ; les valeurs définies le plus haut persisteront à moins d'être remplacées à un niveau hiérarchique inférieur.

Plusieurs blocs `\paper` peuvent cohabiter à un même niveau, notamment en raison de la présence d'inclusion de fichiers. Dans une telle éventualité, les champs sont regroupés par niveau, la dernière valeur rencontrée ayant préséance en cas de doublon.

Peuvent apparaître dans un bloc `\paper` :

- la fonction Scheme `set-paper-size`,
- des variables propres au bloc `\paper` qui viendront adapter la mise en page,

- la définition des différents *markups* qui personnaliseront la mise en forme des entêtes et pieds de page ainsi que des titrages.

La fonction `set-paper-size` fait l'objet de la rubrique qui suit – Section 4.1.2 [Format du papier et adaptation automatique], page 572. Les variables du bloc `\paper` chargées de la mise en page sont abordées plus loin dans ce chapitre. Quant aux définitions relatives aux *markups* des entête, pied de page et titrage, elles sont étudiées à la rubrique Section 3.2.2 [Titrages personnalisés], page 523.

La plupart des variables gérant le papier ne sont fonctionnelles que lorsque mentionnées dans un bloc `\paper`. Certaines, qui peuvent toutefois apparaître dans un bloc `\layout`, sont référencées à la rubrique Section 4.2.1 [Le bloc `\layout`], page 583.

Sauf mention contraire, toutes les variables du bloc `\paper` qui correspondent à des dimensions sont exprimées en millimètre – vous pouvez bien entendu spécifier un autre système de mesure. Voici comment, par exemple, définir la marge haute (`top-margin`) à dix millimètres :

```
\paper {
  top-margin = 10
}
```

Si vous préférez lui affecter une valeur de 0,5 pouce, vous devrez mentionner le suffixe d'unité `\in` :

```
\paper {
  top-margin = 0.5\in
}
```

LilyPond accepte les suffixes d'unité `\mm`, `\cm`, `\in` et `\pt`. Ces unités sont des conversions de millimètres, répertoriées dans le fichier `ly/paper-defaults-init.ly`. Pour plus de lisibilité, et bien que ce ne soit pas techniquement requis, nous vous conseillons d'ajouter `\mm` à votre code lorsque vous travaillez en millimètres.

Vous pouvez aussi définir les valeurs du bloc `\paper` à l'aide de fonctions Scheme. Voici l'équivalent de l'exemple précédent :

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

Voir aussi

Manuel de notation : Section 4.1.2 [Format du papier et adaptation automatique], page 572, Section 4.2.1 [Le bloc `\layout`], page 583, Section 3.2.2 [Titrages personnalisés], page 523.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

4.1.2 Format du papier et adaptation automatique

Format du papier

LilyPond génère par défaut, et en l'absence de mention explicite d'un format de papier particulier, un fichier imprimable au format A4. Vous pouvez cependant utiliser un autre format à l'aide des deux fonctions :

`set-default-paper-size`

```
#(set-default-paper-size "quarto")
```

qui se place en début de fichier, et

`set-paper-size`

```
\paper {
  #(set-paper-size "tabloid")
```

```
}
```

qui s'inscrit au sein d'un bloc `\paper`.

La seule restriction à l'utilisation isolée de la fonction `set-default-paper-size` est qu'elle doit intervenir avant le premier bloc `\paper`. `set-default-paper-size` fixe le format pour toutes les pages, alors que `set-paper-size` détermine le format des feuilles rattachées à un bloc `\paper` particulier. Ainsi, lorsque le bloc `\paper` se trouve en tête de fichier, le format du papier s'appliquera à toutes les pages ; si `\paper` apparaît dans un bloc `\book`, la taille ne s'appliquera qu'au *book* en question.

À l'intérieur d'un bloc `\paper`, la fonction `set-paper-size` doit intervenir avant toute autre variable. Les raisons à ceci sont abordées à la rubrique [Adaptation automatique au format], page 573.

Différents formats de papier sont définis dans le fichier `scm/paper.scm`. Bien que vous puissiez y ajouter votre propre format, sachez cependant que celui-ci est écrasé à chaque mise à jour de LilyPond. Les différents formats disponibles sont répertoriés à l'annexe Section A.5 [Formats de papier prédéfinis], page 713.

La commande suivante, inscrite dans votre fichier, vous permettra d'ajouter votre format personnalisé à ceux déjà connus, puis d'y faire appel à l'aide des fonctions `set-default-paper-size` et `set-paper-size` :

```
#(set! paper-alist
  (cons '("mon format" . (cons (* 15 in) (* 3 in))) paper-alist))

\paper {
  #(set-paper-size "mon format")
}
```

Les unités peuvent s'exprimer aussi bien en `in` (pouces), qu'en `cm` (centimètres) ou `mm` (millimètres).

Le fait d'ajouter l'argument `'landscape` à l'instruction stipulant le format du papier permet d'obtenir une présentation à l'italienne – ou paysage si vous préférez – et donc des lignes plus longues.

```
#(set-default-paper-size "a6" 'landscape)
```

L'inversion des dimensions du papier sans pour autant basculer la présentation – comme pour imprimer sur une carte postale ou créer un graphique destiné à inclusion – s'obtient en ajoutant '`landscape`' au nom du format de papier :

```
#(set-default-paper-size "a6landscape")
```

Lorsque la taille du papier comporte explicitement '`landscape`' ou '`portrait`', la présence d'un argument `'landscape` aura pour seul effet de modifier l'orientation de l'image et non les dimensions de la feuille.

Voir aussi

Manuel de notation : [Adaptation automatique au format], page 573, Section A.5 [Formats de papier prédéfinis], page 713.

Fichiers d'initialisation : `scm/paper.scm`.

Adaptation automatique au format

Toute modification du format de papier à l'aide des fonctions Scheme `set-default-paper-size` ou `set-paper-size`, que nous avons vues à la rubrique [Format du papier], page 572, se traduira automatiquement par l'ajustement d'un certain nombre de variables attachées au bloc `\paper` afin qu'elles soient en concordance avec le format spécifié. Vous pouvez annuler l'ajustement

automatique d'une variable particulière en redéfinissant sa valeur après avoir spécifié le format de papier utilisé. Notez bien que le simple fait d'affecter une valeur à `paper-height` ou `paper-width` ne déclenchera pas l'étalonnage automatique, bien que spécifier une largeur de papier *(*`paper-width`*)* peut influencer d'autres valeurs – mais c'est une autre histoire dont nous parlerons plus tard et qui n'a rien à voir avec la mise à l'échelle.

L'adaptation automatique affecte les dimensionnements verticaux `top-margin` et `bottom-margin` – voir Section 4.1.3 [Variables d'espacement vertical fixe], page 574, –, ainsi que les dimensionnements horizontaux `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent` et `short-indent` – voir Section 4.1.5 [Variables d'espacement horizontal], page 577.

Les valeurs par défaut de ces dimensionnements sont contenues dans le fichier `ly/paper-defaults-init.ly` et utilisent les variables internes `top-margin-default`, `bottom-margin-default`, etc. correspondant au format par défaut – papier A4 – pour lequel `paper-height` est à `297\mm` et `paper-width` à `210\mm`.

Voir aussi

Manuel de notation : Section 4.1.5 [Variables d'espacement horizontal], page 577, Section 4.1.3 [Variables d'espacement vertical fixe], page 574.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`, `scm/paper.scm`.

4.1.3 Variables d'espacement vertical fixe

> **Note :** Certains dimensionnements attachés au bloc `\paper` sont automatiquement ajustés selon le format du papier, ce qui peut conduire à un résultat inattendu – voir [Adaptation automatique au format], page 573.

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`paper-height`
: La hauteur de la feuille. Il s'agit par défaut de la dimension du papier utilisé. Notez bien que cette variable n'affectera pas l'ajustement automatique d'un certain nombre de dimensionnements verticaux.

`top-margin`
: La marge entre le bord supérieur de la feuille et la surface imprimable. Elle est fixée par défaut à `5\mm` et s'ajustera selon le format de papier.

`bottom-margin`
: La marge entre la surface imprimable et le bord inférieur de la feuille. Elle est fixée par défaut à `6\mm` et s'ajustera selon le format de papier.

`ragged-bottom`
: L'activation de cette variable permet de ne pas répartir verticalement les systèmes sur les pages hormis la dernière. La valeur par défaut est `#f`. Lorsque la partition ne comporte que deux ou trois systèmes par page, comme pour un conducteur d'orchestre, nous vous conseillons d'activer cette variable.

`ragged-last-bottom`
: La désactivation de cette variable permet de répartir verticalement les systèmes de la dernière page d'une partition. La valeur par défaut est `#t`. Nous vous conseillons, lorsque des pièces couvrent deux pages ou plus, de désactiver cette variable.

Notez bien que la variable `ragged-last-bottom` affecte aussi la dernière page de chacune des parties – créées à l'aide d'un bloc `\bookpart` – d'un même ouvrage.

Voir aussi

Manuel de notation : [Adaptation automatique au format], page 573.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Les titrages (contenus dans le bloc `\header{}`) sont considérés comme des systèmes à part entière ; ils seront donc affectés par `ragged-bottom` et `ragged-last-bottom`, qui éventuellement ajouteront de l'espace avant le premier système de la partition.

La définition explicite d'un format de papier annulera tout réglage des marges haute et basse.

4.1.4 Variables d'espacement vertical fluctuant

Il est souvent judicieux d'apporter un peu de flexibilité à l'espacement entre différents éléments (marges, titres, systèmes ou mouvements), en dilatation ou compression selon le cas. Un certain nombre de variables de type `\paper` répertoriées ci-dessous vous permettront d'affiner ces dimensionnements.

Gardez à l'esprit que les variables du bloc `\paper` dont nous parlons ici n'influencent en rien l'espacement des portées d'un même système. L'espacement au sein des systèmes est géré par des propriétés attachées à des objets graphiques (*grobs*) qui, elles, se définissent au niveau du bloc `\score` – voir à ce sujet Section 4.4.1 [Espacement vertical au sein d'un système], page 595.

Structure des variables d'espacement vertical fluctuant

Chacune de ces variables attachées au bloc `\paper` est constituée d'une liste associative *(alist)* à quatre *clés* :

- `basic-distance` *(distance de base)* – la grandeur d'espace par défaut, exprimée en hauteur de portée, séparant les *points de référence* de deux éléments, qui évite tout risque de collision en l'absence de dilatation ou compression. Le point de référence d'un titre ou d'un *markup* est son sommet, celui d'un système est le centre vertical du `StaffSymbol` le plus proche – même lorsqu'une ligne de « non-portée » viendrait à s'intercaler. Une `basic-distance` inférieure à `padding` ou `minimum-distance` sera sans effet, dans la mesure où l'espacement résultant ne saurait être inférieur à `padding` ou `minimum-distance`.
- `minimum-distance` *(distance minimale)* – l'espacement minimal, exprimé en hauteur de portée, entre les points de référence des deux éléments alors qu'il y a déjà un effet de compression. Une `minimum-distance` inférieure à la valeur du `padding` sera sans effet, dans la mesure où l'espacement résultant ne saurait être inférieur au `padding`.
- `padding` *(décalage)* – la grandeur minimale de « blanc » qui sépare deux éléments, exprimée en hauteur de portée. On peut le voir comme la hauteur minimale d'un rectangle vide qui devrait s'étendre sur toute la largeur des deux éléments.
- `stretchability` *(dilatation)* – le coefficient d'étirement de cet espace. Un coefficient nul permet de figer l'espacement, à moins qu'il n'en résulte des collisions. Un coefficient positif déterminera la propension d'un espacement à s'étirer, tout en tenant compte du coefficient affecté aux autres espacements. Par exemple, lorsque le coefficient de dilatation d'une dimension est double de celui d'une autre, elle pourra s'étirer deux fois plus que cette dernière. Il ne saurait être négatif. La valeur `+inf.0` provoque une `programming_error` (erreur de programmation) et est ignorée ; vous pouvez toutefois utiliser `1.0e7` pour obtenir une valeur proche de l'infini. Lorsque cette *clé* n'est pas définie, sa valeur est par défaut égale à `space`.

Notez bien que l'utilisateur ne peut définir une propension à la compression ; elle est en fait égale à (`basic-distance` − `minimum-distance`).

Lorsque l'impression n'est pas en pleine page – elle est donc *ragged bottom* pour les anglophones – l'élément `space` n'est pas étiré. Les hauteurs sur une telle page correspondront donc au maximum de

- `basic-distance`, plus
- `minimum-distance` et
- `padding`, augmenté de ce qu'il faut pour éviter les chevauchements.

Cependant, lorsque la partition fait plusieurs pages, la dernière page reprendra dans la mesure du possible l'espacement de la page précédente.

Les manières de modifier des listes associatives font l'objet d'un Section "chapitre spécifique" dans *Manuel de notation*. L'exemple suivant indique deux façons de modifier une liste associative. La première déclaration intervient sur une seule clé, alors que la deuxième redéfinit complètement la variable.

```
\paper {
  system-system-spacing.basic-distance = #8

  score-system-spacing =
    #'((padding . 1)
       (basic-distance . 12)
       (minimum-distance . 6)
       (stretchability . 12))
}
```

Liste des variables d'espacement fluctuant

Le nom des dimensionnements à hauteur variable sont de la forme *`avant-après`*`-spacing`, où *`avant`* et *`après`* représentent les éléments qui doivent être espacés. La distance s'établit entre les points de référence des deux éléments concernés (voir la rubrique précédente pour plus de précision). Notez bien que, dans les règles de nommage des variables qui suivent, le terme `markup` fait référence aussi bien à un *markup de titrage* (`bookTitleMarkup` ou `scoreTitleMarkup`) qu'à un *markup de haut niveau* (voir Section 3.1.5 [Structure de fichier], page 513). Toutes les distances sont exprimées en espace de portée.

Leurs valeurs par défaut sont inscrites dans le fichier `ly/paper-defaults-init.ly`.

`markup-system-spacing`
: détermine l'espacement entre un titre ou un *markup* de premier niveau, et le système qui le suit.

`score-markup-spacing`
: détermine l'espacement entre le dernier système et le titre ou *markup* de haut niveau qui le suit.

`score-system-spacing`
: détermine l'espacement entre le dernier système d'une partition et le premier système de la partition suivante, en l'absence de titrage ou *markup* qui les sépare.

`system-system-spacing`
: détermine l'espacement entre deux systèmes d'un même mouvement.

`markup-markup-spacing`
: détermine l'espacement entre deux titres ou *markups* de premier niveau.

`last-bottom-spacing`
: détermine la distance entre le dernier système ou le dernier *markup* de haut niveau, et le bas de la surface imprimable – autrement dit le haut de la marge basse.

`top-system-spacing`
: détermine l'espace entre le haut de la surface imprimable (le bas de la marge haute) et le milieu du premier système. Cette variable n'est effective qu'en l'absence de titre ou *markup* de premier niveau en haut de page.

`top-markup-spacing`
: détermine l'espace entre le haut de la surface imprimable (le bas de la marge haute) et le premier titre ou *markup* de premier niveau. Cette variable n'est effective qu'en l'absence de système en haut de page.

Voir aussi

Manuel de notation : Section 4.4.1 [Espacement vertical au sein d'un système], page 595.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.1.5 Variables d'espacement horizontal

> **Note :** Certains dimensionnements attachés au bloc `\paper` sont automatiquement ajustés selon le format du papier, ce qui peut conduire à un résultat inattendu – voir [Adaptation automatique au format], page 573.

Variables de marge et de largeur

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`paper-width`
: La largeur de la page. Elle correspond par défaut à la largeur du format de papier utilisé. Si `paper-width` n'a aucun effet en matière d'ajustement automatique, cette variable influe sur la variable `line-width`. Lorsque vous définissez à la fois les valeurs de `paper-width` et `line-width`, les valeurs de `left-margin` et `right-margin` seront recalculées. Voir aussi `check-consistency`.

`line-width`
: La longueur d'une ligne. Lorsque spécifié dans un bloc `\paper`, ceci définit l'étendue horizontale dont disposeront les lignes de portée d'un système non indenté. La valeur par défaut est égale à `paper-width`, auquel sont retranchés `left-margin` et `right-margin`. Lorsque vous définissez `line-width` sans modifier les valeurs de `left-margin` et `right-margin`, les marges seront alors recalculées de telle sorte que les systèmes soient centrés. Voir aussi `check-consistency`.

  La valeur de `line-width` peut aussi se spécifier individuellement au niveau de la partition, au sein d'un bloc `\layout`. Ceci permet de contrôler la longueur des lignes partition par partition. Si la longueur de ligne n'est pas spécifiée dans une partition particulière, elle sera valorisée à celle du `line-width` mentionné dans le bloc `\paper`. La détermination de `line-width` pour un `\score` particulier n'a aucun effet sur les marges. Les lignes d'une portée dont la longueur est déterminée par le `line-width` d'une partition seront alignées par la gauche sur la surface de papier telle que définie par le `line-width` du bloc `\paper`. Dès lors que les valeurs de `line-width` de la

partition et du papier sont égales, les lignes s'étendront de la marge gauche à la marge droite. En cas de `line-width` de la partition supérieur au `line-width` du papier, les lignes de portée déborderont dans la marge de droite.

`left-margin`
: La marge entre le bord gauche de la feuille et le début de chaque système. La valeur par défaut est de `10\mm` ; elle sera ajustée selon le format du papier. Lorsque vous définissez `line-width` et `right-margin` sans modifier la valeur de `left-margin`, cette dernière sera alors égale à `(paper-width − line-width − right-margin)`. Lorsque seule `line-width` est définie, les deux marges correspondent à `((paper-width − line-width) / 2)`, ce qui a pour effet de centrer les systèmes sur la page. Voir aussi `check-consistency`.

`right-margin`
: La marge entre le bord droit de la page et la fin des systèmes en pleine largeur (non *ragged*). La valeur par défaut est de `10\mm` et s'ajustera selon le format du papier. Lorsque vous définissez `line-width` et `left-margin`, sans modifier la valeur de `right-margin`, cette dernière sera alors égale à `(paper-width − line-width − left-margin)`. Lorsque seule `line-width` est définie, les deux marges correspondent à `((paper-width − line-width) / 2)`, ce qui a pour effet de centrer les systèmes sur la page. Voir aussi `check-consistency`.

`check-consistency`
: Lorsqu'elle est activée, cette variable vérifie que `left-margin`, `line-width` et `right-margin` sont en cohérence, et que l'addition de ces trois éléments ne dépassera pas la largeur du papier (`paper-width`). La valeur par défaut est `#t`. Dans le cas d'une incohérence, un message d'avertissement est émis et les trois variables – marges et longueur de ligne – rétablies à leur valeur par défaut (ajustées selon le format du papier). En cas de désactivation de cette variable (valorisation à `#f`, toute incohérence sera ignorée, et les systèmes pourront déborder de la page.

`ragged-right`
: Lorsque cette variable est activée, les systèmes ne s'étendront pas sur la longueur de la ligne, mais s'arrêteront à leur longueur normale. La valeur par défaut est `#f` mais, si la partition ne comporte qu'un seul système, elle passe à `#t`. Cette variable peut aussi se gérer au sein d'un bloc `\layout`.

`ragged-last`
: Lorsqu'elle est activée, cette variable permet de ne pas étendre le dernier système de façon à occuper toute la longueur de la ligne. La valeur par défaut est `#f`. Cette variable peut aussi se gérer au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : [Adaptation automatique au format], page 573.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Problèmes connus et avertissements

La définition explicite d'un format de papier annulera tout réglage des marges gauche et droite.

Variables spécifiques à l'impression recto-verso

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`two-sided`
: Cette variable permet de gérer efficacement les impressions recto-verso. Lorsqu'elle est activée, les réglages affectés à `inner-margin`, `outer-margin` ainsi que

`binding-offset` détermineront les différentes marges selon qu'il s'agit d'une page paire ou impaire. Cette variable s'applique en lieu et place de `left-margin` et `right-margin`. La valeur par défaut est `#f`.

`inner-margin`
: La marge que toutes les pages d'une partie ou de tout un ouvrage devront avoir du côté intérieur. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à vrai. La valeur par défaut est de `10\mm` et s'ajustera selon le format du papier.

`outer-margin`
: la marge que toutes les pages d'une partie ou de tout un ouvrage devront avoir du côté extérieur – opposé à la reliure. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à vrai. La valeur par défaut est de `20\mm` et s'ajustera selon le format du papier.

`binding-offset`
: La gouttière, ou marge de reliure, permet d'augmenter en conséquence la valeur de la marge intérieure `inner-margin` de telle sorte que rien ne soit masqué par la reliure. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à vrai. La valeur par défaut est de `0` et s'ajustera selon le format du papier.

Voir aussi

Manuel de notation : [Adaptation automatique au format], page 573.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Variables d'indentation et de décalage

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`horizontal-shift`
: Tous les systèmes, ainsi que les titres et séparateurs de systèmes, seront poussés d'autant vers la droite. La valeur par défaut est de `0.0\mm`.

`indent`
: Le niveau d'indentation du premier système d'une partition. La valeur par défaut est de `15\mm` en A4 et s'ajustera selon le format du papier. L'espace correspondant à `line-width` est réduit d'autant pour le premier système. Cette variable peut aussi se gérer partition par partition au sein d'un bloc `\layout`.

`short-indent`
: Le niveau d'indentation de tous les systèmes hormis le premier. La valeur par défaut est de `0` pour du A4, et s'ajustera selon le format du papier dès lors que vous lui aurez affecté une valeur. Bien entendu, l'espace spécifié par `line-width` sera réduit d'autant. Cette variable peut aussi se gérer partition par partition, au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : [Adaptation automatique au format], page 573.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.1.6 Autres variables du bloc \paper

Variables de gestion des sauts de ligne

`max-systems-per-page`
: Le nombre maximal de systèmes qu'une page pourra comporter. Cette variable n'est prise en compte, à ce jour, que par l'option `ly:optimal-breaking`, et n'est pas définie.

`min-systems-per-page`
: Le nombre minimal de systèmes qu'une page pourra comporter. Attention cependant aux risques de débordement s'il est trop important. Cette variable n'est prise en compte, à ce jour, que par l'option `ly:optimal-breaking`, et n'est pas définie.

`systems-per-page`
: Le nombre de systèmes que devrait comporter chaque page. Cette variable n'est à ce jour prise en charge que par l'algorithme `ly:optimal-breaking` et n'est pas définie par défaut.

`system-count`
: Le nombre de systèmes requis par la partition. Cette variable n'est pas définie par défaut. Cette variable peut se gérer au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : Section 4.3.1 [Sauts de ligne], page 588.

Variables de gestion des sauts de page

Les valeurs par défaut sont définies dans le fichier `ly/paper-defaults-init.ly`.

`page-breaking`
: L'algorithme de calcul des sauts de page à utiliser. Vous avez le choix entre `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-page-breaking`, `ly:one-line-breaking`, `ly:one-line-auto-height-breaking`, et `ly:optimal-breaking`. La valeur par défaut est `ly:optimal-breaking`.

`page-breaking-system-system-spacing`
: Cette variable permet de « tromper » l'algorithme de gestion des sauts de page quant à la valeur de `system-system-spacing`. Ainsi, lorsque `page-breaking-system-system-spacing.padding` a une valeur nettement supérieure à `system-system-spacing.padding`, l'algorithme en question aura tendance à disposer moins de systèmes sur une même page. Cette variable est par défaut non définie.

`page-count`
: Le nombre de pages que devra comporter la partition. Cette variable est par défaut non définie.

`page-spacing-weight`
: Lorsqu'il utilise l'algorithme `ly:optimal-breaking` pour gérer les sauts de page, LilyPond doit faire des compromis entre étirement horizontal et vertical afin de présenter un espacement acceptable. Cette variable définit l'importance relative des espacements entre la page (verticalité) et la ligne (horizontalité). Une valeur élevée privilégiera l'espacement au niveau de la page. La valeur par défaut est de `10`.

Les variables qui suivent ne sont effectives que lorsque l'algorithme `page-breaking` adopte la fonction `ly:page-turn-breaking`. Les sauts de page sont alors positionnés de sorte à minimiser le nombre de tournes. Dans la mesure où il faut tourner la feuille pour passer d'une page impaire à une page paire, sera privilégiée une répartition qui présente une dernière page impaire. Les

endroits où une tourne serait appropriée peuvent s'indiquer à l'aide d'un `\allowPageTurn` ou laissés à l'appréciation du `Page_turn_engraver` – voir [Optimisation des tournes], page 594.

Lorsqu'aucune option n'est satisfaisante pour placer judicieusement les tournes, LilyPond peut décider d'insérer une page blanche au milieu d'une partition ou entre deux partitions successives, voire même finir par une page paire. La valeur des trois variables qui suivent peut se voir augmentée de façon à diminuer ces risques.

Il s'agit ici de pénalité ; autrement dit, au plus la valeur est élevée, au moins l'action associée sera favorisée en regard des autres choix.

`blank-page-penalty`
: Pénalité pour apparition d'une page blanche en cours de partition. L'attribution d'une valeur élevée à `blank-page-penalty` alors qu'a été activé `ly:page-turn-breaking` forcera LilyPond à éviter de placer une page blanche au mileu de la partition, quitte à espacer d'autant plus la musique pour remplir cette page blanche et la suivante. La valeur par défaut est de `5`.

`blank-last-page-penalty`
: Pénalité pour fin de partition intervenant sur une page paire. L'attribution d'une valeur élevée à `blank-last-page-penalty` alors qu'a été activé `ly:page-turn-breaking` forcera LilyPond à éviter de terminer la partition sur une page paire, quitte à ajuster les espacements jusqu'à obtenir une page de plus ou une de moins. La valeur par défaut est de `0`.

`blank-after-score-page-penalty`
: Pénalité pour apparition d'une page blanche entre deux partitions. Sa valeur est par défaut inférieure à celle de `blank-page-penalty` ; nous préférons qu'une page blanche s'insère après la fin de la partition plutôt qu'au milieu. La valeur par défaut est de `2`.

Voir aussi

Manuel de notation : [Minimisation des sauts de page], page 593, [Optimisation des sauts de page], page 593, [Optimisation des tournes], page 594, [Présentation en ligne continue], page 594, [Présentation en page continue], page 594, [Présentation en rouleau], page 594, Section 4.3.2 [Sauts de page], page 592.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Variables de gestion des numéros de page

Les valeurs par défaut sont définies dans le fichier `ly/paper-defaults-init.ly`.

`auto-first-page-number`
: L'algorithme qui gère les sauts de page prend en compte le fait que le premier numéro de page soit pair ou impair. Lorsque cette fonctionnalité est activée, l'algorithme des sauts de page décidera de lui-même si le premier numéro sera pair ou impair, ce qui se traduira par un éventuel incrément de un. La valeur par défaut est `#f`.

`first-page-number`
: Le numéro de la première page. La valeur par défaut est de `#1`.

`print-first-page-number`
: Cette variable permet d'imprimer le numéro de page y compris sur la première. La valeur par défaut est `#f`.

`print-page-number`
: La désactivation de cette variable permet d'obtenir des pages non numérotées. La valeur par défaut est `#t`.

`page-number-type`
: Le type de chiffres à utiliser pour la numérotation : `roman-lower` (romains minuscules), `roman-upper` (romains majuscules) ou `arabic` (arabes). La valeur par défaut est `'arabic`.

Voir aussi

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Problèmes connus et avertissements

Les pages au numéro impair sont toujours à droite. Pour que la musique commence en page 1, le dos de la page de garde doit être vide de telle sorte que la page une se retrouve à droite.

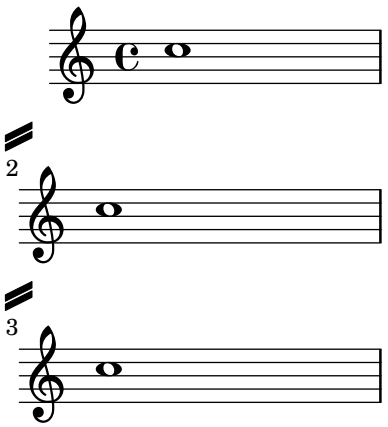
Variables supplémentaires d'entête et *markup*

`print-all-headers`
: Lorsque cette variable est activée, l'intégralité des champs d'entête sera imprimée pour chaque bloc `\score`, plutôt que les seuls champs `piece` et `opus`. Voir Section 3.2 [Titres et entêtes], page 515, pour les cas d'usage. La valeur par défaut est `#f`.

`system-separator-markup`
: Il s'agit en l'occurrence d'insérer un objet de type *markup* entre chaque système, comme on le voit dans nombre de partitions orchestrales. Cette variable n'est pas définie par défaut. La commande `\slashSeparator` – définie dans le fichier `ly/titling-init.ly` – fournit un *markup* relativement courant :

```
#(set-default-paper-size "a8")

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}
```



`footnote-separator-markup`
: Il s'agit d'un objet *markup* qui vient s'insérer au-dessus du texte de la note de bas de page. C'est, par défaut, une ligne horizontale centrée, définie dans `ly/paper-defaults-init.ly`.

Voir aussi

Fichiers d'initialisation : `ly/paper-defaults-init.ly`, `ly/titling-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

L'entête par défaut, formé d'une seule ligne, est constitué du numéro de page et du champ `instrument` contenu dans le bloc `\header`.

Variables de débogage

Les variables `debug-beam-scoring`, `debug-slur-scoring` et `debug-tie-scoring` permettent d'afficher des informations de débogage en matière de ligature et liaisons de phrasé ou de tenue. Voir Section "Débogage des algorithmes d'évaluation" dans *Guide du contributeur*, en anglais, pour des informations détaillées sur l'utilité de ces variables.

4.2 Mise en forme de la partition

Nous allons voir ici les options du bloc `\layout`. Elles sont plus particulièrement destinées à gérer la mise en forme de la partition.

4.2.1 Le bloc `\layout`

Alors que le bloc `\paper` définit le formatage des pages pour l'intégralité du document, le bloc `\layout` gère la mise en forme spécifique à la partition. La mise en forme de la musique peut concerner toutes les partitions d'un même ouvrage, auquel cas un bloc `\layout` indépendant se placera en tête de fichier. Dans le cas où la mise en forme concerne une partition en particulier, un bloc `\layout` se placera au sein du bloc `\score` en question. Sont susceptibles d'apparaître dans un bloc `\layout` :

- la fonction Scheme `layout-set-staff-size`,
- dans des blocs `\context`, les modifications apportées aux différents contextes, et
- les variables normalement attachées au bloc `\paper` qui affecteront la mise en forme de la partition.

La fonction `layout-set-staff-size` fait l'objet de la rubrique suivante, Section 4.2.2 [Définition de la taille de portée], page 585. La modification des contextes est abordée dans d'autres chapitres – voir Section 5.1.4 [Modification des greffons de contexte], page 636, et Section 5.1.5 [Modification des réglages par défaut d'un contexte], page 638.

Les variables du bloc `\paper` que l'on peut retrouver dans un bloc `\layout` sont :

- `line-width`, `ragged-right` et `ragged-last` (voir [Variables de marge et de largeur], page 577)
- `indent` et `short-indent` (voir [Variables d'indentation et de décalage], page 579)
- `system-count` (voir [Variables de gestion des sauts de ligne], page 580)

Voici un exemple de bloc `\layout` :

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
```

```
    \override StaffGrouper.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Il est tout à fait possible que plusieurs blocs `\layout` cohabitent en tant qu'expressions de niveau supérieur. Ceci se révèle particulièrement utile lorsque différents réglages sont stockés dans des fichiers séparés qui sont inclus au besoin. Lorsqu'un bloc `\layout` est évalué, une copie de la configuration du `\layout` actuel est réalisée en interne, augmentée des aménagements apportés. Bien qu'on puisse considérer que le contenu des différents blocs `\layout` se cumule, c'est la dernière adaptation qui aura préséance en cas de situation conflictuelle – cas typique d'une même propriété modifiée dans différents blocs.

Par exemple, placer le bloc suivant

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```

après celui de l'exemple précédent aura pour effet de cumuler les adaptations de `'padding` et `'color` pour l'objet `TextScript`, mais la dernière adaptation apportée à la propriété `'thickness` de `Glissando` remplace, ou masque, celle précédemment établie.

Les blocs `\layout` peuvent faire l'objet de variables, aux fins de les utiliser ultérieurement. Ceci requiert toutefois une attention particulière dans la mesure où cette manière de procéder n'est pas équivalente à une définition complète et globale.

Lorsque nous définissons la variable suivante,

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
  }
}
```

qui contient une configuration de `\layout` avec l'adaptation `NoteHead.font-size`, cette combinaison n'est pas enregistrée en tant que configuration courante. Notez bien que la « configuration courante » est lue lorsque la variable est définie, non lorsqu'elle est utilisée ; par voie de conséquence, le contenu de la variable dépend de l'endroit où elle se trouve dans le code source.

Notre variable peut alors être utilisée au sein d'un autre bloc `\layout`, comme par exemple :

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

Un bloc `\layout` qui contient une variable comme ci-dessus ne recopie pas la configuration actuelle ; il utilise en fait le contenu de `layoutVariable` en tant que configuration de base

pour les adaptations ultérieures, en conséquence de quoi toute modification intervenant entre la définition et l'utilisation de la variable sera perdue.

Si `layoutVariable` est définie, ou rapatriée par un `\indude`, juste avant d'être utilisée, son contenu devient la configuration actuelle augmentée des adaptations que la variable contient. Considérant l'exemple d'utilisation de `layoutVariable` ci-dessus, le bloc `\layout` final contiendra donc :

```
TextScript.padding = #1
TextScript.color = #magenta
Glissando.thickness = #1.5
NoteHead.font-size = #4
NoteHead.color = #red
```

ainsi que les adaptations de `indent` et `StaffGrouper`.

Cependant, si la variable avait été définie bien avant le premier bloc `\layout`, la configuration actuelle ne contiendrait que

```
NoteHead.font-size= #4 % (écrit dans la définition de la variable)
NoteHead.color = #red % (ajouté après l'utilisation de la variable)
```

Une gestion attentive des variables de `\layout` se révèle être un outil précieux dans la mise en forme des sources et le retour à une configuration donnée.

Voir aussi

Manuel de notation : Section 5.1.5 [Modification des réglages par défaut d'un contexte], page 638.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.2.2 Définition de la taille de portée

La **taille de portée** *(staff size)* est fixée par défaut à 20 points, ce qui correspond à 7,03 cm – 1 point équivaut à 100/7227 pouce, soit 2 540/7 227 mm. Il existe trois manières de la modifier :

1. La taille des portées peut se définir globalement pour toutes les partitions d'un même fichier, ou plus précisément d'un bloc `\book`, à l'aide de `set-global-staff-size`.

   ```
   #(set-global-staff-size 14)
   ```

   Ceci définit donc la hauteur des portées à 14 points (4,92 mm) par défaut ; toutes les fontes seront ajustées en conséquence.

2. La taille d'une partition particulière au sein d'un ouvrage se définit à l'aide d'un `layout-set-staff-size` placé dans le bloc `\layout` approprié :

   ```
   \score{
     ...
     \layout{
       #(layout-set-staff-size 14)
     }
   }
   ```

3. Pour l'affectation d'une taille particulière à l'une des portées d'un système, LilyPond dispose de la commande `\magnifyStaff`. Par exemple, les partitions traditionnelles de musique de chambre avec piano présentaient souvent des portées de piano de 7 mm alors que les autres portées étaient gravées à une hauteur de cinq septièmes (s'il y avait assez de place) ou trois cinquièmes (en cas de présentation resserrée) de cette hauteur. Une proportion de 5/7 se libelle ainsi :

   ```
   \score {
     <<
   ```

```
      \new Staff \with {
        \magnifyStaff #5/7
      } { ... }
      \new PianoStaff { ... }
    >>
  }
```

Si la valeur de `fontSize` à utiliser est connue, la forme suivante peut s'employer :

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #(magstep -3)
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Mieux vaut éviter de réduire l'épaisseur des lignes si l'on veut que la partition s'approche au plus près des canons de la gravure traditionnelle.

Relation automatique entre fonte et taille

La fonte Ementaler fournit le jeu de symboles musicaux *Feta* dans huit tailles différentes. Chaque fonte correspond à une hauteur particulière de portée ; les petites tailles comportent des symboles plus épais pour être cohérent avec l'épaisseur relativement plus importante des lignes de la portée. Le tableau suivant répertorie les différentes tailles de police.

nom de la fonte	**hauteur de portée (pt)**	**hauteur de portée (mm)**	**utilisation**
feta11	11,22	3,9	format de poche
feta13	12,60	4,4	
feta14	14,14	5,0	
feta16	15,87	5,6	
feta18	17,82	6,3	carnet de chant
feta20	20	7,0	partition standard
feta23	22,45	7,9	
feta26	25,2	8,9	

Voir aussi

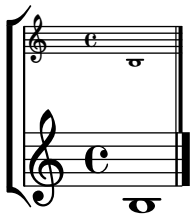
Manuel de notation : [Indication de la taille de fonte musicale], page 232, Section A.8 [La fonte Emmentaler], page 718.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Lorsque `\magnifyStaff` est utilisé au sein d'un `StaffGroup`, les objets `BarLine` ne s'alignent plus, en raison des modifications apportées à leurs propriétés `thick-thickness`, `hair-thickness` et `kern`.

```
\new StaffGroup
  <<
    \new Staff \with { \magnifyStaff #1/2 } { b1 \bar "|." }
    \new Staff { b }
  >>
```



Vous pouvez opter pour annuler le redimensionnement des objets `BarLine`, l'imiter pour les autres portées, ou bien encore appliquer une valeur intermédiaire à toutes les portées.

```
#(define bar-line-props
  '((BarLine thick-thickness)
    (BarLine hair-thickness)
    (BarLine kern)))

mus = { b1 \bar "|."}

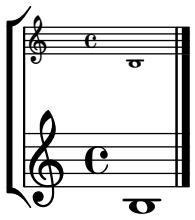
\markup "Annulation de \\magnifyStaff pour les barres de mesure :"
\new StaffGroup
  <<
    \new Staff
      \with {
        \magnifyStaff
        #1/2 #(revert-props 'magnifyStaff 0 bar-line-props)
      }
      \mus
    \new Staff
      \mus
  >>

\markup "Imitation de \\magnifyStaff dans les autres portées :"
\new StaffGroup
  <<
    \new Staff
      \with { \magnifyStaff #1/2 }
      \mus
    \new Staff
      \with {
        #(scale-props 'magnifyStaff 1/2 #t bar-line-props)
      }
      \mus
  >>
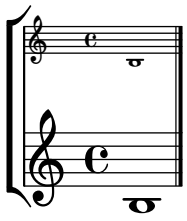
\markup "Application à tous de valeurs intermédiaires :"
\new StaffGroup
  <<
    \new Staff
      \with {
        \magnifyStaff #1/2
        #(scale-props 'magnifyStaff 3/2 #t bar-line-props)
      }
      \mus
    \new Staff
      \with {
        #(scale-props 'magnifyStaff 3/4 #t bar-line-props)
```

```
    }
    \mus
>>
```

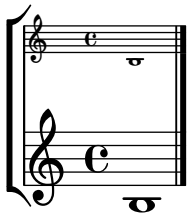
Annulation de \magnifyStaff pour les barres de mesure :



Imitation de \magnifyStaff dans les autres portées :



Application à tous de valeurs intermédiaires :
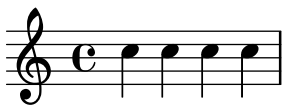


4.3 Sauts
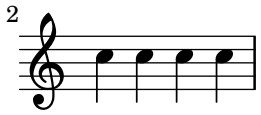
4.3.1 Sauts de ligne

Les sauts de ligne sont normalement gérés de façon automatique. Ils interviennent de telle sorte qu'une ligne ne soit ni trop resserrée, ni trop aérée, et que des lignes consécutives aient à peu près la même densité.

Vous pouvez cependant insérer l'instruction `\break` à l'endroit où vous le jugez utile pour « forcer » le passage à la ligne suivante :
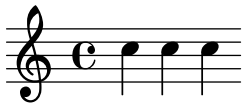
```
\relative c'' {
  c4 c c c | \break
  c4 c c c |
}
```
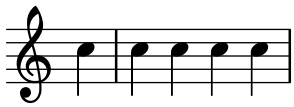




Par défaut, un saut de ligne ne saurait intervenir au beau milieu d'une mesure ; LilyPond vous le signalera par un message lors de la compilation du fichier. Si d'aventure vous voulez forcer un saut de ligne en l'absence de barre de mesure, vous devrez auparavant insérer une barre invisible – à l'aide de `\bar ""`.
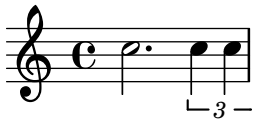
```
\relative c'' {
  c4 c c
  \bar ""
  \break
  c |
  c4 c c c |
}
```
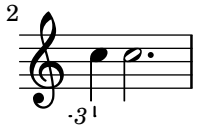




LilyPond ignorera un `\break` placé sur une barre à la fin d'une mesure dès lors que la précédente avait une note en suspend – c'est typiquement le cas lorsqu'un n-olet est à cheval sur deux mesures. L'instruction `\break` sera toutefois opérationnelle si vous avez auparavant désactivé le `Forbid_line_break_engraver` du contexte `Voice` concerné. Notez bien qu'en pareil cas, les sauts de ligne forcés doivent être saisis au sein d'une expression polyphonique :

```
\new Voice \with {
  \remove "Forbid_line_break_engraver"
} \relative {
  <<
    { c''2. \tuplet 3/2 { c4 c c } c2. | }
    { s1 | \break s1 | }
  >>
}
```

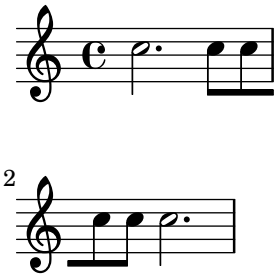




Selon le même principe, un saut de ligne ne peut intervenir alors qu'une ligature s'étend sur deux mesures consécutives. Il faut en ce cas là introduire la dérogation `\override Beam.breakable = ##t`.

```
\relative c'' {
  \override Beam.breakable = ##t
  c2. c8[ c | \break
```

```
  c8 c] c2. |
}
```



L'instruction opposée, `\noBreak`, interdira toute tentative de saut de ligne à la fin de la mesure où elle est explicitée.

Au sein même d'une pièce, les sauts de lignes automatiques sont inhibés dans la musique encadrée par les commandes `\autoLineBreaksOff` et `\autoLineBreaksOn`. Dans le cas où les sauts de page automatiques devraient eux aussi être inhibés, ce sont les commandes `\autoBreaksOff` et `\autoBreaksOn` qu'il faudrait utiliser. Les sauts de ligne ou de page manuels ne sont pas affectés par ces commandes. Notez bien que la désactivation du positionnement automatique des sauts des ligne peut avoir pour effet un débordement de la musique dans la marge de droite si tout ne peut être contenu sur une ligne.

Des sauts de ligne automatiques peuvent cependant être autorisés en un point particulier à l'aide d'un `\once \autoLineBreaksOn` sur une barre de mesure. Ceci ne concerne pas les sauts de page. Ceci indique qu'un saut de ligne peut intervenir à cet endroit précis, mais ne le force en aucun cas.

LilyPond dispose de deux variables de base pour influencer l'espacement au niveau des lignes. Toutes deux se définissent dans un bloc `\layout`, `indent` réglant l'indentation de la première ligne, et `line-width` la longueur des lignes.

L'activation du commutateur `ragged-right` au sein du bloc `\layout` aura pour effet de terminer les systèmes là où ils prendraient fin normalement plutôt que de les étirer sur toute la longueur de la ligne. Ceci est particulièrement utile pour de petits fragments ou pour vérifier la densité induite par l'espacement naturel.

Le commutateur `ragged-last` est équivalent à `ragged-right`, à ceci près qu'il n'affecte que la dernière ligne de la pièce.

```
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

L'utilisation conjointe de `\break` et de blancs dans une section `\repeat` vous permettra de positionner des sauts de ligne à intervalle régulier. Par exemple, les 28 mesures de ce qui suit, si l'on est à 4/4, seront coupées toutes les quatre mesures, pas ailleurs :

```
<<
  \repeat unfold 7 {
    s1 \noBreak s1 \noBreak
    s1 \noBreak s1 \break
  }
  { et ici la musique... }
>>
```

Commandes prédéfinies

`\break`, `\noBreak`, `\autoBreaksOff`, `\autoBreaksOn`, `\autoLineBreaksOff`, `\autoLineBreaksOn`.

Morceaux choisis

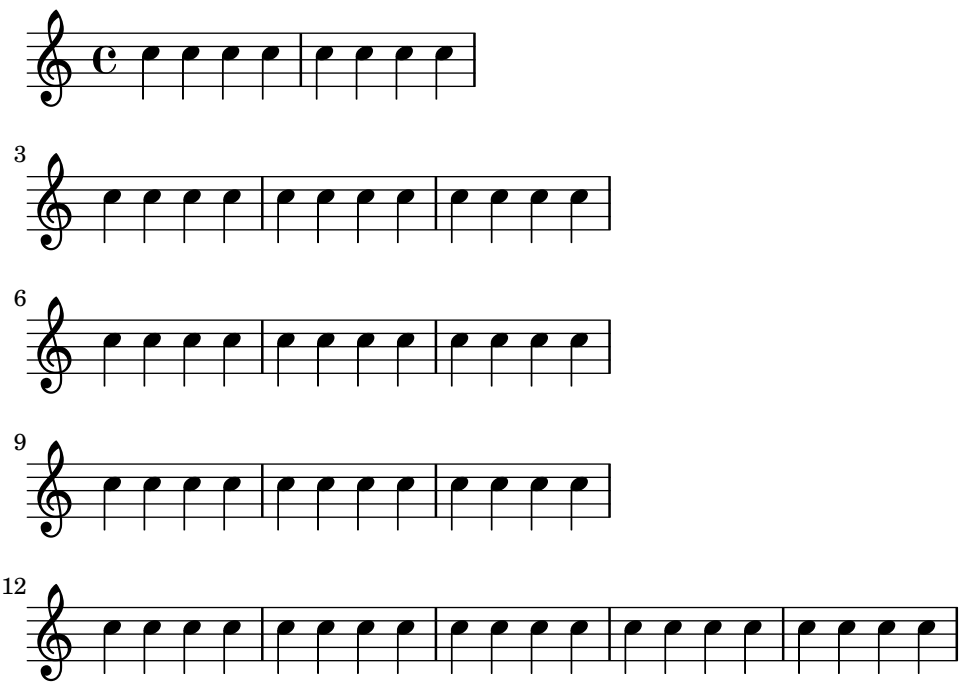
Recours à une voix supplémentaire pour gérer les sauts

Il est souvent plus pratique de séparer ce qui est purement musical et les informations concernant les sauts de ligne ou de page, en créant une voix supplémentaire dédiée. Cette voix spécifique ne contiendra que des blancs – des silences invisibles `\skip` –, des `\break`, des `\pageBreak` et autres informations concernant les ruptures.

Cette manière de procéder est tout à fait indiquée lorsque vous ajustez les `line-break-system-details` et autres propriétés fort intéressantes de `NonMusicalPaperColumnGrob`.

```
music = \relative c'' { c4 c c c }

\score {
  \new Staff <<
    \new Voice {
      s1 * 2 \break
      s1 * 3 \break
      s1 * 6 \break
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 6 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}
```



Voir aussi

Manuel de notation : Section 4.2.1 [Le bloc `\layout`], page 583, [Variables de gestion des sauts de ligne], page 580.

Référence des propriétés internes : Section "LineBreakEvent" dans *Référence des propriétés internes*.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Les commandes `\autoLineBreaksOff` et `\autoBreaksOff` doivent impérativement se placer après le début de la musique pour éviter tout message d'erreur.

4.3.2 Sauts de page

Cette section présente les différentes méthodes de gestion des sauts de page, ainsi que les moyens de les modifier.

Saut de page manuel

La gestion automatique des sauts de page se contrôle à l'aide des commandes `\pageBreak` et `\noPageBreak`. Ces commandes fonctionnent de manière analogue à `\break` et `\noBreak` pour les sauts de ligne et se placent donc au moment d'une barre de mesure. Elles permettent de forcer, ou d'interdire, un saut de page à la prochaine barre de mesure. Comme on peut s'y attendre, `\pageBreak` force aussi le saut de ligne.

Les commandes `\pageBreak` et `\noPageBreak` peuvent se trouver à des niveaux supérieurs, entre deux partitions ou *markups* de premier rang.

Au sein même d'une pièce, les sauts de page automatiques sont inhibés dans la musique encadrée par les commandes `\autoPageBreaksOff` et `\autoPageBreaksOn`. Les sauts de page manuels ne sont pas affectés par ces commandes.

Tout comme `ragged-right` et `ragged-last` qui permettent de gérer la répartition horizontale, LilyPond dispose de commutateurs équivalents au niveau de la verticalité. `ragged-bottom`, une fois activé, empèchera les systmèes de se répartir sur la page. Quant à `ragged-last-bottom` (valorisé à `#t` par défaut), il laissera un espace vide en dernière page, y compris pour chaque `\bookpart`. Pour de plus amples détails, reportez-vous à Section 4.1.3 [Variables d'espacement vertical fixe], page 574.

Les sauts de page sont générés par la fonction `page-breaking`. LilyPond dispose de plusieurs algorithmes en la matière : `ly:optimal-breaking`, `ly:page-turn-breaking` et `ly:minimal-breaking`. C'est `ly:optimal-breaking` qui est activé par défaut, mais rien ne vous empêche d'en changer, par l'intermédiaire du bloc `\paper` :

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

Lorsqu'un ouvrage contient plusieurs partitions et un certain nombre de pages, la gestion des sauts de page finit par devenir très gourmande, tant au niveau du processeur que de la mémoire. Vous pouvez cependant alléger la charge en recourant à des blocs `\bookpart` afin de sectionner l'ouvrage que vous traitez ; les sauts de page seront alors gérés individuellement au niveau de chacune des parties. Par ailleurs, cela vous autorisera une gestion différente selon les sections.

```
\bookpart {
  \header {
    subtitle = "Préface"
  }
  \paper {
```

```
      %% Pour une partie constituée principalement de texte
      %% ly:minimal-breaking est plus judicieux.
      page-breaking = #ly:minimal-breaking
    }
    \markup { ... }
    ...
  }
  \bookpart {
    %% Cette partie étant purement musicale,
    %% retour au style par défaut (optimal-breaking).
    \header {
      subtitle = "Premier mouvement"
    }
    \score { ... }
    ...
  }
```

Commandes prédéfinies

`\pageBreak`, `\noPageBreak`, `\autoPageBreaksOn`, `\autoPageBreaksOff`.

Voir aussi

Manuel de notation : [Variables de gestion des sauts de page], page 580.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Le préfixe `\once` est ineffectif en ce qui concerne les commandes `\autoPageBreaksOn` et `\autoPageBreaksOff`. Si le positionnement automatique des sauts de page est désactivé et qu'il est réactivé pour permettre un saut de page, il doit le rester pendant quelques mesures (le nombre précis de mesures dépendant de la pièce) avant d'être à nouveau désactivé, autrement l'opportunité de passer à la page suivante ne sera pas saisie.

Optimisation des sauts de page

LilyPond, pour déterminer où placer un saut de page, utilise par défaut la fonction `ly:optimal-breaking`. Celle-ci tend à trouver une rupture qui évite d'obtenir à la fois une page trop dense ou exagérément aérée. Contrairement à la fonction `ly:page-turn-breaking`, elle n'a aucune notion de ce qu'est une « tourne ».

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Minimisation des sauts de page

La fonction `ly:minimal-breaking` est celle qui réalise le moins de calculs pour positionner les sauts de page. Elle mettra le plus de systèmes possible sur une page avant de passer à la suivante. On peut donc la préférer lorsque la partition s'étend sur beaucoup de pages ou lorsque les autres fonctions de gestion des sauts de page ralentissent nettement le traitement, sont trop gourmandes en mémoire ou qu'il y a beaucoup de texte. Il suffit de la mentionner au sein du bloc `\paper` :

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Présentation en page continue

La fonction `ly:one-page-breaking` constitue un algorithme de calcul des sauts de page particulier en ceci que la hauteur de page sera automatiquement ajustée à la longueur de la partition, de telle sorte que toute la musique tienne sur une seule page. La variable `paper-height` du bloc `\paper` est ignorée, mais les autres réglages restent disponibles. En particulier, l'espacement entre le dernier système (ou un *markup* de premier niveau) et le pied de page est réglable à l'aide de la variable `last-bottom-spacing` du bloc `\paper`. La largeur de page n'est, par défaut, pas modifiée ; ceci peut s'ajuster à l'aide de la variable `paper-width` du bloc `\paper`.

Problèmes connus et avertissements

`ly:one-page-breaking` est à ce jour incompatible avec l'utilisation de `\bookpart`.

Présentation en ligne continue

La fonction `ly:one-line-breaking` constitue un algorithme de calcul des sauts de page particulier en ceci que chaque partition fait l'objet d'une page unique, d'une seule ligne. Cette fonctionnalité s'affranchit de l'impression des titres et marges ; seule la partition est affichée.

La largeur de page est ajustée de telle sorte que la pièce la plus longue tienne sur une seule ligne. En particulier, les variables `paper-width`, `line-width` et `indent` du bloc `\paper` seront ignorées ; les `left-margin` et `right-margin` seront honorées. La hauteur de page ne sera pas modifiée.

Présentation en rouleau

La fonction `ly:one-line-auto-height-breaking` opère comme `ly:one-line-breaking`, à ceci près que la hauteur de page s'adapte automatiquement à la hauteur de la musique. Dans les faits, la variable `paper-height` du bloc `\paper` s'ajuste de façon à englober la hauteur de la partition la plus étendue, plus les `top-margin` et `bottom-margin`.

L'affectation d'une valeur à `top-system-spacing` influencera le positionnement vertical de la musique. Sa désactivation – valorisation à `##f` au sein du bloc `\paper` – aura pour effet de simplement placer la musique entre les marges supérieure et inférieure.

Optimisation des tournes

Aboutir à une configuration des sauts de page de telle sorte que les pages de droite se terminent toujours par un silence devient souvent une nécessité. En effet, l'exécutant pourra alors tourner la page sans risquer de manquer des notes. La fonction `ly:page-turn-breaking` tend à trouver une rupture qui évite d'obtenir à la fois une page trop dense ou exagérément aérée, tout en tenant compte du fait qu'une tourne ne saurait intervenir qu'à certains endroits.

L'utilisation de cette fonction se fait en deux étapes. Il vous faut tout d'abord l'activer au sein du bloc `\paper` comme indiqué à la rubrique Section 4.3.2 [Sauts de page], page 592. Vous devrez, dans un deuxième temps, informer la fonction des endroits où les sauts de page sont permis.

Cette deuxième étape se réalise de deux manières différentes. Vous pouvez spécifier manuellement chaque tourne potentielle en insérant un `\allowPageTurn` à l'endroit approprié de votre fichier source.

Toutefois, cette option peut vite se révéler fastidieuse selon l'ampleur de l'œuvre. Vous pouvez alors recourir au `Page_turn_engraver` que vous mentionnerez dans un contexte de voix ou de portée. Ce graveur de tournes recherchera dans le contexte en question les passages sans note. Notez bien qu'il ne recherche pas des silences, mais l'absence de notes ; autrement dit, il ne

restera pas inactif dans le cadre d'une portée polyphonique dont l'une des parties contiendrait des silences. Lorsqu'il rencontre un fragment suffisamment long ne contenant aucune note, il insère un `\allowPageTurn` à la barre terminant ce fragment, à moins qu'il ne rencontre en chemin une « barre spéciale » – telle une double barre – auquel cas il y déposera le `\allowPageTurn`.

Le `Page_turn_engraver` examine la propriété de contexte `minimumPageTurnLength` pour déterminer quelle doit être la longueur d'un fragment sans note avant une tourne. La valeur par défaut de `minimumPageTurnLength` est `(ly:make-moment 1 1)`, soit une ronde, et s'ajuste de la manière suivante :

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % une tourne peut se placer ici
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
  R1 | % il ne peut pas y avoir de tourne ici
  a4 b r2 |
  R1*2 | % une tourne peut se placer ici
  a1
}
```

Le `Page_turn_engraver` tient compte des reprises. C'est pourquoi il ne permettra une tourne que dans le cas où il y aura suffisamment de temps au début et à la fin de la reprise pour que l'exécutant puisse aisément revenir à la page précédente. Le `Page_turn_engraver` est même capable d'interdire un tourne dans le cas d'une reprise de courte durée, ajustable au travers de la propriété de contexte `minimumRepeatLengthForPageTurn`.

Les commandes de tourne – `\pageTurn`, `\noPageTurn` et `\allowPageTurn` – peuvent s'utiliser à des niveaux supérieurs, entre des blocs `\score` ou des *markups* de haut niveau.

Commandes prédéfinies

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Voir aussi

Manuel de notation : [Variables de gestion des sauts de ligne], page 580.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Une partition ne devrait contenir qu'une seule instance du `Page_turn_engraver`, au risque de les voir se contredire.

4.4 Espacement vertical

L'espacement vertical dépend de trois éléments : la surface disponible – format de papier et marges –, l'espace qui doit séparer les systèmes, et l'espace qui sépare les portées d'un même système.

4.4.1 Espacement vertical au sein d'un système

LilyPond dispose de trois différents mécanismes permettant de contrôler l'espacement au sein d'un système selon trois catégories :

- *portées isolées*,
- *portées regroupées* (portées d'un même groupe, telles celles d'un `ChoirStaff`, etc.), et
- *lignes de non-portée* (`Lyrics`, `ChordNames`, etc.).

La hauteur de chaque système se détermine en deux phases. Les portées sont tout d'abord espacées selon la surface disponible. Puis les lignes autres que des portées, comme les paroles ou les accords, sont réparties entre les portées.

Les paragraphes qui suivent traitent exclusivement de la manière de gérer l'espacement entre les lignes d'un système – portée musicale ou non. Pour ce qui a trait aux espacements entre les systèmes, mouvements, annotations et marge, ils sont contrôlés par des variables attachées au bloc `\paper` et font l'objet du chapitre Section 4.1.4 [Variables d'espacement vertical fluctuant], page 575.

Propriétés d'espacement au sein d'un système

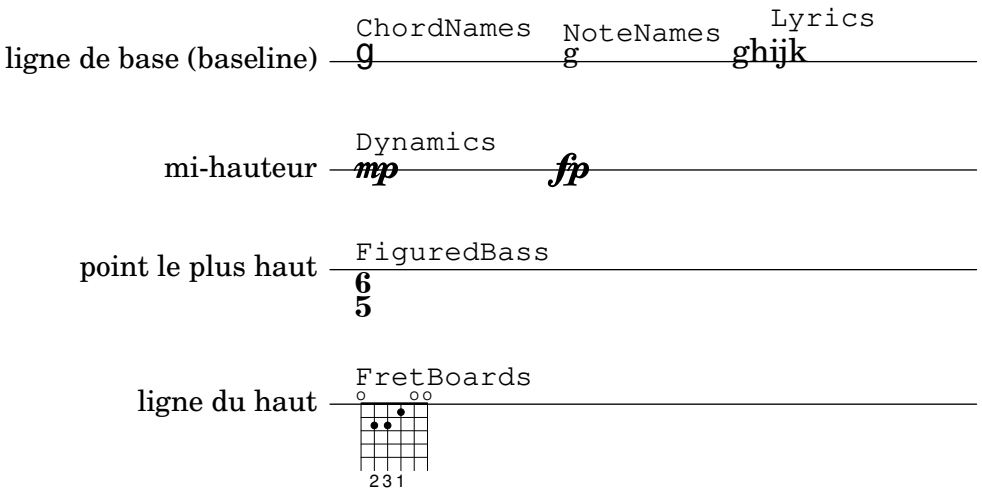
L'espacement entre les portées est géré par deux jeux de propriétés d'objet graphique (*grob*). Le premier, associé à l'objet graphique `VerticalAxisGroup`, est créé pour toute ligne de portée ou de non-portée. Le second, associé à l'objet graphique `StaffGrouper`, doit être explicitement créé pour un regroupement de portées particulier. Les propriétés qui leur sont attachées sont abordées en fin de section.

Le nom de ces propriétés, sauf `staff-affinity`, suit le schéma *`item1`*`-`*`item2`*`-spacing` – *`item1`* et *`item2`* étant les éléments à espacer. Notez bien que *`item2`* n'est pas forcément placé au-dessous : c'est le cas pour la propriété `nonstaff-relatedstaff-spacing` qui spécifie l'espacement d'une ligne de non-portée alors que sa `staff-affinity` a été déterminée à `UP`.

Toutes ces distances sont mesurées entre les points de référence respectifs des éléments considérés. Le *point de référence* d'une portée est le centre vertical du `StaffSymbol` – la ligne médiane si `line-count` est impair, l'interligne médian si `line-count` est pair. Quant aux lignes rattachées à des portées – lignes de non-portée – le tableau suivant présente le *point de référence* pour chacune d'elles :

Ligne de non-portée	**Point de référence**
`ChordNames`	ligne de base
`NoteNames`	ligne de base
`Lyrics`	ligne de base
`Dynamics`	mi-hauteur du « m »
`FiguredBass`	point le plus haut
`FretBoards`	ligne supérieure

En voici une représentation graphique :



Hormis `staff-affinity` – propriété attachée au *grob* `VerticalAxisGroup` –, chacune de ces propriétés est enregistrée sous la forme d'une liste associative dont la structure est identique à

celle des variables du bloc `\paper` que nous avons examinées au chapitre Section 4.1.4 [Variables d'espacement vertical fluctuant], page 575. Les particularités en matière de modification d'une liste associative font l'objet d'un Section "chapitre particulier" dans *Manuel de notation*. Les propriétés des objets graphiques se règlent avec un `\override` mentionné dans un bloc `\score` ou `\layout`, pas dans le bloc `\paper`.

L'exemple suivant illustre deux façons de modifier une liste associative. La première déclaration n'agit que sur une seule clé, alors que la seconde redéfinit la propriété dans son intégralité.

```
\new Staff \with {
  \override VerticalAxisGroup
              .default-staff-staff-spacing.basic-distance = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing =
    #'(('basic-distance  . 10)
       (minimum-distance . 9)
       (padding . 1)
       (stretchability . 10))
} { ... }
```

La modification d'un espacement au niveau global se mentionne au sein du bloc `\layout` :

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup
      .default-staff-staff-spacing.basic-distance = #10
  }
}
```

Les réglages concernant les propriétés d'espacement vertical des objets graphiques sont répertoriées aux chapitres Section "VerticalAxisGroup" dans *Référence des propriétés internes* et Section "StaffGrouper" dans *Référence des propriétés internes*. Les propriétés relatives aux lignes de non-portée sont répertoriées selon la définition de leur contexte dans la Section "Contexts" dans *Référence des propriétés internes*.

Propriétés de l'objet `VerticalAxisGroup`

Les propriétés de l'objet `VerticalAxisGroup` s'ajustent à l'aide d'un `\override` au niveau d'un contexte `Staff` (ou son équivalent).

`staff-staff-spacing`

Il s'agit de la distance entre la portée en cours et la portée qui suit au sein du même regroupement, qu'il y ait ou des lignes de non-portée (`Lyrics` ou autre entre les deux. Cette propriété ne s'applique pas à la dernière portée d'un système.

En tout état de cause, la fonction Scheme `staff-staff-spacing` d'un `VerticalAxisGroup` affectera les propriétés du `StaffGrouper` si la portée est incluse dans un regroupement ; elle s'appliquera au `default-staff-staff-spacing` en l'absence de regroupement. Les portées peuvent donc s'aligner différemment selon qu'elles sont ou non regroupées. Pour obtenir le même espacement sans tenir compte des éventuels regroupements, cette fonction peut faire place à une complète redéfinition des espacements fluctuants à l'aide de règles dérogatoires comme vu précédemment. Au cas où seulement certaines `+`valeurs font l'objet

d'une dérogation, les valeurs non mentionnées +seront ajustées sur celles de `default-staff-staff-spacing` (si +tant est qu'elles y soient définies).

`default-staff-staff-spacing`
: Il s'agit de la distance qui s'appliquera par défaut aux portées isolées, à moins que `staff-staff-spacing` n'ait été redéfini explicitement par un `\override`.

`staff-affinity`
: Il s'agit de la direction – `UP`, `DOWN` ou `CENTER` – que prendra une ligne de non-portée pour aller s'accoler aux portées adjacentes. Si vous lui attribuez `CENTER`, cette ligne de non-portée ira se placer à équidistance entre les portées qui l'encadrent, tout en tenant compte des éventuels risques de collision et autres contraintes d'espacement. Des lignes de non-portée adjacentes devraient avoir une `staff-affinity` allant de haut en bas – autrement dit, pas de `UP` après un `DOWN`. Une ligne de non-portée en dessous d'un système devrait avoir sa `staff-affinity` définie à `UP`. De la même manière, lorsque cette ligne surplombe un système, sa `staff-affinity` devrait être définie à `DOWN`. Prenez garde à la valeur que vous affectez à `staff-affinity` : si vous affectez la valeur `#f` à une ligne de non-portée, cette ligne sera considérée comme étant une portée ; à l'inverse, utiliser la propriété `staff-affinity` pour une portée lui fera perdre cette qualité.

`nonstaff-relatedstaff-spacing`
: Il s'agit de la distance entre la ligne de non-portée en cours et la portée la plus proche selon la `staff-affinity`, à la double condition qu'il n'y ait pas déjà une autre ligne de non-portée et que la valeur de `staff-affinity` soit `UP` ou `DOWN`. Lorsque la valeur de `staff-affinity` est égale à `CENTER`, la valeur de `nonstaff-relatedstaff-spacing` servira à centrer la ligne de non-portée entre les deux portées adjacentes même si une autre non-portée est présente (quelque soit le côté). Le positionnement d'une ligne de non-portée dépend donc à la fois des portées qui l'entourent tout comme des autres lignes de non-portée adjacentes. L'affectation d'une faible valeur à la propriété `stretchability` de l'un de ces types d'espacement l'avantagera ; lui affecter une valeur élevée aura pour conséquence de diminuer l'influence de l'espacement considéré.

`nonstaff-nonstaff-spacing`
: Il s'agit de la distance entre deux lignes de non-portée selon l'orientation définie par `staff-affinity` et dès lors qu'elles ont la même orientation. Bien entendu, ceci ne peut concerner que les valeurs `UP` et `DOWN` de `staff-affinity`.

`nonstaff-unrelatedstaff-spacing`
: Il s'agit de la distance entre une ligne de non-portée et la portée à l'opposé de l'orientation adoptée, à la double condition qu'il n'y ait pas déjà une autre ligne de non-portée et que la valeur de `staff-affinity` soit `UP` ou `DOWN`. Cette propriété trouve toute sa légitimité pour décaler une ligne de `Lyrics` de la portée à laquelle elle ne correspond pas.

Propriétés de l'objet `StaffGrouper`

Les propriétés de l'objet `StaffGrouper` s'ajustent à l'aide d'un `\override` au niveau d'un contexte `StaffGroup` (ou son équivalent).

`staff-staff-spacing`
: Il s'agit de la distance entre deux portées consécutives d'un même système. La propriété `staff-staff-spacing` de l'objet `VerticalAxisGroup` d'une portée en particulier peut se redéfinir à l'aide de règles dérogatoires.

`staffgroup-staff-spacing`
: Il s'agit de la distance entre la dernière portée d'un regroupement et la portée suivante, au sein d'un même système, y compris lorsqu'une ou plusieurs lignes de non-portée (tel `Lyrics`) s'insèrent entre les deux. Cette propriété ne concerne pas la dernière portée d'un système. Dans le cas où la propriété `staff-staff-spacing` d'une portée du regroupement a été ajustée au niveau de son propre `VerticalAxisGroup`, cette dernière aura préséance.

Voir aussi

Manuel de notation : Section 5.3.7 [Modification de listes associatives], page 664, Section 4.1.4 [Variables d'espacement vertical fluctuant], page 575.

Fichiers d'initialisation : `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

Espacement de portées isolées

Les `Staff`, `DrumStaff`, `TabStaff` entre autres sont des contextes de « portée » pouvant contenir plusieurs voix, mais pas une portée.

L'espacement de ces *portées isolées* est géré par les propriétés suivantes :

- Propriétés du `VerticalAxisGroup` :
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

Ces propriétés d'objet graphique sont expliquées une à une au chapitre [Propriétés d'espacement au sein d'un système], page 596.

Certaines propriétés supplémentaires s'appliqueront dès lors que ces portées sont regroupées – voir [Espacement de portées regroupées], page 600.

L'exemple suivant illustre la manière de gérer l'espacement de portées isolées à l'aide de la propriété `default-staff-staff-spacing`. Les mêmes règles appliquées de manière dérogatoire au `staff-staff-spacing` produiront les mêmes effets, y compris au sein de regroupements.

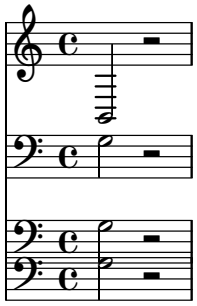
```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 8)
         (minimum-distance . 7)
         (padding . 1))
  }
}

<<
  % The very low note here needs more room than 'basic-distance
  % can provide, so the distance between this staff and the next
  % is determined by 'padding.
  \new Staff { b,2 r | }

  % Here, 'basic-distance provides enough room, and there is no
  % need to compress the space (towards 'minimum-distance) to make
  % room for anything else on the page, so the distance between
```

```
  % this staff and the next is determined by 'basic-distance.
  \new Staff { \clef bass g2 r | }

  % By setting 'padding to a negative value, staves can be made to
  % collide.  The lowest acceptable value for 'basic-distance is 0.
  \new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 3.5)
         (padding . -10))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



Voir aussi

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

Espacement de portées regroupées

Dans les partitions orchestrales ou de grande ampleur, il arrive souvent que des portées soient regroupées. L'espacement est alors plus important entre deux regroupements qu'entre les portées d'un même groupe.

Les *regroupements de portées* tels le `StaffGroup` ou le `ChoirStaff` sont des contextes qui peuvent contenir simultanément une ou plusieurs portées.

L'espacement entre les portées d'un même regroupement est géré par les propriétés suivantes :

- Propriétés du `VerticalAxisGroup` :
  - `staff-staff-spacing`
- Propriétés du `StaffGrouper` :
  - `staff-staff-spacing`
  - `staffgroup-staff-spacing`

Ces propriétés d'objet graphique sont expliquées une à une au chapitre [Propriétés d'espacement au sein d'un système], page 596.

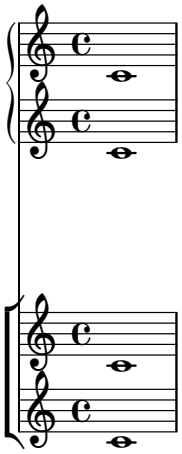
L'exemple suivant illustre la manière de gérer l'espacement de portées regroupées, à l'aide des propriétés de l'objet graphique `StaffGrouper` :

```
\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = #0
```

```
    \override StaffGrouper.staff-staff-spacing.basic-distance = #1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper
                .staffgroup-staff-spacing
                .basic-distance = #20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>
>>
```



Voir aussi

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

Espacement des lignes rattachées à des portées

Les *lignes de non-portée*, comme les `Lyrics` ou les `ChordNames` sont des contextes dont les objets de rendu sont gravés à l'instar des portées – une ligne horizontale dans un système. En fait, les lignes de non-portée sont des contextes qui vont créer un objet de rendu `VerticalAxisGroup` auxquel est attaché le Section "Axis_group_engraver" dans *Référence des propriétés internes*.

L'espacement des lignes de non-portée est géré par les propriétés suivantes :

- Propriétés du `VerticalAxisGroup` :
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`

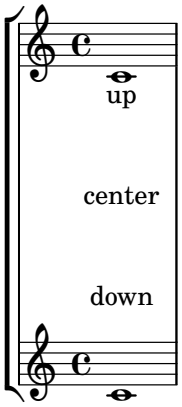
- `nonstaff-unrelatedstaff-spacing`

Ces propriétés d'objet graphique sont expliquées une à une au chapitre [Propriétés d'espacement au sein d'un système], page 596.

L'exemple suivant utilise la propriété `nonstaff-nonstaff-spacing` pour gérer l'espacement entre des lignes consécutives de non-portée. Vous noterez que la valeur élevée attribuée à la clé `stretchability` permet aux paroles de s'étirer plus que de raison.

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
                .nonstaff-nonstaff-spacing
                .stretchability = #1000
  }
}

\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup.staff-staff-spacing =
      #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>
```



Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

4.4.2 Positionnement explicite des portées et systèmes

Pour bien comprendre comment fonctionnent les réglages de `VerticalAxisGroup` et de `\paper` abordés dans les deux rubriques précédentes, rien ne vaut une collection d'exemples illustrant les différentes mises au point du décalage vertical appliqué aux portées et systèmes distribués sur une page.

Une autre approche de l'espacement vertical est le recours à la propriété `NonMusicalPaperColumn.line-break-system-details`. Alors que `VerticalAxisGroup` et `\paper` gèrent un décalage vertical, `NonMusicalPaperColumn.line-break-system-details` spécifiera le positionnement vertical absolu sur la page.

`NonMusicalPaperColumn.line-break-system-details` prend en charge une liste associative de quatre mises au point :

- `X-offset`
- `Y-offset`
- `extra-offset`
- `alignment-distances`

Les dérogations en matière d'objet graphique, y compris celles concernant les `NonMusicalPaperColumn` ci-dessus, peuvent se placer à trois différents endroits de votre fichier source :

- directement au beau milieu des notes
- au sein d'un bloc `\context`
- dans un bloc `\with`

Le réglage de `NonMusicalPaperColumn` s'effectue à l'aide d'une simple commande `\override` au sein d'un bloc `\context` ou `\with`. Dans le cas où il est stipulé au fil des notes, c'est la commande spécifique `\overrideProperty` qui doit intervenir. Voici quelques exemples de réglages de `NonMusicalPaperColumn` à l'aide de la commande `\overrideProperty` :

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 40))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
     (Y-offset . 40))
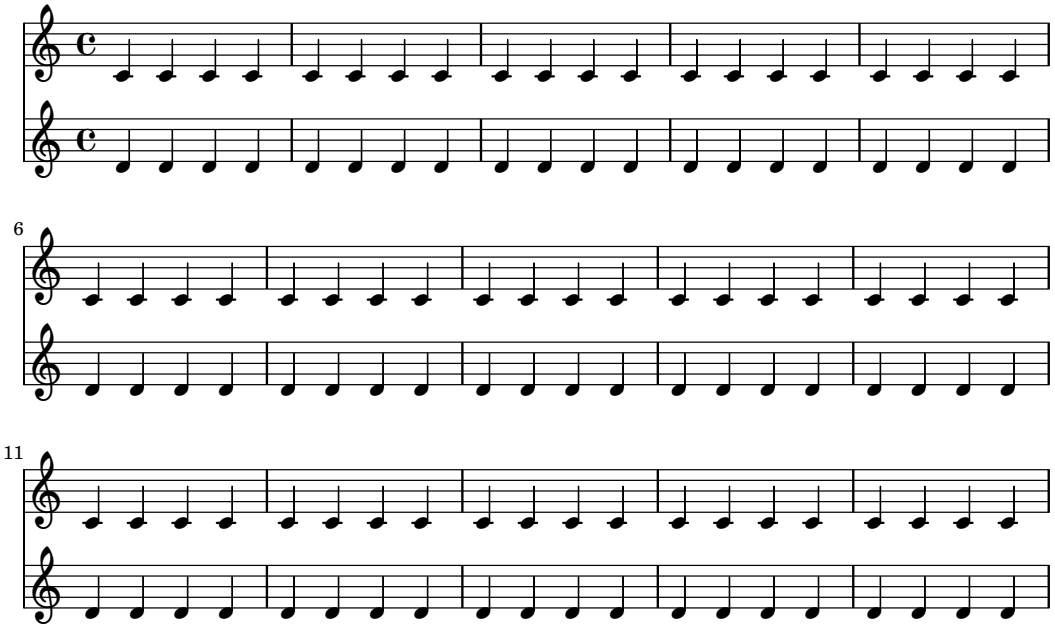
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((alignment-distances . (15)))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
     (Y-offset . 40)
     (alignment-distances . (15)))
```

Nous allons maintenant voir ces différents réglages en action. Commençons par examiner un exemple dépourvu de toute mise au point.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
```

```
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```

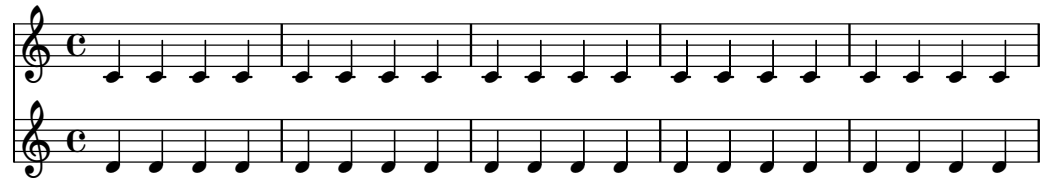


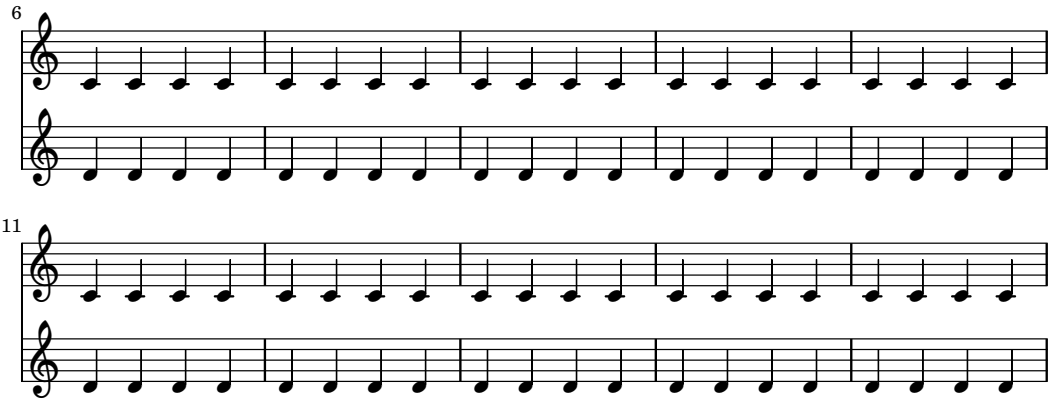
Cette partition isole les informations de saut de ligne ou de page dans une voix spécifique. La mise en forme est ainsi séparée des événements musicaux ; ceci nous permettra d'y voir plus clair au fur et à mesure que nous avancerons. Pour plus de précisions, relisez Section 4.3 [Sauts], page 588.

Les `\break` explicites répartissent la musique en lignes de cinq mesures chacune. L'espacement vertical est celui que LilyPond attribue par défaut. Nous pouvons, afin de fixer explicitement le point d'attache vertical de chacun des systèmes, définir un doublet `Y-offset` en tant qu'attribut du `line-break-system-details` de l'objet `NonMusicalPaperColumn` :

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn
            .line-break-system-details #'((Y-offset . 0))
```

```
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn
            .line-break-system-details #'((Y-offset . 40))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn
            .line-break-system-details #'((Y-offset . 60))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```
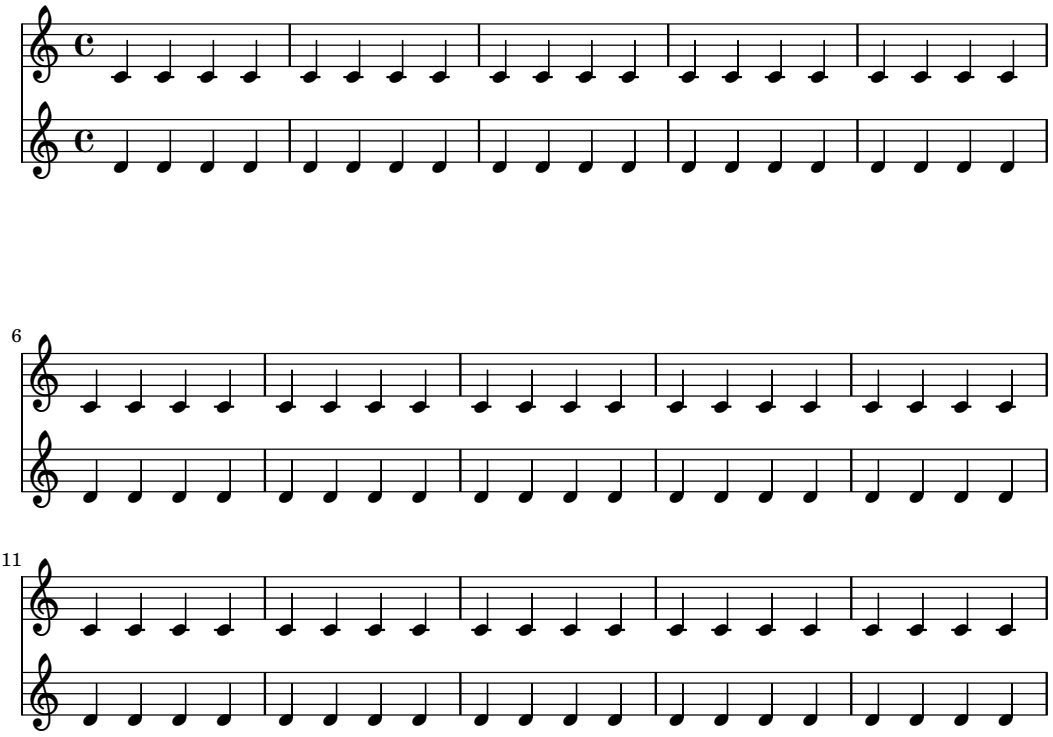




Vous aurez remarqué que nous n'avons déterminé qu'une seule valeur, même si la liste associative de `line-break-system-details` peut en comporter un certain nombre. Vous aurez aussi noté que la propriété `Y-offset` détermine ici le point de départ de chacun des systèmes de la page.

Contrairement au positionnement absolu accessible par `Y-offset` et `X-offset`, il est possible d'opter pour un positionnement relatif à l'aide de la propriété `extra-offset` des `line-break-system-details`. Le placement sera relatif à la mise en forme par défaut ou au positionnement absolu géré par `X-offset` et `Y-offset`. La propriété `extra-offset` prend en argument une `paire` constituée des déplacements sur les axes horizontal et vertical.
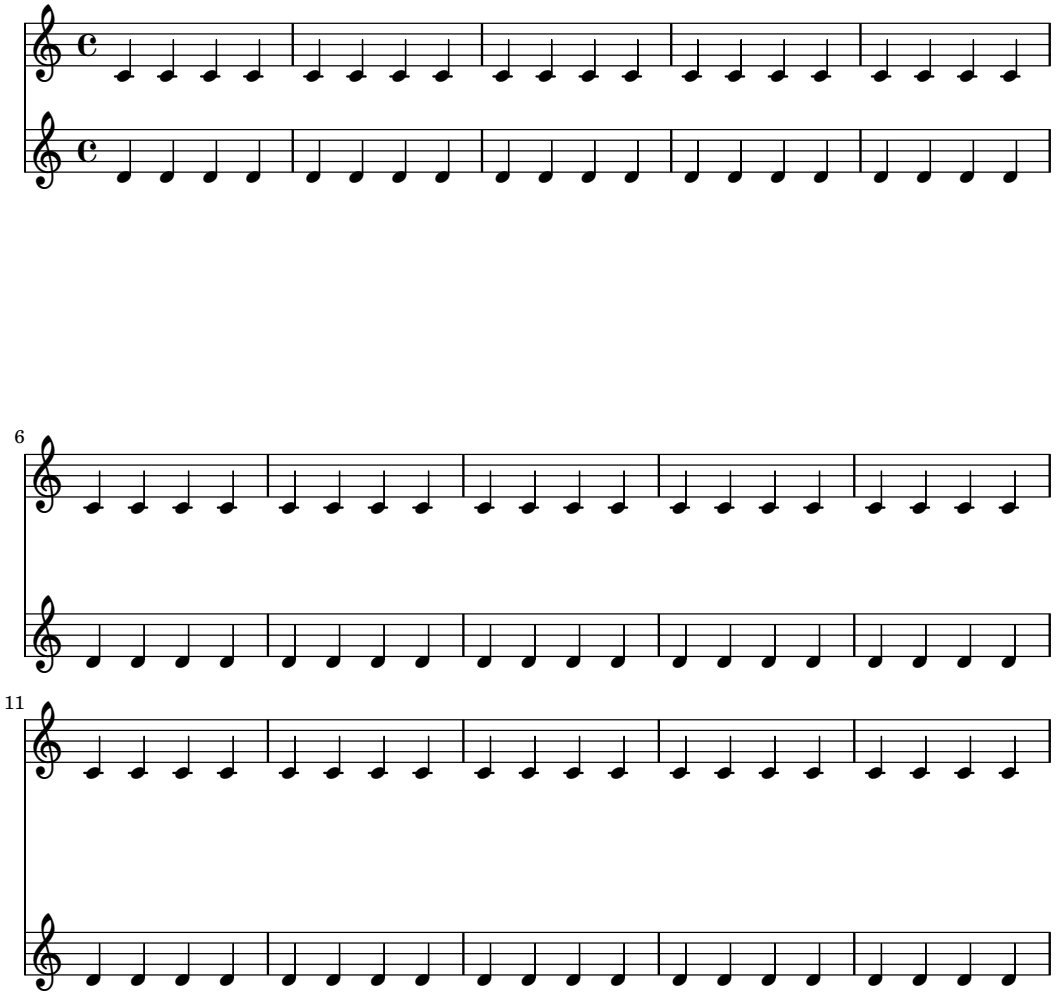
```
\header { tagline = ##f }
```

```
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details #'((extra-offset . (0 . 10)))
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details #'((extra-offset . (0 . 10)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



Maintenant que chaque système est explicitement positionné, nous pouvons jouer sur la distance séparant les portées de chacun des systèmes, grâce à la sous-propriété `alignment-distances` de `line-break-system-details`.
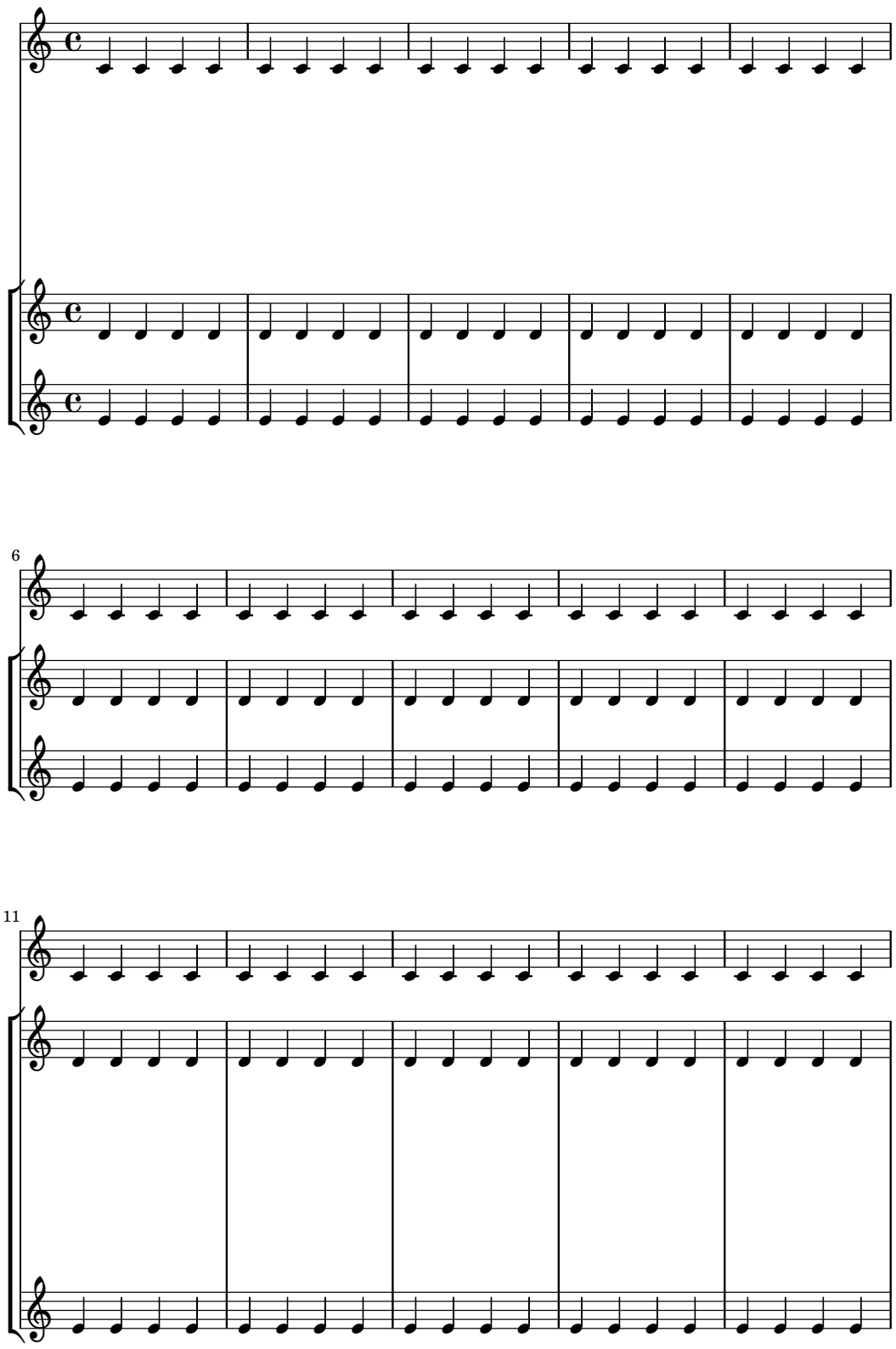
```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details
              #'((Y-offset . 20)
                 (alignment-distances . (10)))
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details
              #'((Y-offset . 60)
                 (alignment-distances . (15)))
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details
              #'((Y-offset . 85)
                 (alignment-distances . (20)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



Nous avons maintenant assigné deux valeurs différentes à l'attribut `line-break-system-details` de l'objet `NonMusicalPaperColumn`. `line-break-system-details` pourrait prendre bien d'autres paramètres d'espacement, y compris un doublet `X-offset`, mais nous n'avons utilisé que `Y-offset` et `alignment-distances` pour contrôler le positionnement de chaque système et de chaque portée. Vous noterez enfin que `alignment-distances` traite le positionnement des portées, non d'un regroupement de portées.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty
            Score
              .NonMusicalPaperColumn
              .line-break-system-details
                #'((Y-offset . 0)
                   (alignment-distances . (30 10)))
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details
```

```
            #'((Y-offset . 60)
               (alignment-distances . (10 10)))
          s1*5 \break
          \overrideProperty
            Score
            .NonMusicalPaperColumn
            .line-break-system-details
              #'((Y-offset . 100)
                 (alignment-distances . (10 30)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new StaffGroup <<
        \new Staff { \repeat unfold 15 { d'4 d' d' d' } }
        \new Staff { \repeat unfold 15 { e'4 e' e' e' } }
      >>
    >>
  }
}
```



Quelques points à prendre en considération :

- Lorsque vous utilisez `alignment-distances`, les paroles et autres lignes de non-portée ne comptent pas pour une portée.
- Les nombres fournis à `X-offset`, `Y-offset`, `extra-offset` et `alignment-distances` sont considérés comme des multiples de la distance entre des portées adjacentes. Des valeurs positives remontent les portées et paroles, des valeurs négatives les descendent.
- Dans la mesure où `NonMusicalPaperColumn.line-break-system-details` permet de positionner systèmes et portées n'importe où sur une page, vous pourriez être en contradiction

avec les dimensionnements de la feuille ou bien aboutir à des surimpressions. Soyez donc raisonnable quant aux différentes valeurs que vous affectez à ces réglages.

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.4.3 Résolution des collisions verticales

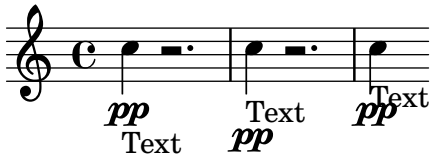
Vous savez de manière intuitive qu'un certain nombre d'objets en matière de notation musicale appartiennent à la portée, et que d'autres se placent en dehors de la portée. Entre autres objets externes, nous avons les marques repères, les textes et les nuances ; nous les appellerons « objets extérieurs à la portée ». La règle adoptée par LilyPond pour positionner verticalement ces objets extérieurs consiste à les placer au plus près de la portée tout en prenant garde d'éviter qu'il y ait chevauchement.

LilyPond utilise la propriété `outside-staff-priority` afin de déterminer si un objet est ou non un objet extérieur à la portée : lorsque la valeur de `outside-staff-priority` est numérique, il s'agit d'un objet extérieur à la portée. De plus, la propriété `outside-staff-priority` indique à LilyPond l'ordre dans lequel ces objets doivent être disposés.

Tout d'abord, LilyPond dispose tous les objets qui ne sont pas externes. Les objets extérieurs à la portée sont alors triés selon l'ordre croissant de leur `outside-staff-priority`. Enfin, LilyPond prend chacun des ces objets et les positionne de telle sorte qu'il n'entrent pas en collision avec ceux qui on déjà été placés. Autrement dit, lorsque deux objets devraient se placer au même endroit, celui dont la `outside-staff-priority` est la plus faible sera disposé au plus près de la portée.

Une liste des `outside-staff-priorities` est disponible à la rubrique Section "La propriété outside-staff-priority" dans *Manuel d'initiation*.

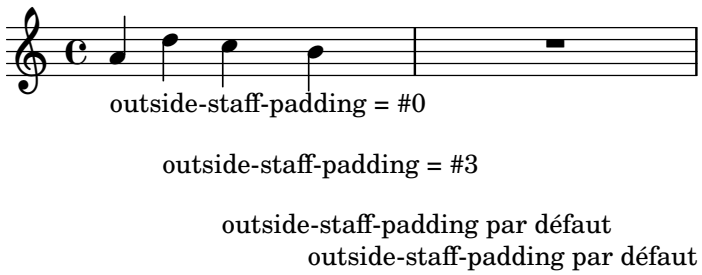
```
\relative c'' {
  c4_"Text"\pp
  r2.
  \once \override TextScript.outside-staff-priority = #1
  c4_"Text"\pp % this time the text will be closer to the staff
  r2.
  % by setting outside-staff-priority to a non-number,
  % we disable the automatic collision avoidance
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Text"\pp % now they will collide
}
```



Le décalage vertical entre des objets extérieurs à la portée se contrôle par la propriété `outside-staff-padding`.

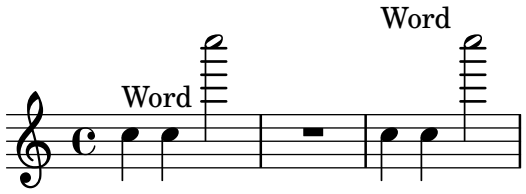
```
\relative {
  \once \override TextScript.outside-staff-padding = #0
  a'4-"outside-staff-padding = #0"
  \once \override TextScript.outside-staff-padding = #3
  d-"outside-staff-padding = #3"
```

```
  c-"outside-staff-padding par défaut"
  b-"outside-staff-padding par défaut"
  R1
}
```



Par défaut, les objets extérieurs à la portée sont positionnés en évitant les collisions horizontales avec des objets précédemment positionnés. Ceci peut cependant générer des situations où des objets se trouvent horizontalement trop proches. Comme l'illustre l'exemple suivant, la propriété `outside-staff-horizontal-padding` permet d'accroître l'espace horizontal requis et repoussera verticalement un objet pour éviter qu'il ne soit trop proche d'éventuelles lignes supplémentaires.

```
\relative {
  c''4^"Word" c c''2
  R1
  \once \override TextScript.outside-staff-horizontal-padding = #1
  c,,4^"Word" c c''2
}
```



Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.5 Espacement horizontal
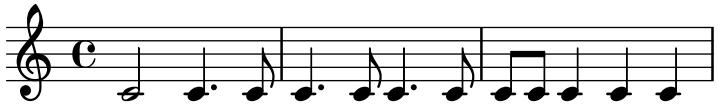
4.5.1 Généralités sur l'espacement horizontal

Le moteur d'espacement traduit les différences de durée en distances étirables (*springs* pour ressorts) de différentes longueurs. Des durées importantes prennent ainsi plus de place que des durées moins longues. Les durées les plus courtes se verront attribuer un espace fixe, contrôlé par la propriété `shortest-duration-space` de l'objet Section "SpacingSpanner" dans *Référence des propriétés internes*. Au plus la durée s'allonge, au plus elle prendra d'espace : le doublement d'une durée attribuera à la note un espace fixé d'après la propriété `spacing-increment`.

L'exemple suivant comporte des blanches, des noires et un certain nombre de croches. La croche est suivie d'un espace de la largeur d'une tête de note ; pour la noire, cet espace est de deux têtes ; il est de trois pour la blanche.

```
\relative c' {
  c2 c4. c8
```

```
  c4. c8 c4. c8
  c8 c c4 c c
}
```



`spacing-increment` est normalement défini à 1,2 espace de portée – ce qui correspond à peu près à la largeur d'une tête de note – et `shortest-duration-space` à 2,0. La note la plus courte s'étendra donc sur l'équivalent de 2,4 espaces de portée (deux fois le `spacing-increment`). Le point de départ de cet espace se situe à l'extrémité gauche du symbole ; la note la plus courte est donc suivie en général d'un espace égal à la largeur d'une tête de note.
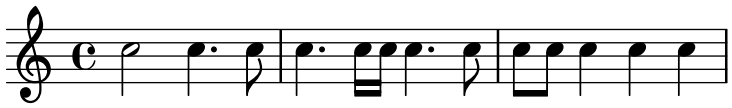
Si l'on suit à la lettre ce qui précède, ajouter une simple triple croche à une partition qui comporte déjà des croches et des doubles augmentera considérablement son volume : la durée la plus courte n'est plus la double mais la triple croche, ce qui aura pour conséquence d'ajouter une largeur de tête à chacune des notes. Pour s'affranchir de cet effet quelque peu pervers, la durée la plus courte prise en considération au niveau de l'espacement n'est pas la note la plus brève de la partition, mais celle qui apparaît le plus souvent.

La courte durée la plus fréquente est déterminée à partir de la note la plus courte de chaque mesure. C'est elle qui servira de base pour l'espacement, à cette nuance près que la plus courte durée ne saurait être strictement supérieure à la croche.

Ces durées peuvent être adaptées. Vous pouvez définir la durée de base pour les espacements grâce à la propriété `common-shortest-duration` de l'objet Section "SpacingSpanner" dans *Référence des propriétés internes*. La durée maximale de cet étalon, normalement la croche, est gérée par la propriété `base-shortest-duration`.

Les notes plus courtes que la note témoin sont suivies d'un espace proportionnel à la durée témoin. Si donc nous ajoutions quelques doubles croches à l'exemple précédent, elles seraient suivies d'une demie largeur de tête :
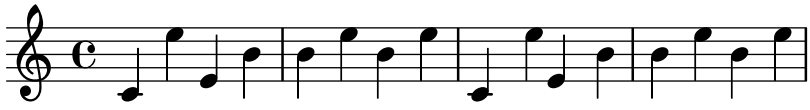
```
\relative { c''2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
```



Comme le met en évidence notre *Essai sur la gravure musicale automatisée*, la direction des hampes peut influencer l'espacement – voir Section "Espacement" dans *Essai* et s'ajuster à l'aide de la propriété `stem-spacing-correction` de l'objet Section "NoteSpacing" dans *Référence des propriétés internes* créé pour chaque contexte Section "Voice" dans *Référence des propriétés internes*.

L'objet `StaffSpacing`, généré au niveau d'un contexte Section "Staff" dans *Référence des propriétés internes*, possède une même propriété qui contrôlera l'espacement hampe-barre de mesure.

L'exemple suivant montre ces adaptations, tout d'abord selon les réglages par défaut, puis avec des corrections forcées.



L'espacement spécifique à la notation proportionnelle fait l'objet d'une Section "rubrique dédiée" dans *Manuel de notation*.

Voir aussi

Essai sur la gravure musicale automatisée : Section "Espacement" dans *Essai*.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SpacingSpanner" dans *Référence des propriétés internes*, Section "NoteSpacing" dans *Référence des propriétés internes*, Section "StaffSpacing" dans *Référence des propriétés internes*, Section "NonMusicalPaperColumn" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Il n'existe pas de mécanisme simple et efficace qui permette de forcer manuellement l'espacement. La solution ci-dessous permet cependant « d'aérer » artificiellement une partition ; il vous suffit d'ajuster la valeur du décalage *(padding)* autant que de besoin.

```
\override Score.NonMusicalPaperColumn.padding = #10
```
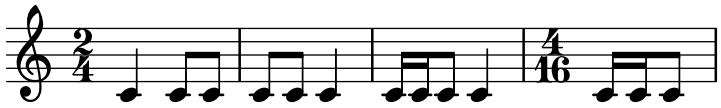
Il n'y a aucun moyen de diminuer l'espacement.

4.5.2 Changement d'espacement en cours de partition

Il arrive, au cours d'un même mouvement, qu'une nouvelle partie modifie substantiellement la notion de valeur brève et valeur longue. La commande `newSpacingSection` permet alors de réinitialiser les paramètres d'espacement.
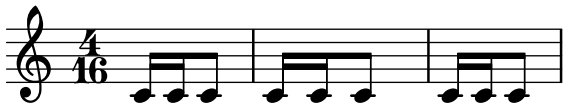
Dans l'exemple qui suit, le changement de métrique marque le début d'une nouvelle partie ; remarquez comme les doubles-croches sont alors automatiquent un peu plus espacées :

```
\time 2/4
c4 c8 c
c8 c c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```



La commande `\newSpacingSection` crée un nouvel objet `SpacingSpanner` à cet instant musical. Si toutefois les ajustements apportés à l'espacement automatique ne se révélent pas satisfaisants, ils peuvent s'adapter à l'aide d'`\overrides`. Ces amendements doivent intervenir au même moment que la commande `\newSpacingSection` ; ils produiront leurs effets jusqu'à ce qu'ils soient à nouveau modifiés par une nouvelle section, comme ici :

```
\relative c' {
  \time 4/16
  c16[ c c8]
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #2
  c16[ c c8]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
  c16[ c c8]
}
```



Voir aussi
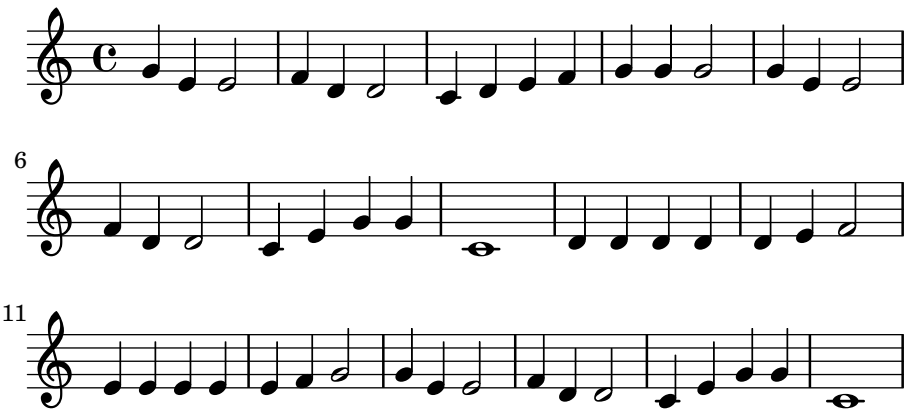
Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SpacingSpanner" dans *Référence des propriétés internes*.
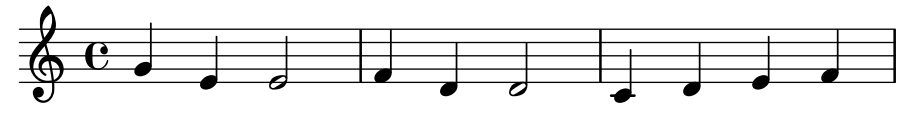
4.5.3 Modification de l'espacement horizontal
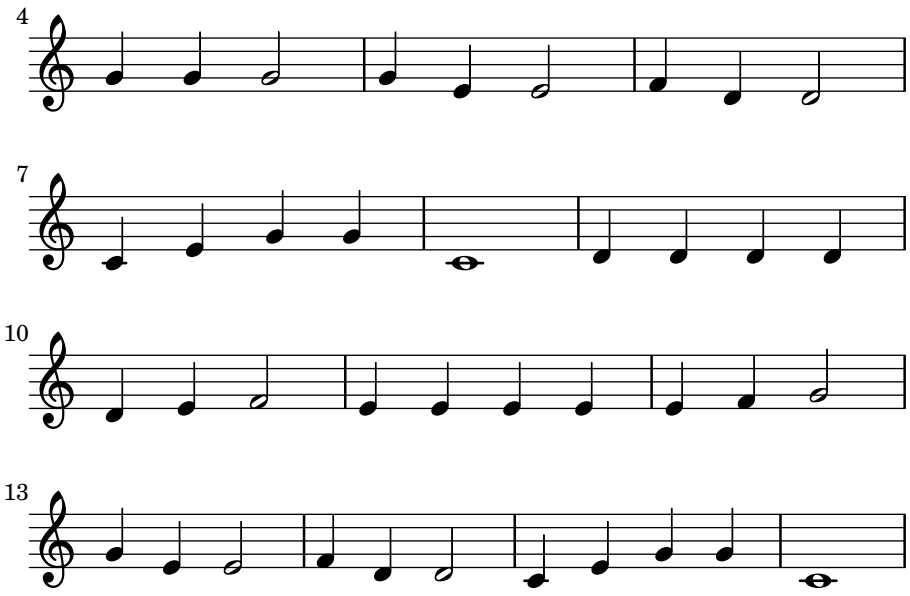
Vous pouvez influencer l'espacement horizontal à l'aide de la propriété `base-shortest-duration`. Comparons les deux partitions qui suivent, toutes deux montrant la même musique. La première partition applique les réglages par défaut, alors que la seconde bénéficie d'un ajustement de la propriété `base-shortest-duration`. Au plus la valeur de `ly:make-moment` est grande, au plus la musique sera resserrée. En effet, `ly:make-moment` construit une durée : `1 4` est plus long que `1 16`.

```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
}
```



```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
    }
  }
}
```

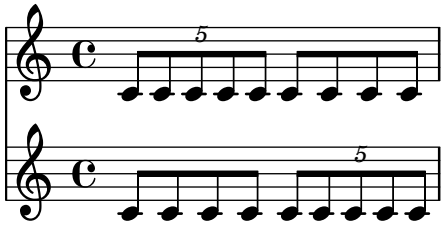




Étirement uniforme des n-olets

L'espacement au sein d'un n-olet dépend par défaut d'un certain nombre de facteurs qui ne sont pas liés à la durée (altération, changement de clef, etc.). `Score.SpacingSpanner.uniform-stretching` permet d'ignorer ces symboles et, par voie de conséquence, de forcer l'espacement sur la simple durée. Notez bien que cette propriété s'appliquera à toute la partition, puisque mentionnée au sein d'un bloc `\layout`.

```
\score {
  <<
    \new Staff \relative c' {
      \tuplet 5/4 { c8 c c c c } c8 c c c
    }
    \new Staff \relative c' {
      c8 c c c \tuplet 5/4 { c8 c c c c }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```

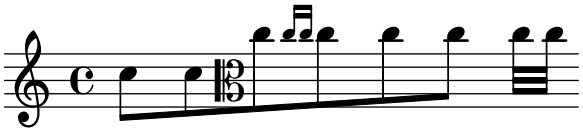


Espacement strict des notes

L'activation du commutateur `strict-note-spacing` permet d'espacer les notes sans tenir compte des clefs, barres de mesure ou notes d'ornement qui pourraient apparaître :

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative {
```

```
c''8[ c \clef alto c \grace { c16 c } c8 c c]  c32[ c] }
```



Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.5.4 Largeur de ligne

Deux réglages de base ont une influence considérable sur l'espacement : `line-width` et `indent`. Tous deux se placent dans le bloc `\layout`. Ils contrôleront la longueur des lignes et l'indentation de la première.

L'activation du commutateur `ragged-right` au sein du bloc `\layout` permet de terminer les systèmes naturellement plutôt que de les voir s'étirer sur toute la largeur de la page. Cette option est particulièrement utile lorsque vous traitez de courts fragments, ou bien pour vérifier ce que donnerait l'espacement naturel. Bien qu'il soit désactivé par défaut, il sera activé si la partition ne comporte qu'un seul système.

Le fonctionnement de l'option `ragged-last` est en tout point identique à celui de `ragged-right`, à ceci près qu'il ne concerne que la dernière ligne de la partition. Il n'y a pas de restriction quant à cette ligne. Il en va de même que pour le formatage d'un paragraphe de texte, la dernière ligne s'arrête au dernier caractère.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.5.5 Notation proportionnelle

LilyPond prend en charge la notation proportionnelle. Il s'agit dans ce cas de représenter la notation selon un espacement strictement relatif aux durées. Ce type d'espacement pourrait se comparer à l'utilisation de papier millimétré pour positionner les notes au fil de la portée. Certaines œuvres de la fin du XX^{e} siècle et à l'aube du XXI^{e} utilisent cette proportionnalité dans le but de clarifier des structures rythmiques complexes, d'aider au positionnement d'indications temporelles ou autres éléments graphiques directement dans la partition.

LilyPond met à votre disposition cinq réglages différents, qui peuvent s'utiliser conjointement ou individuellement, aux fins de mettre au point cette notation proportionnelle.

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove "Separating_line_group_engraver"`
- `\override PaperColumn.used = ##t`

Nous allons examiner, dans les différents exemples qui suivent, les effets de ces réglages et comment ils interagissent.

Commençons par cette mesure toute simple qui utilise l'espacement classique et justifiée à gauche.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
}
```



Vous constatez que la blanche qui entame la mesure prend moins de la moitié de l'espace. De même, les doubles croches et le quintolet de doubles (donc des vingtièmes de ronde) qui terminent cette mesure n'en occupent pas la moitié de l'espace horizontal.

En matière de gravure traditionnelle, cet espacement correspond tout à fait à nos attentes, puisque nous pouvons rogner de l'espace sur la blanche et ainsi gagner en largeur sur toute la mesure qui fait une ronde.

Par contre, si nous avons besoin d'insérer une indication temporelle ou un autre graphisme en surplomb ou en dessous de notre partition, nous aurons besoin de la notation proportionnelle. Celle-ci s'active en définissant la propriété `proportionalNotationDuration`.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
 \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



La blanche du début et les notes plus rapides de la deuxième moitié de la mesure occupent maintenant exactement le même espace horizontal. Nous pourrions donc y insérer, au-dessus ou au-dessous, une indication temporelle ou autre graphisme.

`proportionalNotationDuration` est une propriété attachée au contexte `Score`. Rappelez-vous que vous pouvez régler les propriétés d'un contexte à trois différents endroits de votre fichier : dans un bloc `\with`, dans un bloc `\context` ou au beau milieu de la musique à l'aide de la commande `\set`. Vous pouvez donc définir `proportionalNotationDuration` selon l'une de ces trois façons, à l'instar de n'importe quelle définition de contexte.

La propriété `proportionalNotationDuration` prend en unique argument la durée de référence qui servira de base pour espacer toute la musique. La fonction Scheme `make-moment` intégrée à LilyPond prend deux arguments : un numérateur et un dénominateur qui représentent

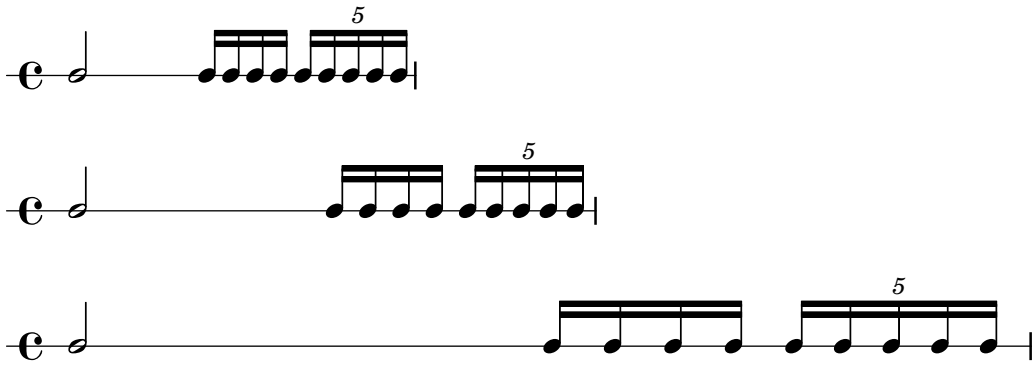
une fraction de ronde. L'appel de `(ly:make-moment 1/20)` produit donc une durée de référence égale à un vingtième de ronde. Vous pourriez tout aussi bien utiliser `(ly:make-moment 1/16)`, `(ly:make-moment 1/8)` ou `(ly:make-moment 3/97)`.

Se pose alors le problème de fournir la juste durée de référence à `proportionalNotationDuration`. Il faut en l'occurrence procéder par tâtonnement, en commençant par une valeur proche de la note la plus rapide (la durée la plus courte) du morceau. Au plus la durée de référence est petite, au plus la musique sera étalée ; à l'inverse, une durée de référence élevée produira une musique resserrée.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/8)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/16)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/32)
    }
  }
}
```

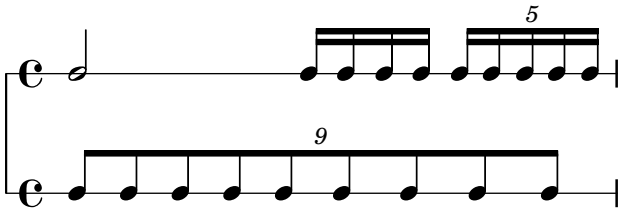


Vous ne manquerez pas de noter qu'une durée de référence trop grande, comme la croche pour la première ligne, a pour conséquence de resserrer la musique, ce qui peut aboutir à des chevauchements de têtes. Vous remarquez aussi que, par principe, la notation proportionnelle occupe beaucoup plus d'espace horizontal que l'espacement traditionnel. La notation proportionnelle met en évidence le rythme au détriment de l'espacement horizontal.

Examinons à présent le moyen d'optimiser l'espacement de n-olets en tuilage.

Reprenons notre exemple de départ, avec son espacement traditionnel, et ajoutons lui une portée incluant un autre type de n-olet.

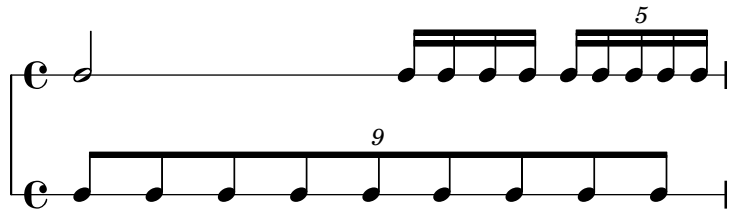
```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
}
```



L'espacement est loin d'être idéal, pour la simple raison que l'espacement régulier des notes de la portée inférieure ne s'étire pas uniformément. Il est vrai que de telles constructions complexes en n-olets sont assez rares en gravure traditionnelle, ce qui explique que les règles qu'elle applique peuvent amener à ce résultat. Le recours à `proportionalNotationDuration` permet d'arranger les choses.

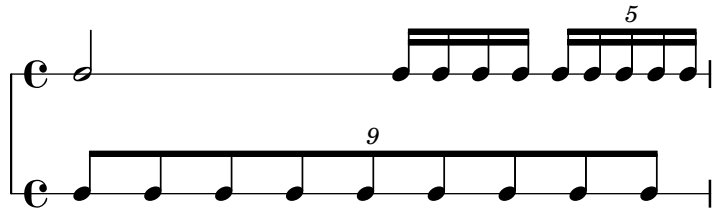
```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
```

```
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



Cependant, si l'on observe de près, il est évident que les notes de la deuxième moitié du ennaolet ont tendance à s'espacer légèrement plus que celles de la première moitié. Afin d'uniformiser cet étalement, nous allons activer le `uniform-stretching`, propriété attachée au `SpacingSpanner`.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



L'espacement sur les deux portées est maintenant correct, les relations rythmiques sont clairement perceptibles, et nous pourrions y insérer une indication temporelle ou autre graphisme selon notre envie.

Notez bien que la prise en charge de la notation proportionnelle par LilyPond demande que, dans chaque partition, soit activée la propriété `uniform-stretching` du `SpacingSpanner`. Dans le cas contraire, utiliser `proportionalNotationDuration` aura pour conséquence, entre autres, un espacement erroné lorsque vous y aurez inséré des silences invisibles *skip*.

Le `SpacingSpanner` est en fait un objet graphique abstrait attaché au contexte `Score`. Tout comme pour la propriété `proportionalNotationDuration`, les réglages du `SpacingSpanner`

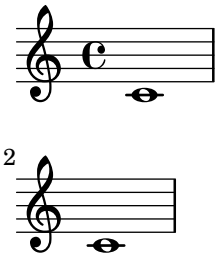
peuvent se faire à trois différents endroits de votre fichier : dans un bloc `\with`, dans un bloc `\context` ou au beau milieu de la musique à l'aide de la commande `\set`.

Gardez bien à l'esprit qu'il n'y a qu'un seul `SpacingSpanner` par `Score`. Il s'ensuit que `uniform-stretching` est soit activé, soit désactivé, et dans tous les cas pour l'intégralité de la partition. Vous pourriez toutefois avoir besoin de modifier ce comportement en cours de partition, et recourir alors à l'instruction `\newSpacingSection` – pour de plus amples détails, voir la rubrique Section 4.5.2 [Changement d'espacement en cours de partition], page 614.

Intéressons-nous maintenant au `Separating_line_group_engraver`, qui est désactivé pour la plupart des partitions en notation proportionnelle. Voici ce qui apparaît dans une partition traditionnelle : il y a toujours un « espace préservé » juste avant la première note de chaque portée.

```
\paper {
  indent = #0
}

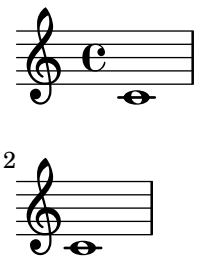
\new Staff {
  c'1
  \break
  c'1
}
```



Cet espace, géré par le `Separating_line_group_engraver`, est aussi présent lorsqu'intervient un changement de métrique, d'armure ou de clef. Désactiver le `Separating_line_group_engraver` revient à réduire cet espace à zéro.

```
\paper {
  indent = #0
}

\new Staff \with {
  \remove "Separating_line_group_engraver"
} {
  c'1
  \break
  c'1
}
```



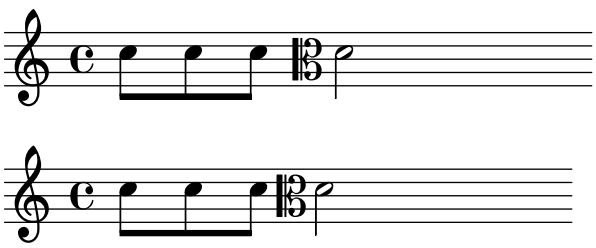
Les éléments non musicaux tels que métrique, armure, clef et altérations, posent problème lorsqu'on travaille en notation proportionnelle. Bien qu'aucune notion de durée ne leur soit attachée, ces éléments « consomment » de l'espace. Différentes approches permettent cependant de gérer ce problème.

Éviter les problèmes d'espacement avec l'armure est chose aisée : il suffit qu'il n'y en ait pas ! C'est bien souvent le cas en musique contemporaine, où l'on trouve le plus d'ouvrages en notation proportionnelle. Il en va de même pour la métrique, et tout particulièrement lorsque la partition comporte un quadrillage temporel ou autres graphismes. L'absence de métrique reste cependant exceptionnelle et la plupart des partitions en notation proportionnelle laissent apparaître quelques métriques. Il est par contre pratiquement impossible de se passer de clef et d'altération.

L'une des options permettant de s'affranchir de l'espacement dû aux éléments non musicaux consiste en l'activation de la propriété `strict-note-spacing` attachée au `SpacingSpanner`. Observons les deux portées suivantes :

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  c''8 8 8 \clef alto d' 2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 8 8 \clef alto d' 2
}
```



Toutes deux affichent un espacement proportionnel. Cependant, la première ligne laisse apparaître un espacement plus lâche en raison de la présence d'un changement de clef. En ce qui concerne la deuxième ligne, l'espacement est strictement observé dès lors que la propriété `strict-note-spacing` a préalablement été activée. Comme vous pouvez le constater, l'activation de `strict-note-spacing` a pour conséquence que l'algorithme d'espacement ignore tout bonnement la largeur des métriques, armures, clefs et altérations.

En plus de ceux que nous venons de voir, vous trouverez d'autres réglages en usage dans la notation proportionnelle comme, entre autres,

- `\override SpacingSpanner.strict-grace-spacing = ##t`
- `\set tupletFullLength = ##t`
- `\override Beam.breakable = ##t`
- `\override Glissando.breakable = ##t`
- `\override TextSpanner.breakable = ##t`
- `\remove "Forbid_line_break_engraver"` (dans un contexte de voix)

Ces différents réglages permettent un espacement strict des notes d'ornement, d'étendre les indications de n-olet afin d'indiquer de façon évidente leurs bornes et d'autoriser le tronçonnement des extenseurs à l'occasion d'un saut de ligne ou de page. Nous vous renvoyons aux différentes rubriques associées du manuel pour chacun de ces réglages.

Voir aussi

Manuel de notation : Section 4.5.2 [Changement d'espacement en cours de partition], page 614.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.6 Réduction du nombre de pages de la partition

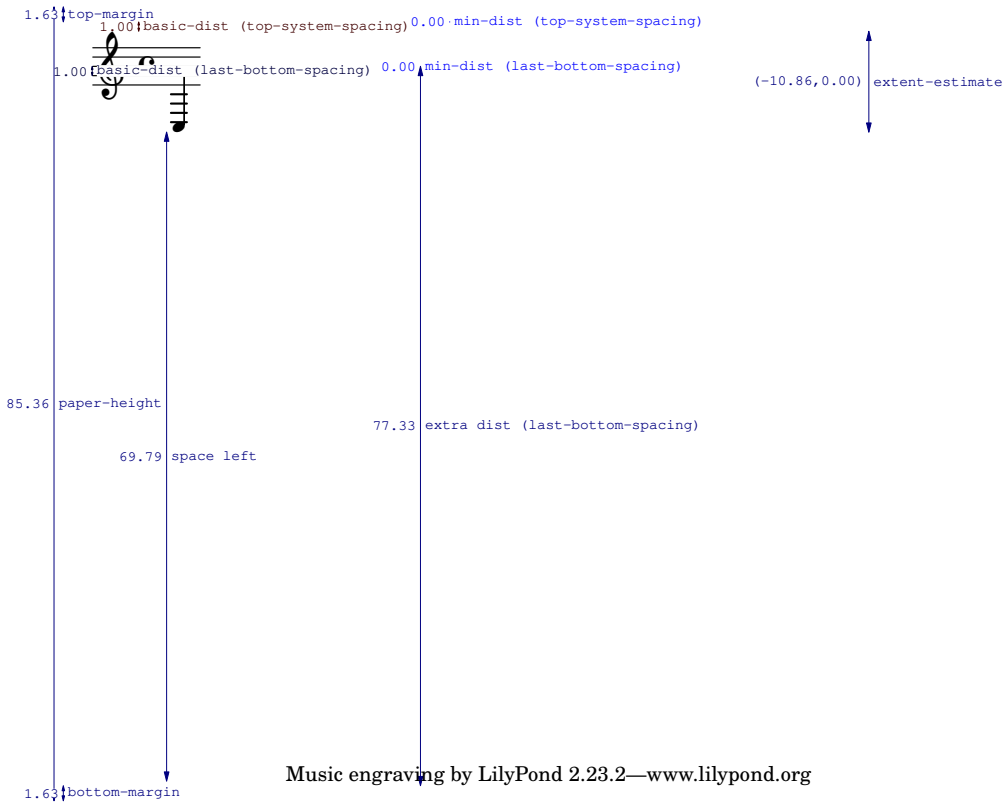
Vous pourriez un jour être confronté au problème suivant : l'une des pages de votre partition ne comporte que deux portées alors que, ce qui est d'autant plus frustrant, l'espace libre sur les autres pages aurait permis une distribution différente.

L'instruction `annotate-spacing` se révèle être un outil indispensable pour l'analyse des problèmes de mise en forme. Cette commande met en surimpression la valeur des différentes variables d'espacement et de mise en forme, comme nous allons le voir dans la rubrique Section 4.6.1 [Mise en évidence de l'espacement], page 624.

4.6.1 Mise en évidence de l'espacement

Le meilleur moyen d'appréhender les différentes variables de dimensionnement vertical sur lesquelles vous pouvez jouer au niveau de la mise en page consiste à activer, au sein du bloc `\paper`, la fonction `annotate-spacing` :

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



Toutes les dimensions sont exprimées en espace de portée *(staff-space)* quelle que soit l'unité mentionnée dans les blocs `\paper` ou `\layout`. Dans cet exemple, la hauteur de la feuille (`paper-height`) est de 59,75 espaces de portée (`staff-spaces`) et la taille de portée (`staff-size`) de 20 points – sa valeur par défaut. Notez que :

1 point = (25,4/72,27) mm

1 staff-space = (`staff-size`)/4 pts
= (`staff-size`)/4 * (25,4/72,27) mm

Dans le cas qui nous occupe, un `staff-space` égale environ 1,757 millimètres. Les 59,75 `staff-spaces` de `paper-height` correspondent donc à 105 millimètres, soit la hauteur d'une feuille au format A6 à l'italienne. Les paires `(a,b)` sont des intervalles, *a* en étant l'extrémité inférieure et *b* l'extrémité supérieure.

Voir aussi

Manuel de notation : Section 4.2.2 [Définition de la taille de portée], page 585.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

4.6.2 Modification de l'espacement

Les informations fournies par `annotate-spacing` en matière de dimensionnement vertical sont incomparables. Pour plus de détails sur les manières de modifier marges et autres variables connexes, consultez la rubrique Section 4.1 [Mise en forme de la page], page 571.

En dehors des marges, vous disposez de quelques moyens supplémentaires pour gagner de l'espace :

- Rapprocher les systèmes le plus possible les uns des autres, de telle sorte qu'il en tienne un maximum sur une même page, tout en les espaçant suffisamment pour éviter le blanc en bas de page.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Forcer le nombre de systèmes par page. Ceci peut se révéler judicieux à deux titres. D'une part, le fait de définir un nombre de systèmes – même s'il est égal à la valeur par défaut – peut aboutir à plus de systèmes par page dans la mesure où l'une des étapes d'estimation des hauteurs est tout simplement sautée. D'autre part, réduire le nombre de systèmes par page permet d'en disposer plus sur les suivantes. Par exemple, avec un nombre par défaut de 11 systèmes par page, l'instruction suivante le force à 10.

```
\paper {
  system-count = #10
}
```
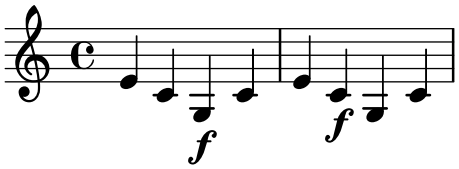
- Forcer le nombre de pages. L'instruction suivante forcera la musique à se répartir sur deux pages.

```
\paper {
  page-count = #2
}
```

- Éviter ou réduire les objets qui augmentent la hauteur des systèmes. Par exemple, un crochet d'alternative en fin de reprise consomme de l'espace. Dans le cas où il s'étend sur deux systèmes, il occupera plus d'espace que si seul le premier comportait l'indication. Autre exemple, les indications de nuance qui se « détachent » d'un système peuvent être rapprochées de la portée :
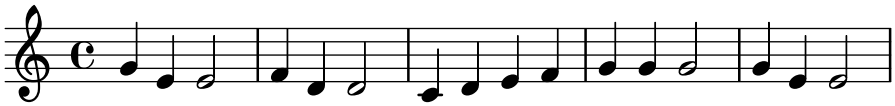
```
\relative e' {
```

```
  e4 c g\f c
  e4 c g-\tweak X-offset #-2.7 \f c
}
```
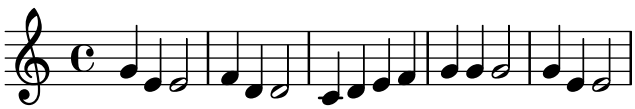


- Modifier l'espacement horizontal à l'aide du `SpacingSpanner`, comme indiqué à la rubrique Section 4.5.3 [Modification de l'espacement horizontal], page 615. Voici ce que donne l'espacement par défaut :

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```



Par contre, le fait de modifier la valeur de la propriété `common-shortest-duration` en passant de `1/4` à `1/2` – bien que la noire soit la durée la plus courante, nous prenons une valeur plus longue – donnera un effet « resserré » à la musique :

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration =
        #(ly:make-moment 1/2)
    }
  }
}
```



La propriété `common-shortest-duration` ne peut être modifiée dynamiquement. Elle se place toujours dans un bloc `\context` et s'applique à l'intégralité de la partition.

Voir aussi

Manuel de notation : Section 4.1 [Mise en forme de la page], page 571, Section 4.5.3 [Modification de l'espacement horizontal], page 615.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

5 Modification des réglages prédéfinis

LilyPond est conçu pour générer, par défaut, des partitions de la plus haute qualité. Cependant, on peut parfois avoir à modifier cette mise en forme par défaut. Celle-ci est réglée par tout un ensemble de « leviers et manettes » plus connus sous le terme de « propriétés », dont ce chapitre ne cherche pas à faire l'inventaire exhaustif – le chapitre Section "Retouche de partition" dans *Manuel d'initiation* du manuel d'initiation vous en propose un aperçu. Le propos est plutôt ici de mettre en évidence les différents groupes auxquels s'apparentent ces contrôles et d'expliquer comment trouver le bon levier pour obtenir tel ou tel effet en particulier.

Les moyens de contrôle des différents réglages sont décrits dans un document séparé, *Référence des propriétés internes*. Ce guide répertorie toutes les variables, fonctions et autres options que LilyPond met à votre disposition. Il est consultable en ligne (`https://lilypond.org/doc/stable/Documentation/internals/`), au format HTML ; il est également inclus dans la documentation fournie avec le logiciel.

En sous-main, LilyPond se sert du langage Scheme (un dérivé du LISP) comme infrastructure. Modifier les choix de mise en page revient à pénétrer dans les entrailles du programme, et de ce fait requiert l'emploi du Scheme. Les fragments de Scheme, dans un fichier `.ly`, sont introduits par le caractère *hash* (`#`), improprement surnommé « dièse ».[1]

5.1 Contextes d'interprétation

Nous allons voir ici ce que sont les contextes et comment les modifier.

Voir aussi

Manuel d'initiation : Section "Contextes et graveurs" dans *Manuel d'initiation*.

Fichiers d'initialisation : `ly/engraver-init.ly`, `ly/performer-init.ly`.

Morceaux choisis : Section "Contextes et graveurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "Engravers and Performers" dans *Référence des propriétés internes*.

5.1.1 Tout savoir sur les contextes

Les contextes sont hiérarchisés :

Définitions de la sortie – hiérarchie des contextes

Les lignes qui suivent traitent de l'intérêt des définitions de sorties lorsque l'on travaille avec les contextes. Des exemples de définitions seront présentés plus avant – voir [Modification de tous les contextes d'un même type], page 639.

Alors que la musique écrite dans un fichier fait référence à des types ou noms de contexte, les contextes ne sont effectivement créés que lorsque la musique est interprétée. LilyPond interprète la musique sous le contrôle d'une « définition de sortie », voire différemment selon le cas, et génère ainsi différents résultats. La définition de sortie appropriée pour une sortie imprimable est spécifiée à l'aide d'un `\layout`.

Une définition de sortie beaucoup plus simple sera utilisée pour produire une sortie MIDI, spécifiée à l'aide d'un `\midi`. LilyPond utilise en interne plusieurs autres définitions de sortie, notamment dans le cadre du combinateur automatique de parties (voir [Regroupement automatique de parties], page 189) ou la reproduction d'extraits (voir [Citation d'autres voix], page 221).

Les définitions de sortie ont pour objet non seulement de définir la relation entre les contextes, mais aussi leurs réglages par défaut. Si la plupart des adaptations prennent habituellement place

[1] Le Section "Tutoriel Scheme" dans *Extension de LilyPond* fournit quelques notions de base pour saisir des nombres, des listes, des chaînes de caractères ou des symboles, en Scheme.

au sein d'un bloc `\layout`, les réglages affectant le Midi ne seront effectifs que s'ils interviennent au sein d'un bloc `\midi`.

Certains réglages affectent plusieurs sorties : par exemple, lorsque `autoBeaming` est désactivé dans un contexte, les ligatures sont considérées comme marquant un mélisme dans le but de faire correspondre la musique aux paroles comme indiqué dans [Durée automatique des syllabes], page 291. Cette correspondance est respectée autant à l'écrit qu'à l'oral. Des modifications apportées à `autoBeaming` par une définition de contexte au sein d'un bloc `\layout` ne seront pas reportées dans le bloc `\midi` correspondant ; paroles et musique ne seront alors plus synchrones dans le fichier Midi.

Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`, `ly/performer-init.ly`.

Score – le père de tous les contextes

Il s'agit en l'occurrence du contexte le plus élevé, autrement dit le plus important, en matière de notation. En effet, c'est au niveau de la partition – *score* en anglais – que se gèrent le temps et la tonalité ; c'est donc là qu'il faut s'assurer que les différents éléments, tels les clefs, métriques et armures sont bien répercutés sur toutes les portées.

Dès lors que LilyPond rencontre un bloc `\score {...}` se crée implicitement un contexte `Score`.

Contextes de haut niveau – les systèmes

De nombreuses partitions sont écrites sur plus d'une portée. Ces portées peuvent être regroupées de différentes manières.

StaffGroup

Le groupe de portées est attaché par un crochet et les barres de mesure sont d'un seul tenant, de la première à la dernière portée. Le `StaffGroup` constitue le regroupement le plus simple.

ChoirStaff

Ce regroupement est identique au `StaffGroup`, à ceci près que les barres de mesure ne traversent pas l'espace inter-portées.

GrandStaff

Le groupe de portées est attaché par une accolade sur la gauche et les barres de mesure sont d'un seul tenant.

PianoStaff

Ce regroupement est identique au `GrandStaff`, à ceci près que le nom de l'instrument sera directement attaché au système.

Contextes de niveau intermédiaire – les portées

Staff

La portée prend en charge les clefs, barres de mesure, armures et les altérations accidentelles. Un contexte `Staff` peut contenir plusieurs contextes `Voice`.

RhythmicStaff

De même nature qu'un `Staff`, ce contexte est destiné à n'imprimer que du rythme. Quelle que soit la hauteur, les notes seront imprimées sur une même et unique ligne ; la sortie MIDI rendra les hauteurs saisies.

TabStaff

Ce contexte permet de générer des tablatures. La mise en forme par défaut correspond à une tablature pour guitare, sur six lignes.

DrumStaff

Contexte dévolu tout spécialement aux parties de percussion ; il peut contenir plusieurs `DrumVoice`.

VaticanaStaff

Identique au contexte `Staff`, à ceci près qu'il est tout particulièrement adapté au grégorien.

MensuralStaff

Identique au contexte `Staff`, à ceci près qu'il est tout particulièrement adapté au style mensural de musique ancienne.

Contextes de bas niveau – les voix

Les contextes de niveau « voix » initialisent un certain nombre de propriétés et activent les graveurs appropriés. Un contexte de bas niveau est un contexte n'ayant aucun contexte enfant – ou `defaultchild`. Bien qu'ils puissent accepter ou contenir des sous-contextes, ceux-ci devront être libellés et créés explicitement.

Voice

Correspond à une voix positionnée sur une portée. Le contexte `Voice` s'occupe des indications de nuance, des hampes, des ligatures, des scripts placés au-dessus ou au-dessous de la portée, des différentes liaisons et des silences. Lorsque plusieurs voix doivent cohabiter sur la même portée, il est indispensable de les instancier explicitement.

VaticanaVoice

Fonctionnant comme le contexte `Voice`, il est tout particulièrement destiné à gérer le grégorien.

MensuralVoice

Fonctionnant comme le contexte `Voice`, il est tout particulièrement adapté aux musiques anciennes.

Lyrics

Correspond à une voix contenant des paroles. Le contexte `Lyrics` gère l'impression d'une ligne de paroles.

DrumVoice

Contexte de voix dévolu à une portée de percussions.

FiguredBass

Contexte prenant en charge les objets `BassFigure` – la basse chiffrée – créés à partir de ce qui a été saisi en mode `\figuremode`.

TabVoice

Contexte de voix dévolu au contexte `TabStaff`, il est habituellement créé implicitement.

CueVoice

Contexte de voix utilisé essentiellement dans le cadre de citations ajoutées à une portée – voir [Mise en forme d'une citation], page 224, –, il est habituellement créé implicitement.

ChordNames

Permet d'imprimer des noms d'accord.

5.1.2 Création et référencement d'un contexte

LilyPond crée automatiquement des contextes de bas niveau lorsque l'expression musicale intervient avant qu'un contexte adéquat n'existe, ce qui peut être pratique dans le cadre d'une partition simple ou de courts fragments tels ceux inclus dans cette documentation. Dès que la

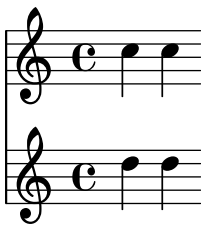
structure s'étoffe, il devient nécessaire de créer explicitement tous les contextes, à l'aide des commandes `\new` ou `\context`. Leur syntaxe est très similaire :

`[\new | \context]` *`Contexte`* `[ =` *`nom`*`] [`*`musique`*`]`

où peuvent intervenir aussi bien `\new` que `\context`. Le *Contexte* est le nom du contexte à créer, qui éventuellement s'appellera plus particulièrement *nom* ; il contient l'expression musicale unique *musique* qui devra être interprétée dans ce contexte par les graveurs ou exécutants.
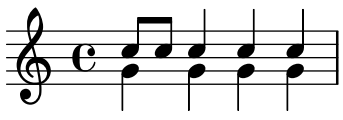
Le préfixe `\new` non suivi d'un nom s'utilise principalement pour créer une partition avec plusieurs portées :

```
<<
  \new Staff \relative {
    % leave the Voice context to be created implicitly
    c''4 c
  }
  \new Staff \relative {
    d''4 d
}
>>
```



et pour regrouper des voix sur une même portée :

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    c''8 c c4 c c
  }
  \new Voice \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```
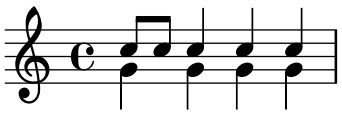


`\new` est à priviliégier lorsque les contextes ne sont pas nommés.

La différence entre les commandes `\new` et `\context` se situe au niveau de leurs effets :

- La commande `\new`, suivie ou non d'un nom, crée un tout nouveau contexte même s'il en existe déjà un portant le même nom :

```
\new Staff <<
  \new Voice = "A" \relative {
    \voiceOne
    c''8 c c4 c c
  }
```

```
  \new Voice = "A" \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```
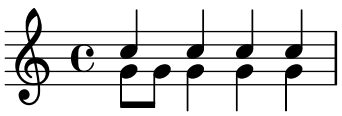


- La commande `\context` avec nommage créera un contexte distinct uniquement dans le cas où ne préexiste aucun contexte du même nom dans la même hiérarchie de contextes. Dans le cas contraire, il servira de référence au contexte précédemment créé, et son expression musicale sera transmise dans ce contexte pour interprétation.

  Le nommage des contextes se révèle fort utile dans le cadre des paroles ou de la basse chiffrée comme indiqué dans [Travail avec des paroles et variables], page 300, ou Section “Modèles pour ensemble vocal” dans *Manuel d'initiation* pour le premier cas et [Gravure de la basse chiffrée], page 462, pour le second. Par ailleurs, cette procédure est tout à fait pertinente lorsque l'on sépare mise en forme de la partition et contenu musical. Les deux formulations ci-après sont tout à fait valides :

```
\score {
  <<
    % score layout
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>
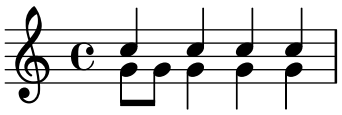
    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



```
\score {
  <<
```

```
    % score layout
    \new Staff <<
      \context Voice = "one" {
        \voiceOne
      }
      \context Voice = "two" {
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



Par ailleurs, le recours à des variables produira les mêmes effets – voir Section “Organisation du code source avec des variables” dans *Manuel d'initiation*.

- La commande `\context` utilisée sans nommage recherchera le premier de tous les contextes du même type précédemment créés dans la même hiérarchie de contextes ; l'expression musicale lui sera alors transmise pour interprétation. Bien que rarement utilisée, cette formulation de `\context` sans nommage ni expression musicale permet de définir le contexte dans lequel une procédure Scheme comportant une clause `\applyContext` devra s'exécuter.

```
\new Staff \relative {
  c'1
  \context Timing
  \applyContext #(lambda (ctx)
                   (newline)
                   (display (ly:context-current-moment ctx)))
  c1
}
```

Un contexte auquel il est ultérieurement fait référence doit impérativement être nommé. C'est le cas par exemple lorsque des paroles sont associées à de la musique :

```
\new Voice = "tenor" musique
...
\new Lyrics \lyricsto "tenor" paroles
```

L'association de paroles à de la musique est abordée en détails à la rubrique [Durée automatique des syllabes], page 291.

Les propriétés de tous les contextes d'un même type se modifient au sein d'un bloc `\layout`, selon une syntaxe différente – voir [Modification de tous les contextes d'un même type], page 639. Une telle construction permet de séparer mise en forme et contenu musical. Lorsque un seul contexte requiert une adaptation, mieux vaut recourir à un bloc `\with` – voir [Modification d'un contexte particulier], page 641.

Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : [Durée automatique des syllabes], page 291, [Modification d'un contexte particulier], page 641.

5.1.3 Conservation d'un contexte

En règle générale, un contexte disparaît dès qu'il n'y a plus rien à faire. Autrement dit, un contexte `Voice` disparaît dès après le dernier événement qu'il contient, et un contexte `Staff` dès que les contextes `Voice` qu'il supporte ne contiennent plus rien. Ceci peut avoir des conséquences néfastes lorsqu'il est fait référence à un contexte alors disparu, comme dans le cas d'un changement de portée introduit par la commande `\change`, l'association de paroles à l'aide de la commande `\lyricsto` ou si des événements surviennent à nouveau pour ce contexte précédemment actif.

Une exception cependant à cette règle : en présence d'un contexte `Staff` ou dans une construction `<< ... >>`, un seul des contextes `Voice` inclus restera actif jusqu'à la fin du contexte `Staff` ou de la construction `<< ... >>`, y compris s'il y a des « trous ». Le contexte alors persistant sera le premier rencontré dans la construction `{ ... }` sans tenir compte des éventuels `<< ... >>` qu'elle pourrait contenir.

Un contexte restera actif dès lors qu'il s'y passera toujours quelque chose. Un contexte `Staff` restera actif si l'une des voix qu'il supporte est toujours active. L'un des moyens de s'en assurer consiste à ajouter des silences invisibles parallèlement à la musique. Vous devrez les ajouter dans tous les contextes `Voice` qui doivent rester actifs. Nous vous conseillons, lorsque plusieurs voix interviennent de manière sporadique, de toutes les maintenir actives plutôt que de vous fier aux exceptions mentionnées plus haut.
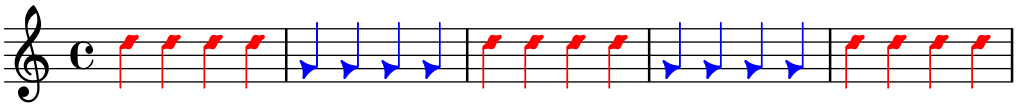
Dans l'exemple suivant, les deux voix A et B sont maintenues actives jusqu'à la fin du morceau :

```
musicA = \relative { d''4 d d d }
musicB = \relative { g'4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 }  % Keep Voice "A" alive for 5 bars
    \new Voice = "B" { s1*5 }  % Keep Voice "B" alive for 5 bars
  >>
}

music = {
  \context Voice = "A" {
    \voiceOneStyle
    \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle
    \musicB
```

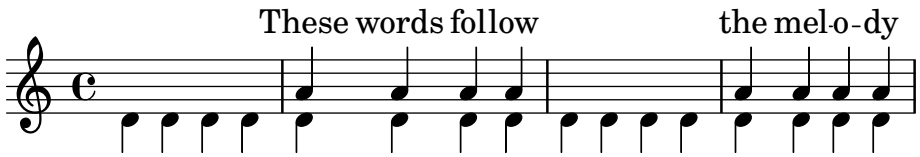
```
  }
  \context Voice = "A" { \musicA }
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



L'exemple suivant illustre la manière d'écrire selon cette méthode une mélodie discontinue à laquelle se rattachent des paroles. Dans la réalité, mélodie et accompagnement feraient l'objet de portées séparées.

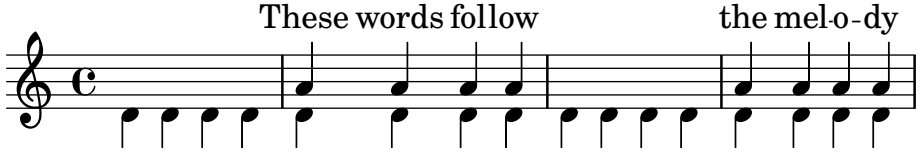
```
melody = \relative { a'4 a a a }
accompaniment = \relative { d'4 d d d }
words = \lyricmode { These words fol -- low the mel -- o -- dy }
\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          s1*4  % Keep Voice "melody" alive for 4 bars
        }
        {
          \new Voice = "accompaniment" {
            \voiceTwo
            \accompaniment
          }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
          \context Voice = "accompaniment" { \accompaniment }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "music" }
    \lyricsto "melody" { \words }
  >>
}
```



Une autre méthode, qui s'avère plus productive dans nombre de cas, consiste à maintenir active la ligne mélodique en y insérant des silences invisibles tout au long de l'accompagnement :

```
melody = \relative {
  s1  % skip a bar
  a'4 a a a
  s1  % skip a bar
  a4 a a a
}
accompaniment = \relative {
  d'4 d d d
  d4 d d d
  d4 d d d
  d4 d d d
}
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          \melody
        }
        \new Voice = "accompaniment" {
          \voiceTwo
          \accompaniment
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "music" }
    \lyricsto "melody" { \words }
  >>
}
```



5.1.4 Modification des greffons de contexte

Les contextes, tels que `Score` ou `Staff`, ne contiennent pas que des propriétés ; ils mettent également en œuvre certains sous-programmes (*plug-ins* pour employer le terme consacré) nommés « graveurs » (*engravers* pour reprendre le terme anglais). Ces sous-programmes sont chargés de créer les différents éléments de notation : on trouve ainsi dans le contexte `Voice` un graveur `Note_heads_engraver`, chargé des têtes de notes et, dans le contexte `Staff`, un graveur `Key_engraver`, chargé de l'armure.

Vous trouverez une description exhaustive de chaque graveur dans Référence des propriétés internes ↦ Translation ↦ Engravers. Chaque contexte mentionné dans Référence des propriétés internes ↦ Translation ↦ Context. répertorie les graveurs mis en œuvre.
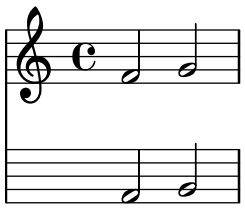
On peut faire, au moyen de ces graveurs, sa propre « cuisine », en modifiant les contextes à volonté.

Lorsqu'un contexte est créé, par la commande `\new` ou `\context`, on peut y adjoindre un bloc `\with` (en anglais « avec »), dans lequel il est possible d'ajouter (commande `\consists`) ou d'enlever (commande `\remove`) des graveurs :

```
\new contexte \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ...musique...
}
```

Ici chacun des points de suspension ... devront être remplacés par le nom des graveurs désirés. Dans l'exemple suivant, on enlève du contexte `Staff` la métrique (graveur `Time_signature_engraver`) et la clef (graveur `Clef_engraver`).

```
<<
  \new Staff \relative {
    f'2 g
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } \relative {
    f'2 g2
  }
>>
```

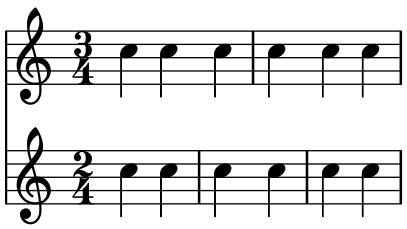


La clef et le chiffre de mesure ont disparu de la deuxième portée. C'est une méthode quelque peu radicale puisqu'elle affectera toute la portée jusqu'à la fin de la partition. L'espacement s'en trouve également affecté, ce qui peut être ou non l'effet recherché. Une méthode plus sophistiquée aurait été de rendre ces objets transparents (voir Section "Visibilité et couleur des objets" dans *Manuel d'initiation*).

Dans l'exemple suivant, voici une mise en pratique plus utile. En temps normal, les barres de mesure et la métrique sont synchronisées verticalement dans toute la partition. Les graveurs qui en sont responsables se nomment `Timing_translator` et `Default_bar_line_engraver`. En les enlevant du contexte `Score` pour les attribuer au contexte `Staff`, chaque portée peut désormais avoir sa propre métrique.

```
\score {
```

```
<<
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \relative {
      \time 3/4
      c''4 c c c c c
  }
\new Staff \with {
  \consists "Timing_translator"
  \consists "Default_bar_line_engraver"
}
\relative {
    \time 2/4
    c''4 c c c c c
}
>>
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    }
  }
}
```



Problèmes connus et avertissements

L'ordre dans lequel les graveurs sont spécifiés correspond à leur ordre d'apparition dans le processus d'élaboration de la partition. En règle générale, l'ordre dans lequel les graveurs sont mentionnés importe peu. Il se peut toutefois qu'un graveur écrive une propriété qui sera interprétée par un autre, ou qu'un graveur crée un objet graphique qui sera traité par un autre ; l'ordre d'apparition de ces graveurs prendra alors toute son importance.

Pour information, les ordonnancements suivants sont importants :

- le `Bar_engraver` devrait toujours être le premier ;
- le `New_fingering_engraver` doit toujours précéder le `Script_column_engraver` ;
- le `Timing_translator` doit toujours précéder le `Bar_number_engraver`.

Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`.

5.1.5 Modification des réglages par défaut d'un contexte

Les propriétés des contextes et objets graphiques se modifient à l'aide des commandes `\set` et `\override`, comme indiqué à la rubrique Section 5.3 [Modification de propriétés], page 651. Ces

commandes créent des événements musicaux qui feront que la modification produira ses effets dès l'instant où la musique est traitée.

Le propos est ici de voir comment modifier les valeurs *par défaut* des propriétés de contexte ou d'objet graphique dès la création de ces contextes. Deux manières de procéder sont envisageables : l'une consiste à modifier les valeurs pour tous les contextes d'un même type, l'autre s'attache à adapter les valeurs par défaut d'une instance particulière d'un contexte.

Modification de tous les contextes d'un même type

L'adaptation des réglages par défaut d'un contexte, qu'il s'agisse de `Score`, `Staff` ou `Voice`, peut se réaliser indépendamment de la musique dans un bloc `\layout` – placé dans le bloc `\score` auquel ces modifications doivent s'appliquer – au moyen d'un bloc `\context`.

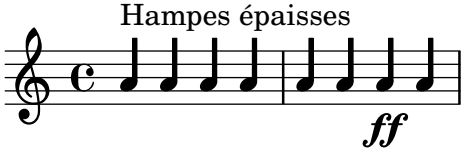
Les réglages dévolus à la sortie MIDI viendront quant à eux se placer dans un bloc `\midi` – voir [Définitions de la sortie – hiérarchie des contextes], page 628.

```
\layout {
  \context {
    \Voice
    [réglage de contexte pour tous les contextes Voice]
  }
  \context {
    \Staff
    [réglage de contexte pour tous les contextes Staff]
  }
}
```

La spécification des adaptations peut se faire de différentes manières :

- à l'aide d'une commande `\override`, sans lui adjoindre le nom du contexte :
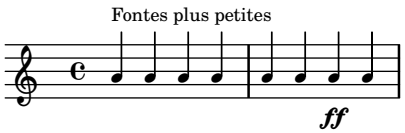
```
\score {
  \relative {
    a'4^"Hampes épaisses" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      \override Stem.thickness = #4.0
    }
  }
}
```



- en définissant directement une propriété de contexte :

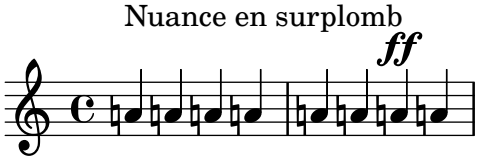
```
\score {
  \relative {
    a'4^"Fontes plus petites" a a a
    a4 a a\ff a
  }
  \layout {
```

```
    \context {
      \Staff
      fontSize = #-4
    }
  }
}
```



- à l'aide d'une commande prédéfinie comme `\dynamicUp`, ou bien une expression musicale telle que `\accidentalStyle dodecaphonic` :

```
\score {
  \relative {
    a'4^"Nuance en surplomb" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}
```

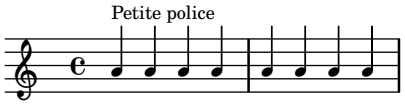


- à l'aide d'une variable personnalisée contenant un bloc `\with` ; pour de plus amples informations sur le bloc `\with`, voir [Modification d'un contexte particulier], page 641.

```
StaffDefauts = \with {
  fontSize = #-4
}
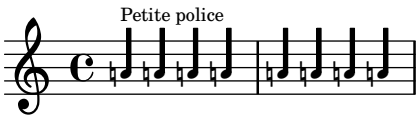
\score {
  \new Staff {
    \relative {
      a'4^"Petite police" a a a
      a4 a a a
    }
  }
  \layout {
    \context {
      \Staff
```

```
        \StaffDefauts
      }
    }
  }
```



Les instructions destinées à modifier les propriétés peuvent se placer dans un bloc `\layout` sans pour autant être incluses dans un bloc `\context`. Expliciter des réglages de la sorte équivaut à inclure les commandes de modification des propriétés au début de chacun des contextes du type en question. Lorsque le contexte n'est pas spécifié, *tous* les contextes de bas niveau seront affectés – voir [Contextes de bas niveau – les voix], page 630. La syntaxe appropriée répond aux mêmes critères que si la commande était écrite dans le flot musical.

```
\score {
  \new Staff {
    \relative {
      a'4^"Petite police" a a a
      a4 a a a
    }
  }
  \layout {
    \accidentalStyle dodecaphonic
    \set fontSize = #-4
    \override Voice.Stem.thickness = #4.0
  }
}
```



Modification d'un contexte particulier

Dans le cas d'un contexte pris individuellement, ses propriétés se modifient à l'aide d'un bloc `\with`. Toutes les autres instances de contexte appartenant au même type seront affectées des réglages prédéfinis par LilyPond, modifiés le cas échéant au sein d'un bloc `\layout`. Le bloc `\with` se place directement à la suite de la commande `\new` *type-de-contexte*.

```
\new Staff \with {
  [réglages pour ce contexte pris individuellement]
} {
  ...
}
```

De la même manière, si la musique est saisie à la suite d'une commande abrégée, telle que `\chords` au lieu de `\chordmode`, l'instruction `\with` viendra se placer juste après la commande spécifiant le mode :

```
\chords \with {
  [réglages pour ce contexte (implicite) pris individuellement]
} {
  ...
}
```
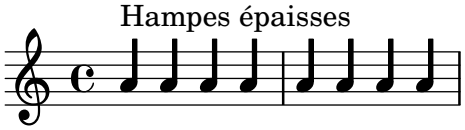
puisque c'est le contexte implicite alors créé qui devra être modifié. Cette manière de procéder s'applique à toutes les autres formes abrégées de spécification du mode de saisie (`\drums`, `\figures`) – voir Section 5.4.1 [Modes de saisie], page 666.

Dans la mesure où une telle « modification de contexte » est spécifiée au sein même de la musique, ses effets toucheront **toutes** les sorties (imprimable **et** Midi), contrairement à ce qui se passe lorsque les adaptations sont réalisées dans la définition d'une sortie.

La spécification des adaptations peut se faire de différentes manières :

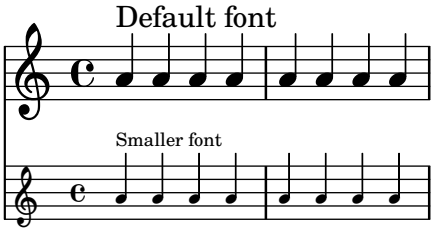
- à l'aide d'une commande `\override`, sans lui adjoindre le nom du contexte :

```
\score {
  \new Staff {
    \new Voice \with { \override Stem.thickness = #4.0 }
    {
      \relative {
        a'4^"Hampes épaisses" a a a
        a4 a a a
      }
    }
  }
}
```
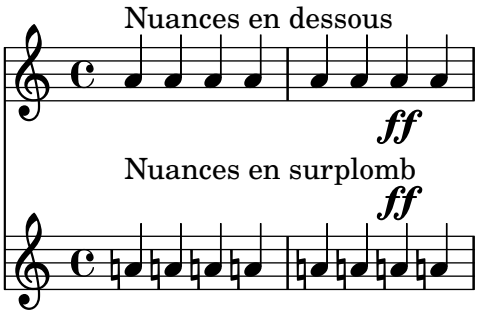


- en définissant directement une propriété de contexte :

```
\score {
  <<
    \new Staff {
      \relative {
        a'4^"Default font" a a a
        a4 a a a
      }
    }
    \new Staff \with { fontSize = #-4 }
    {
      \relative {
        a'4^"Smaller font" a a a
        a4 a a a
      }
    }
  >>
}
```



- à l'aide d'une commande prédéfinie comme `\dynamicUp` :

```
\score {
  <<
    \new Staff {
      \new Voice {
        \relative {
          a'4^"Nuances en dessous" a a a
          a4 a a\ff a
        }
      }
    }
    \new Staff \with { \accidentalStyle dodecaphonic }
    {
      \new Voice \with { \dynamicUp }
      {
        \relative {
          a'4^"Nuances en surplomb" a a a
          a4 a a\ff a
        }
      }
    }
  >>
}
```



Voir aussi

Manuel de notation : Section 5.4.1 [Modes de saisie], page 666.

Ordre de préséance

La valeur d'une propriété qui doit s'appliquer à un instant particulier est déterminée comme suit :

- s'il y a une instruction `\override` ou `\set` active dans le flot d'information, sa valeur s'applique,
- en l'absence de quoi sera utilisée la valeur par défaut telle que définie dans une clause `\with` stipulée à l'initialisation du contexte,
- en l'absence de quoi sera retenue la valeur par défaut issue du bloc `\context` approprié le plus récent dans les blocs `\layout` ou `\midi`,
- en l'absence de quoi s'appliqueront les réglages prédéfinis de LilyPond.

Voir aussi

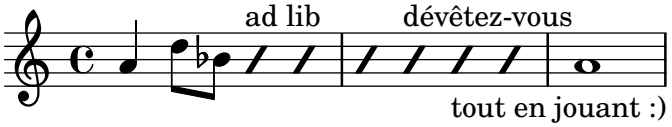
Manuel d'initiation : Section "Modification des propriétés d'un contexte" dans *Manuel d'initiation*.

Manuel de notation : [Contextes de bas niveau – les voix], page 630, Section 5.3.3 [La commande de dérogation `\override`], page 654, Section 5.3.2 [La commande de fixation `\set`], page 652, Section 4.2.1 [Le bloc `\layout`], page 583, Section 5.1.1 [Tout savoir sur les contextes], page 628.

5.1.6 Définition de nouveaux contextes

Les contextes tels que `Staff` ou `Voice` sont faits de briques de construction empilées. En combinant divers graveurs, il est possible de créer de nouveaux types de contextes.

Dans l'exemple suivant on construit, de zéro, un nouveau contexte très semblable à `Voice`, mais qui n'imprime que des têtes de notes en forme de barre oblique au centre de la portée. Un tel contexte peut servir, par exemple, à indiquer un passage improvisé dans un morceau de jazz.



On a rassemblé les réglages dans un bloc `\context`, lui-même placé dans le bloc `\layout` :

```
\layout {
  \context {
    ...
  }
}
```

En lieu et place des points (. . .), voici les éléments à saisir :

Tout d'abord, il convient de donner un nom à notre nouveau contexte :

```
\name ImproVoice
```

Comme il est très semblable à un contexte `Voice`, nous souhaitons que toutes les commandes associées au `Voice` déjà existant restent valables. D'où nécessité de la commande `\alias`, qui va l'associer au contexte `Voice` :

```
\alias Voice
```

Ce contexte doit pouvoir imprimer des notes et des indications textuelles ; on ajoute donc les graveurs appropriés ainsi que celui dévolu au regroupement sous forme de colonne des notes, hampes et silences qui interviennent au même moment musical :

```
\consists "Note_heads_engraver"
\consists "Text_engraver"
\consists "Rhythmic_column_engraver"
```

Toutes les notes devraient s'afficher au centre de la portée :

```
\consists "Pitch_squash_engraver"
squashedPosition = #0
```

Le graveur `Pitch_squash_engraver` intercepte les notes créées par le `Note_heads_engraver`, et les « écrase » pour qu'elles aient toutes la même position verticale, définie par `squashedPosition` : ici il s'agit de la valeur `0`, c'est-à-dire la ligne du milieu.

On veut que les notes aient la forme d'une barre oblique, sans aucune hampe :

```
\override NoteHead.style = #'slash
\hide Stem
```

Tous ces modules doivent communiquer sous le contrôle du contexte. Les mécanismes permettant aux contextes de communiquer sont établis dès lors que le mot-clé `\type` précède le contexte. La plupart des contextes mentionnés au sein d'un bloc `\layout` seront de type `Engraver_group`. Certains contextes spécifiques, ainsi que ceux mentionnés dans les blocs `\midi`, reposent sur

d'autres types de contexte. Recopier un contexte préexistant pour en modifier la définition lui affecte le type adéquat. Dans la mesure où notre exemple consiste à créer une définition de toute pièce, son type doit être explicitement spécifié.

```
\type "Engraver_group"
```

Récapitulons ; on se retrouve avec le bloc suivant :

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Pitch_squash_engraver"
  squashedPosition = #0
  \override NoteHead.style = #'slash
  \hide Stem
  \alias Voice
}
```

Ce n'est pas tout. En effet, on veut intégrer le nouveau contexte `ImproVoice` dans la hiérarchie des contextes. Tout comme le contexte `Voice`, sa place est au sein du contexte `Staff`. Nous allons donc modifier la définition du contexte `Staff`, au moyen de la commande `\accepts` :

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Le contraire de `\accepts` est `\denies` ; il est parfois utile lorsque l'on recycle des définitions de contextes déjà existantes.

Enfin, tout cela doit prendre place dans le bloc `\layout`, comme ceci :

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
    \accepts "ImproVoice"
  }
}
```

On peut alors saisir la musique, comme dans l'exemple plus haut :

```
\relative {
  a'4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"dévêtez-vous"
    c c_"tout en jouant :)"
  }
  a1
}
```

Pour être tout à fait complet, les modifications apportée à la hiérarchie des contextes devraient être répétés au niveau du bloc `\midi` de telle sorte que la sortie Midi dépende des mêmes relations.

Voir aussi

Référence des propriétés internes : Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “Pitch_squash_engraver” dans *Référence des propriétés internes*, Section “Rhythmic_column_engraver” dans *Référence des propriétés internes*, Section “Text_engraver” dans *Référence des propriétés internes*.

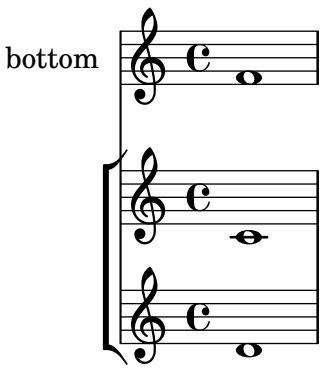
5.1.7 Ordonnancement des contextes

Les contextes viennent en principe se positionner selon leur ordre d’apparition dans le fichier source. Lorsque plusieurs contextes sont imbriqués, le contexte englobant supportera les différents contextes mentionnés dans le fichier source, à la stricte condition qu’ils soient dûment « agréés ». Les contextes imbriqués qui ne font pas partie de la « liste d’agréments » du contexte englobant se retrouveront en dessous de celui-ci au lieu d’y être imbriqués.

La liste des « agréments » d’un contexte se gère à l’aide des instructions `\accepts` ou `\denies` – `\accepts` pour ajouter un contexte à la liste, `\denies` pour retirer l’agrément.

Par exemple, on ne trouve normalement pas de portées regroupées par un crochet au sein d’un groupe matérialisé par une accolade et des barres d’un seul tenant ; un `GrandStaff` n’accepte donc pas, par défaut, d’englober un `StaffGroup`.

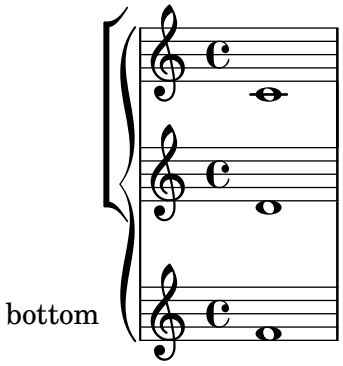
```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
}
```



Néanmoins, et grâce à une instruction `\accepts`, un `StaffGroup` peut se voir ajouté au contexte `GrandStaff` :

```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
  \layout {
    \context {
```

```
      \GrandStaff
      \accepts "StaffGroup"
    }
  }
}
```



L'instruction `\denies` permet, lorsqu'un nouveau contexte reprend les définitions d'un contexte existant, d'en ajuster les composantes. C'est par exemple le cas du contexte `VaticanaStaff`, réplique du contexte `Staff`, au sein duquel le contexte `VaticanaVoice` se substitue au contexte `Voice` dans la « liste d'agrément ».

Gardez à l'esprit que, face à une instruction qui ne s'appliquerait à aucun contexte déjà existant, LilyPond créera un nouveau contexte implicite.

Lors de la définition d'un contexte, les types de contextes sous-jacents susceptibles d'être créés implicitement sont spécifiés à l'aide d'une commande `\defaultchild`. Un certain nombre d'événements musicaux requièrent un contexte de plus bas niveau ; face à un tel événement, LilyPond crée autant de « sous-contextes » que nécessaire, jusqu'au contexte ne comportant aucun *defaultchild*.

La création implicite de contexte peut donc finir par engendrer de manière intempestive une nouvelle portée ou une autre partition. L'utilisation d'une instruction `\new` pour créer explicitement des contextes permet d'éviter ces problèmes.

Il arrive qu'un contexte ne doive exister que pendant un court instant, ce qui est le cas par exemple pour une *ossia*. Le plus simple consiste alors à initialiser la définition d'un contexte à l'endroit approprié, en parallèle avec le fragment correspondant dans la musique principale. Ce contexte temporaire sera par défaut positionné sous les autres contextes existants. Le repositionner au-dessus du contexte « principal » demande de le définir ainsi :

```
\new Staff \with { alignAboveContext = "principal" }
```

Il en va de même pour les contextes temporaires de paroles au sein d'un système à plusieurs portées comme un `ChoirStaff` lorsque, par exemple, un couplet supplémentaire apparaît à l'occasion d'une reprise. Ce contexte de paroles temporaire se place par défaut sous les portées inférieures. Lui adjoindre une instruction `alignBelowContext` dès son initialisation permet de l'accoler au contexte de paroles (nommé) qui contient le premier couplet.

Des exemples de repositionnement de contexte temporaire sont disponibles aux rubriques Section "Expressions musicales imbriquées" dans *Manuel d'initiation*, Section 1.6.2 [Modification de portées individuelles], page 205, et Section 2.1.2 [Situations particulières en matière de paroles], page 300.

Voir aussi

Manuel d'initiation : Section "Expressions musicales imbriquées" dans *Manuel d'initiation*.

Manuel de notation : Section 1.6.2 [Modification de portées individuelles], page 205, Section 2.1.2 [Situations particulières en matière de paroles], page 300.

Manuel d'utilisation : Voir Section “Apparition d'une portée supplémentaire” dans *Utilisation des programmes*.
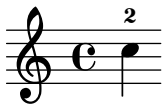
Fichiers d'initialisation : `ly/engraver-init.ly`.

5.2 En quoi consiste la référence des propriétés internes

5.2.1 Navigation dans les références du programme

Comment, par exemple, déplacer le doigté dans le fragment suivant ?

```
c''-2
```



Sur la page de la documentation relative aux doigtés, c'est-à-dire [Doigtés], page 236, se trouve l'indication suivante :

> **Voir aussi**
>
> Référence des propriétés internes : Section “Fingering” dans *Référence des propriétés internes*.

Ladite référence est disponible au format HTML, ce qui rend la navigation bien plus aisée. Il est possible soit de la lire en ligne, soit de la télécharger dans ce format. La démarche présentée ici sera plus difficile à comprendre dans un document au format PDF.

Suivons le lien Section “Fingering” dans *Référence des propriétés internes*. En haut de la nouvelle page, on peut lire

> Fingering objects are created by: Section “Fingering_engraver” dans *Référence des propriétés internes* and Section “New_fingering_engraver” dans *Référence des propriétés internes*.

En d'autres termes, *Les indications de doigtés* (`Fingering` en anglais) *sont créées par les graveurs Section “Fingering_engraver” dans Référence des propriétés internes et Section “New_fingering_engraver” dans Référence des propriétés internes.*

En suivant derechef les liens propres à la référence du programme, on suit en fait le cheminement qui aboutit à la création de la partition :

- Section “Fingering” dans *Référence des propriétés internes*: Section “Fingering” dans *Référence des propriétés internes* objects are created by: Section “Fingering_engraver” dans *Référence des propriétés internes*
- Section “Fingering_engraver” dans *Référence des propriétés internes*: Music types accepted: Section “fingering-event” dans *Référence des propriétés internes*
- Section “fingering-event” dans *Référence des propriétés internes*: Music event type `fingering-event` is in Music expressions named Section “FingeringEvent” dans *Référence des propriétés internes*

Ce cheminement se produit, bien sûr, en sens inverse : nous sommes ici partis du résultat, et avons abouti aux événements (en anglais *Events*) engendrés par le fichier d'entrée. L'inverse est également possible : on peut partir d'un événement et suivre le cheminement de LilyPond qui aboutit à la création d'un ou plusieurs objets graphiques.

La référence des propriétés internes peut également se parcourir comme un document normal. On y trouve des chapitres tels que `Music definitions` Section “Translation” dans *Référence des*

propriétés internes, ou encore Section “Backend” dans *Référence des propriétés internes*. Chaque chapitre recense toutes les définitions employées, et les propriétés sujettes à ajustements.

La Référence des propriétés internes n’est pas traduite en français – notamment du fait qu’elle est en évolution constante, tout comme LilyPond. En revanche, les termes musicaux font l’objet d’un Section “glossaire” dans *Glossaire* fort utile pour les utilisateurs francophones.

5.2.2 Interfaces de rendu

Tous les éléments de notation sont considérés comme des objets graphiques (en anglais *Graphical Object*, d’où le diminutif *Grob*). Chaque objet est doté d’un certain nombre de propriétés (l’épaisseur du trait, l’orientation, etc.), et lié à d’autres objets. Le fonctionnement de ces objets est décrit en détail dans Section “grob-interface” dans *Référence des propriétés internes*.

Prenons l’exemple des doigtés (en anglais *Fingering*). La page `Fingering` de la Référence des propriétés internes établit une liste de définitions propres à ce type d’objet :

> `padding` (dimension, in staff space):
>
> `0.5`

Ce qui signifie que les doigtés doivent être maintenus à une certaine distance (*padding*) des notes : 0,5 unités *staff-space* (espace de portée).

Chaque objet peut avoir plusieurs attributs, en tant qu’élément typographique ou musical. Ainsi, un doigté (objet *Fingering*) possède les attributs suivants :

- Sa taille ne dépend pas de l’espacement horizontal, contrairement aux liaisons ou ligatures.
- C’est du texte – un texte vraiment court, certes.
- Ce texte est imprimé au moyen d’une fonte, contrairement aux liaisons ou ligatures.
- Sur l’axe horizontal, le centre de ce symbole doit être aligné avec le centre de la note.
- Sur l’axe vertical, le symbole doit être proche de la note et de la portée.
- Sur l’axe vertical encore, il doit également s’ordonner avec les éventuels autres symboles, ponctuations ou éléments textuels.

Faire appliquer ces différents attributs est le rôle des *interfaces*, que l’on trouve en bas de la page Section “Fingering” dans *Référence des propriétés internes*.

> This object supports the following interfaces: Section “item-interface” dans *Référence des propriétés internes*, Section “self-alignment-interface” dans *Référence des propriétés internes*, Section “side-position-interface” dans *Référence des propriétés internes*, Section “text-interface” dans *Référence des propriétés internes*, Section “text-script-interface” dans *Référence des propriétés internes*, Section “font-interface” dans *Référence des propriétés internes*, Section “finger-interface” dans *Référence des propriétés internes*, and Section “grob-interface” dans *Référence des propriétés internes*.

En français,

> Cet objet admet les interfaces suivantes :

Suit la liste des interfaces en question, présentées comme autant de liens qui conduisent aux pages dédiées à chacune d’entre elles. Chaque interface est dotée d’un certain nombre de propriétés, dont certaines peuvent être modifiées, d’autres non (les *Internal properties*, ou propriétés internes).

Pour aller encore plus loin, plutôt que de simplement parler de l’objet `Fingering`, ce qui ne nous avance pas à grand chose, on peut aller explorer son âme même, dans les fichiers source de LilyPond (voir Section “Autres sources de documentation” dans *Manuel d’initiation*), en l’occurrence le fichier `scm/define-grobs.scm` :

```
(Fingering
```
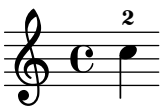
```
. ((padding . 0.5)
   (avoid-slur . around)
   (slur-padding . 0.2)
   (staff-padding . 0.5)
   (self-alignment-X . 0)
   (self-alignment-Y . 0)
   (script-priority . 100)
   (stencil . ,ly:text-interface::print)
   (direction . ,ly:script-interface::calc-direction)
   (font-encoding . fetaText)
   (font-size . -5)  ; don't overlap when next to heads.
   (meta . ((class . Item)
   (interfaces . (finger-interface
                  font-interface
                  text-script-interface
                  text-interface
                  side-position-interface
                  self-alignment-interface
                  item-interface))))))
```

. . . où l'on découvre que l'objet `Fingering` n'est rien de plus qu'un amas de variables et de réglages. La page de la Référence des propriétés internes est en fait directement engendrée par cette définition.

5.2.3 Détermination de la propriété d'un objet graphique (grob)

Nous voulions changer la position du chiffre **2** dans le fragment suivant :

```
c''-2
```



Dans la mesure où le **2** est placé, verticalement, à proximité de la note qui lui correspond, nous allons devoir trouver l'interface en charge de ce placement, qui se trouve être `side-position-interface`. Sur la page de cette interface, on peut lire :

> `side-position-interface`
>
> Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Ce qui signifie

> `side-position-interface`
>
> Placer l'objet affecté à proximité d'autres objets. La propriété `direction` indique où positionner l'objet (à droite ou à gauche, en haut ou en bas).

En dessous de cette description se trouve décrite la variable `padding` :

> `padding` (dimension, in staff space)
>
> Add this much extra space between objects that are next to each other.

Ce qui signifie

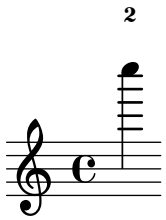
> Ajouter tel espace supplémentaire entre des objets proches les uns des autres.

En augmentant la valeur de `padding`, on peut donc éloigner le doigté de la note. La commande suivante insère trois unités d'espace vide entre la note et le doigté :

```
\once \override Voice.Fingering.padding = #3
```

En ajoutant ce tampon avant la création du doigté (de l'objet `Fingering`), donc avant `c2`, on obtient le résultat suivant :

```
\once \override Voice.Fingering.padding = #3
c''-2
```



Dans le cas présent, le réglage intervient dans le contexte `Voice`, ce qui pouvait également se déduire de la Référence des propriétés internes, où la page du graveur Section "Fingering_engraver" dans *Référence des propriétés internes* indique :

> Fingering_engraver is part of contexts: ... Section "Voice" dans *Référence des propriétés internes*

Ce qui signifie

> Le graveur Fingering_engraver fait partie des contextes : ... Section "Voice" dans *Référence des propriétés internes*

5.2.4 Conventions de nommage

Afin de s'y retrouver plus aisément et d'éviter les erreurs de frappe, voici quelques conventions en matière de nommage :

- fonctions scheme : minuscule-avec-trait-d-union (ce qui inclut les noms en mot-unique)
- fonctions scheme : ly:plus-style-scheme
- événements, classes et propriétés musicaux : identique-aux-fonctions-scheme
- interfaces d'objet graphique : style-scheme
- propriétés d'arrière plan : style-scheme (mais X et Y pour les axes)
- contextes (ainsi que MusicExpressions et grobs) : Capitale initiale ou Camélisation (CamelCase)
- propriétés de contexte : minusculeSuivieDeCamélisation
- graveurs : Capitale_initiale_puis_minuscules_séparées_par_un_souligné

Les questions que vous devez vous poser sont :

- Qu'est-ce qui relève des conventions, et qu'est-ce qui relève de la règle ?
- Qu'est-ce qui relève des règles du langage sous-jacent, et qu'est-ce qui est propre à LilyPond ?

5.3 Modification de propriétés

5.3.1 Vue d'ensemble de la modification des propriétés

Chaque contexte est chargé de créer plusieurs types d'objets graphiques. Il contient également les réglages nécessaires pour chacun de ces objets. Si l'on modifie ces réglages, les objets n'auront plus la même apparence.

Les contextes comportent deux types différents de propriétés : des propriétés de contexte et des propriétés d'objet graphique. Les propriétés de contexte sont celles qui s'appliqueront globalement au contexte en tant que tel ; elles gèrent la manière dont le contexte apparaîtra. Les propriétés d'objet graphique, par contre, s'appliquent à des types particuliers d'objet qui apparaissent dans le contexte en question.

Les commandes `\set` et `\unset` permettent de modifier les valeurs des propriétés de contexte. Les commandes `\override` et `\revert` permettent de modifier les valeurs des propriétés des objets graphiques.

Voir aussi

Référence des propriétés internes : Section "All layout objects" dans *Référence des propriétés internes*, Section "Backend" dans *Référence des propriétés internes*, Section "OverrideProperty" dans *Référence des propriétés internes*, Section "PropertySet" dans *Référence des propriétés internes*, Section "RevertProperty" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La sous-couche Scheme ne vérifie pas la saisie des propriétés de façon très stricte. Des références cycliques dans des valeurs Scheme peuvent de ce fait interrompre ou faire planter le programme – ou bien les deux.

5.3.2 La commande de fixation `\set`

Chaque contexte peut avoir plusieurs **propriétés**, c'est-à-dire des variables qu'il inclut. Ces dernières peuvent être modifiées « à la volée », c'est-à-dire pendant que la compilation s'accomplit. C'est ici le rôle de la commande `\set`.

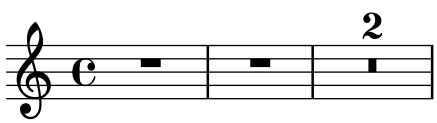
```
\set contexte.propriété = #valeur
```

Dans la mesure où *valeur* est constituée d'un objet Scheme, elle doit être précédée du caractère `#`.

Les propriétés des contextes se libellent sous la forme `minusculeMajuscule`. Leur rôle consiste principalement à traduire la musique en notation : par exemple, `localAlterations` déterminera quand imprimer une altération accidentelle, et `measurePosition` quand imprimer une barre de mesure. La valeur des propriétés des contextes peut évoluer au fur et à mesure que l'on avance dans le morceau – `measurePosition` en est l'illustration parfaite.

Ainsi la propriété de contexte `skipBars` permet de condenser les mesures vides de notes, en des silences multimesures – voir [Compression de mesures vides], page 230, à ce sujet. Il s'agit d'un objet Scheme, auquel on attribue la valeur booléenne « vrai », c'est-à-dire la lettre `#t` pour « True » en anglais :
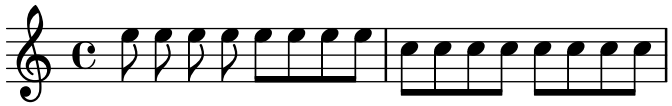
```
R1*2
\set Score.skipBars = ##t
R1*2
```



Si l'argument *contexte* n'est pas spécifié, alors la propriété cherchera à s'appliquer dans le contexte le plus restreint où elle est employée : le plus souvent `ChordNames`, `Voice` ou `Lyrics`.

```
\set Score.autoBeaming = ##f
```
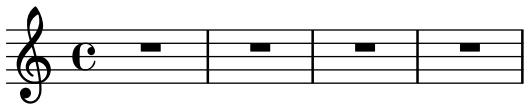
```
\relative {
  e''8 e e e
  \set autoBeaming = ##t
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```



Ce changement étant appliqué « à la volée », il n'affecte que le second groupe de notes.

Notez que le contexte le plus restreint n'est pas toujours le bon, et peut ne pas contenir la propriété qui vous intéresse : ainsi, la propriété `skipBars`, évoquée plus haut, ne relève pas du contexte `Voice`, mais du contexte `Score` – le code suivant ne fonctionnera pas.

```
R1*2
\set skipBars = ##t
R1*2
```
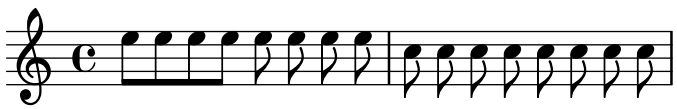


Les contextes s'organisent de façon hiérarchique : aussi, lorsqu'un contexte de niveau supérieur est spécifié (par exemple `Staff`), la propriété sera modifiée dans tous les contextes inférieurs (tous les contextes `Voice`, par exemple) qu'il contient.

La commande `\unset` permet d'annuler la définition d'une propriété :

`\unset` *`contexte.propriété`*

si et seulement si cette *propriété* a été définie dans ce *contexte* précis. En d'autres termes, la commande `\unset` doit impérativement affecter le même contexte que la commande `\set` d'origine, même en cas d'imbrication.

```
\set Score.autoBeaming = ##t
\relative {
  \unset autoBeaming
  e''8 e e e
  \unset Score.autoBeaming
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```



Si l'on se trouve dans le contexte le plus restreint, il n'est pas obligatoire, là encore, de spécifier le *contexte*. Ainsi, les deux lignes suivantes
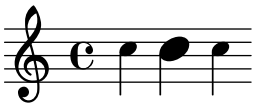
```
\set Voice.autoBeaming = ##t
```

```
\set autoBeaming = ##t
```

sont équivalentes si elles apparaissent dans un contexte `Voice`.

Pour modifier une propriété de façon à ce que l'accommodement ne s'applique qu'une seule fois, il convient d'adjoindre l'instruction `\once` à la commande `\set` ou `\unset` :

```
c''4
\once \set fontSize = #4.7
c''4
c''4
```



Ici le changement de taille est annulé aussitôt après la note concernée.

La référence des propriétés internes contient une description exhaustive de toutes les propriétés, contexte par contexte : voir Translation ↦ Tunable context properties.

Voir aussi

Manuel de notation : [Compression de mesures vides], page 230.

Référence des propriétés internes : Section "Tunable context properties" dans *Référence des propriétés internes*.

5.3.3 La commande de dérogation `\override`
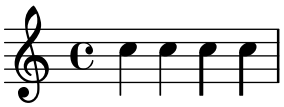
La commande `\override` permet de modifier la mise en forme des objets graphiques. Les descriptions d'objet graphique, dont les noms commencent par une majuscule, puis comprennent une ou plusieurs majuscules (de style `TotoTata`), contiennent les réglages « par défaut » pour les objets graphiques. Ces réglages sont sous forme de liste Scheme ; on peut les consulter dans le fichier `scm/define-grobs.scm`.

`\override` est en fait un raccourci :

```
\override [contexte.]NomObjet.propriété = #valeur
```

Nous pouvons donc par exemple accroître l'épaisseur des hampes en jouant sur la propriété `thickness` de l'objet `stem` :
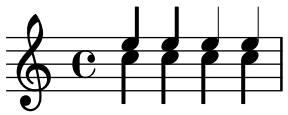
```
c''4 c''
\override Voice.Stem.thickness = #3.0
c''4 c''
```



Lorsqu'aucun contexte n'est spécifié dans une clause `\override`, celle-ci s'appliquera au contexte le plus bas :

```
\override Staff.Stem.thickness = #3.0
<<
  \relative {
    e''4 e
    \override Stem.thickness = #0.5
    e4 e
  } \\
  \relative {
```

```
    c''4 c c c
  }
>>
```



Certaines « sous-propriétés » sont parfois contenues dans une propriété. La commande devient alors :

```
\override Stem.details.beamed-lengths = #'(4 4 3)
```

ou, pour modifier les extrémités d'un objet à extension :

```
\override TextSpanner.bound-details.left.text = "texte de gauche"
\override TextSpanner.bound-details.right.text = "texte de droite"
```
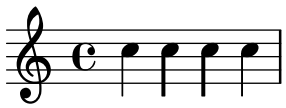
Les effets d'un `\override` prennent fin à l'aide de l'instruction `\revert`.

La syntaxe de la commande `\revert` command est :

```
\revert [contexte.]NomObjet.propriété
```
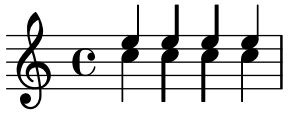
Par exemple :

```
\relative {
  c''4
  \override Voice.Stem.thickness = #3.0
  c4 c
  \revert Voice.Stem.thickness
  c4
}
```
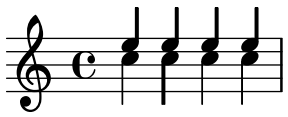


Les effets d'un `\override` ou d'un `\revert` s'appliquent dès l'endroit où ils apparaissent, et à tous les objets dans le contexte mentionné :

```
<<
  \relative {
    e''4
    \override Staff.Stem.thickness = #3.0
    e4 e e
  } \\
  \relative {
    c''4 c c
    \revert Staff.Stem.thickness
    c4
  }
>>
```



Les instructions `\override` et `\revert` doivent être précédées d'un `\once` dès lors que les effets de l'accommodement ne concernent que l'événement qui les suit directement :

```
<<
  \relative c {
    \override Stem.thickness = #3.0
    e''4 e e e
  } \\
  \relative {
    c''4
    \once \override Stem.thickness = #3.0
    c4 c c
  }
>>
```



Voir aussi

Référence des propriétés internes : Section "Backend" dans *Référence des propriétés internes*

5.3.4 La commande d'affinage `\tweak`

L'utilisation d'un `\override` pour modifier les propriétés d'un objet graphique affectera toutes les instances de l'objet en question au sein du contexte, et ce dès son apparition. Il peut parfois être préférable de n'affecter qu'un seul objet en particulier plutôt que tous les objets du contexte. C'est là le rôle de l'instruction `\tweak`, dont la syntaxe est :

```
\tweak [objet-de-rendu.]objet-propriété valeur
```

Mention de l'*objet-de-rendu* est optionnel. La commande `\tweak` s'applique à l'objet qui apparaît immédiatement après `valeur`.
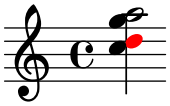
Pour une introduction à la syntaxe et l'utilisation des retouches, voir le chapitre Section "Méthodes de retouche" dans *Manuel d'initiation*.

Lorsque plusieurs éléments de même nature surviennent au même instant, il devient impossible d'utiliser l'instruction `\override` pour n'en modifier qu'un seul individuellement, d'où l'intérêt de la commande `\tweak`. Entre autres éléments qui sont susceptibles de se produire au même instant, nous citerons :

- les têtes de notes au sein d'un accord,
- les signes d'articulation,
- les liaisons de prolongation sur des notes d'un accord,
- les crochets de n-olets démarrant au même instant
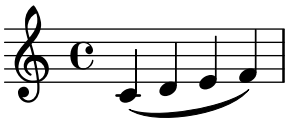
Dans l'exemple suivant, l'une des têtes de note de l'accord est colorisée, et l'aspect d'une autre est changé.

```
< c''
  \tweak color #red
  d''
  g''
  \tweak duration-log #1
  a''
> 4
```
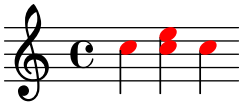


L'instruction `\tweak` permet aussi de modifier l'aspect d'une liaison :

```
\relative { c'-\tweak thickness #5 ( d e f) }
```



La commande `\tweak` ne sera pleinement fonctionnelle que si elle est directement rattachée à l'objet auquel elle doit s'appliquer alors que le fichier source est converti en flux musical. Vouloir modifier la globalité d'un accord est sans résultat dans la mesure où il ne constitue qu'un conteneur pour des événements musicaux et que tous les objets seront créés à partir d'événements appartenant à un `EventChord` (un événement d'accord) :
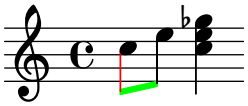
```
\tweak color #red c4
\tweak color #red <c e>4
<\tweak color #red c>4
```



La commande `\tweak` simple ne saurait servir à modifier un élément qui ne serait pas explicitement mentionné dans le fichier source. C'est notamment le cas des hampes, ligatures automatiques ou altérations, dans la mesure où elles seront ultérieurement générées et après les têtes de note (objets `NoteHead`), plutôt qu'au fil des éléments musicaux saisis.

De tels objets créés indirectement ne peuvent être ajustés que par une forme développée de la commande `\tweak`, autrement dit lorsque l'objet est explicitement mentionné :
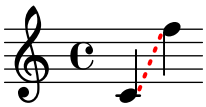
```
\tweak Stem.color #red
\tweak Beam.color #green c''8 e''
<c'' e'' \tweak Accidental.font-size #-3 ges''>4
```



La commande `\tweak` ne peut non plus servir à modifier clefs ou métriques, puisqu'elles seront immanquablement séparées du `\tweak` par l'insertion automatique d'autres éléments requis pour spécifier le contexte.

Plusieurs commandes `\tweak` en enfilade permettent d'affecter un même élément de notation :

```
c'
  -\tweak style #'dashed-line
  -\tweak dash-fraction #0.2
  -\tweak thickness #3
  -\tweak color #red
  \glissando
f''
```



Vous pouvez examiner le flux musical généré par une portion d'un fichier source, y compris les éléments automatiquement insérés, en suivant les indications portées à la rubrique Section "Affichage d'expressions musicales" dans *Extension de LilyPond*. Ceci s'avère tout à fait approprié pour déterminer ce qui peut se modifier à l'aide d'un `\tweak` ou bien aider à rectifier votre source de telle sorte que le `\tweak` produise ses effets.

Voir aussi

Manuel d'initiation : Section "Méthodes de retouche" dans *Manuel d'initiation*.

Manuel d'extension : Section "Affichage d'expressions musicales" dans *Extension de LilyPond*.

Problèmes connus et avertissements

Lorsqu'il y a plusieurs liaisons de prolongation dans un accord, la commande `\tweak` ne permet de modifier les points de contrôle que pour la première rencontrée dans le fichier source.

5.3.5 `\set` ou `\override`

Les instructions `\set` et `\override` manipulent toutes deux des propriétés associées à des contextes. Dans tous les cas, ces propriétés tiennent compte de la *hiérarchie des contextes* : les propriétés qui n'ont pas été définies dans le contexte lui-même héritent des valeurs de leur contexte parent respectif.

Les valeurs et durée de vie des propriétés d'un contexte sont dynamiques et ne sont accessibles qu'au moment où la musique est interprétée. Lors de la création d'un contexte, ses propriétés sont initialisées à partir de la définition du contexte correspondant et de ses éventuelles adaptations. Toute modification ultérieure ne sera obtenue que par des commandes d'adaptation des propriétés, libellées au sein même de la musique.

Les définitions d'un objet graphique (*graphical object* abrégé en *grob*) constituent une catégorie *spécifique* de propriétés de contexte, dans la mesure où leur structure, enregistrement et utilisation diffèrent des propriétés de contextes habituelles.

Contrairement aux propriétés de contextes habituelles, les définitions de *grob* sont subdivisées en propriétés de *grob*. Un *grob* est créé par un graveur lors de l'interprétation d'une expression musicale et reçoit ses propriétés initiales à partir de la définition de *grob* en cours dans le contexte du graveur. Le graveur (ou tout autre « agent » de LilyPond) peut alors ajouter ou modifier des propriétés à cet objet, sans pour autant affecter la définition du *grob* dans ce contexte.

Ce que LilyPond appelle « propriétés de *grob* » dans le cadre de l'affinage par l'utilisateur sont en fait les propriétés de la définition d'un objet dans un contexte.

Contrairement aux propriétés de contexte habituelles, les définitions d'un *grob* doivent être enregistrées pour pouvoir garder trace de ses composants, les propriétés individuelles du *grob* (ainsi que leurs sous-propriétés) séparément. Il sera dès lors possible de définir ces composants dans différents contextes et ainsi disposer d'une définition globale du *grob* à l'instant où la création de cet objet assemblera les éléments relatifs aux différents contextes attachés au contexte en cours et à ses parents.

Les définitions de *grob* se manipulent à l'aide des commandes `\override` et `\revert`, et leur nom commence par une capitale (comme '`NoteHead`') alors que les propriétés de contexte ordinaires – elles commencent par une minuscule – se manipulent avec `\set` et `\unset`.

Les instructions spéciales `\tweak` et `\overrideProperty` modifient les propriétés de *grob* en court-circuitant totalement les propriétés de contexte. En fait, elles capturent les *grobs* au moment de leur création pour y injecter directement des propriétés soit émanant d'un événement musical retouché par un `\tweak`, soit lorsqu'ils sont d'une qualité particulière (un `\overrideProperty`).

5.3.6 La commande de décalage `\offset`

Bien qu'il soit possible d'affecter de nouvelles valeurs aux propriétés d'un objet graphique à l'aide des commandes `\override`, `\tweak` ou `\overrideProperty`, il est souvent plus pratique de modifier de telles propriétés par rapport à une valeur par défaut. Ceci est la raison d'être de la commande `\offset`.

La commande `\offset` répond à la syntaxe suivante :

```
[-]\offset propriété décalages élément
```

La commande `\offset` agit par addition du contenu de *décalages* au réglage par défaut de la propriété *propriété* de l'objet graphique indiqué par *élément*.

Selon la manière dont la commande est formulée, `\offset` agira tantôt comme un `\tweak`, tantôt comme un `\override`. Les différences entre ces utilisations seront abordées après avoir recensé les propriétés qui peuvent être soumises à un `\offset`.

Propriétés acceptant des décalages

Bon nombre de propriétés d'objet graphique, mais pas toutes, peuvent faire l'objet d'un décalage. Si d'aventure *propriété* ne peut être affectée, l'objet restera inchangé et sera émis un message d'avertissement. En pareil cas, l'objet doit être modifié par un `\override` ou un `\tweak`.

Il est toujours possible de procéder à tâtons et laisser les avertissement indiquer si tel objet peut ou ne peut pas être soumis à `\offset`. Néanmoins, une approche plus systématique est possible.

Les critères énoncés ci-après déterminent l'égibilité d'une propriété à être modifiée par la commande `\offset`.

- La propriété possède un « réglage par défaut » au niveau de la définition de l'objet graphique. Les propriétés en question sont listées, pour chacun des *grobs*, dans Section "All layout objects" dans *Référence des propriétés internes* – on les trouvera aussi dans le fichier `scm/define-grobs.scm`.
- La propriété prend une valeur numérique. Les valeurs numériques comprennent `number`, liste de `number`s, `number-pair` et `number-pair-list`. Les pages de Section "All layout objects" dans *Référence des propriétés internes* répertorient le type de donnée propre à chaque propriété. Peu importe que le réglage par défaut soit une fonction.
- La propriété ne saurait constituer une « sous-propriété » – une propriété résidant au sein d'une autre propriété.
- Les propriétés réglées sur des valeurs infinies ne peuvent faire l'objet d'un décalage. Il n'y a aucun moyen d'influencer l'infini, qu'il soit positif ou négatif.

Les exemples qui suivent s'arrêtent sur plusieurs propriétés d'objet graphique au regard des critères énoncés ci-dessus.

- Propriétés qui peuvent être décalées

  `Hairpin.height`
  : Cette propriété n'est pas une sous-propriété, et est référencée à Section "Hairpin" dans *Référence des propriétés internes*. En tant que valeur, elle prend une « dimension, exprimée en espace de portée » réglée à `0.6666` – clairement une valeur `number` non infini.

  `Arpeggio.positions`
  : La page Section "Arpeggio" dans *Référence des propriétés internes* référence une propriété `positions` qui accepte une « paire de nombres ». Sa valeur par défaut est `ly:arpeggio::positions` – une fonction de rappel qui sera évaluée au cours de la phase de typographie pour donner une paire de nombres pour tout objet `Arpeggio`.

- Propriétés qui ne peuvent être décalées

  `Hairpin.color`
  : Aucune référence à `color` n'est mentionnée dans Section "Hairpin" dans *Référence des propriétés internes*.

  `Hairpin.circled-tip`
  : La référence à `Hairpin.circled-tip` dans Section "Hairpin" dans *Référence des propriétés internes* indique que cette propriété prend une valeur `boolean`. Les boléens ne sont pas des nombres.

  `Stem.details.lengths`
  : Bien que mentionnée dans Section "Stem" dans *Référence des propriétés internes* et ayant par défaut une liste de `numbers`, il s'agit d'une « sous-propriété ». Il n'existe à ce jour aucune prise en charge des « propriétés imbriquées ».

`\offset` *en tant que dérogation*

Lorsque *élément* est un nom d'objet graphique comme `Arpeggio` ou `Staff.OttavaBracket`, le comportement de la commande `\offset` est assimilable à un `\override` sur le type d'objet spécifié.

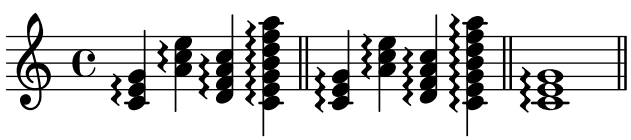
`\offset` *propriété décalages* `[`*contexte.*`]`*NomGrob*

Notez bien qu'il n'y a jamais de tiret avant un comportement « dérogatoire », tout comme il n'y en a jamais avec la commande `\override` elle-même.

L'exemple suivant utilise la forme « dérogation » pour allonger les arpeggios affichés dans la première mesure, jusqu'à couvrir l'accord dans son intégralité. Les arpeggios sont étirés d'un demi espace de portée à leur sommet ainsi qu'à leur base. Est aussi indiquée la même opération sur le premier accord à l'aide d'une simple dérogation appliquée à la propriété `positions`. Cette méthode n'est pas la plus illustrative pour « étirer d'un demi espace de portée » dans la mesure où les extrémités doivent être spécifiées en coordonnées absolues plutôt que relatives. De plus, des dérogations individuelles seraient nécessaires pour les autres accords en raison de leurs différentes amplitude et position.

```
arpeggioMusic = {
  <c' e' g'>\arpeggio <a' c'' e''>\arpeggio
  <d' f' a' c''>\arpeggio <c' e' g' b' d'' f'' a''>\arpeggio
}

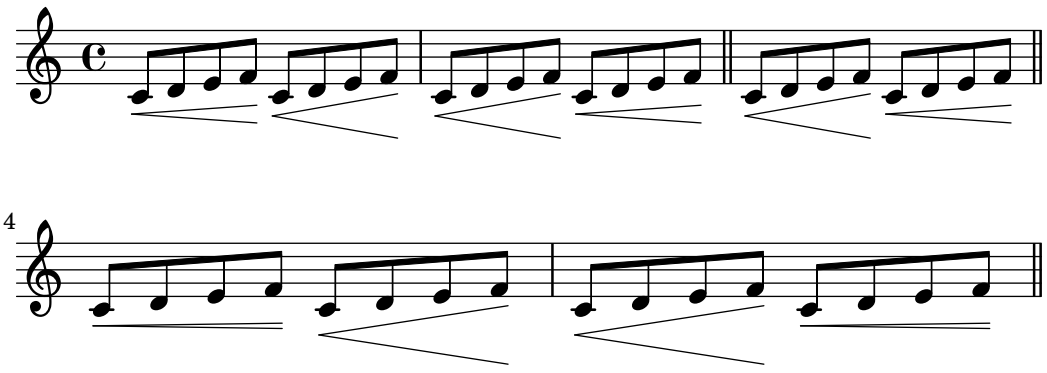
{
  \arpeggioMusic
  \bar "||"
  \offset positions #'(-0.5 . 0.5) Arpeggio
  \arpeggioMusic
  \bar "||"
  \once \override Arpeggio.positions = #'(-3.5 . -0.5)
  <c' e' g'>1\arpeggio
  \bar "||"
}
```



Dans cette utilisation d'*override*, `\offset` peut se préfixer de `\once` ou `\temporary` et être annulé à l'aide d'un `\revert` suivi de *propriété* – voir Section "Fonctions de substitution intermédiaires" dans *Extension de LilyPond*. Ceci tient au fait que `\offset` crée effectivement un `\override` de *propriété*.

```
music = { c'8\< d' e' f'\! }

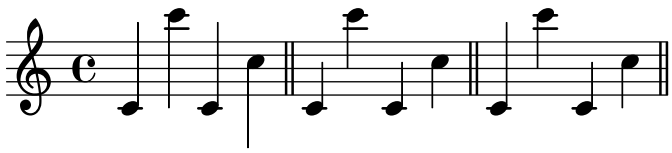
{
  \music
  \offset height 1 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
  \once \offset height 1 Hairpin
  \music \music
  \bar "||"
  \override Hairpin.height = 0.2
  \music
  \temporary \offset height 2 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
}
```



Tout comme `\override`, la forme « dérogation » de `\offset` peut s'utiliser avec `\undo` et `\single`.

```
longStem = \offset length 6 Stem

{
 \longStem c'4 c''' c' c''
 \bar "||"
 \undo \longStem c'4 c''' c' c''
 \bar "||"
 \single \longStem c'4 c''' c' c''
 \bar "||"
}
```



\offset *en tant qu'affinage*
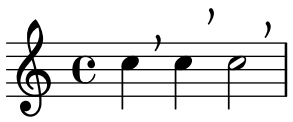
Lorsque *élément* est une expression musicale, comme (ou \arpeggio, le résultat sera la même expression musicale à laquelle aura été appliqué un affinage.

[-]\offset [*NomGrob.*]*propriété décalages expression-musicale*

La syntaxe de \offset dans sa forme « affinage » est en tout point analogue à la commande \tweak, autant dans l'ordre des arguments que dans la présence ou non du tiret l'introduisant.

L'exemple suivant utilise la forme « affinage » pour ajuster le positionnement vertical de l'objet BreathingSign. Les effets de la simple commande \tweak sont présent à titre de comparaison. Leur syntaxe est équivalente. Néanmoins, le résultat de \tweak est moins intuitif dans la mesure où BreathingSign.Y-offset est calculé en référence à la ligne médiane. Il n'est pas nécessaire de savoir comment se calcule Y-offset dans le cas d'un \offset.

```
{
  c''4
  \breathe
  c''4
  \offset Y-offset 2 \breathe
  c''2
  \tweak Y-offset 3 \breathe
}
```
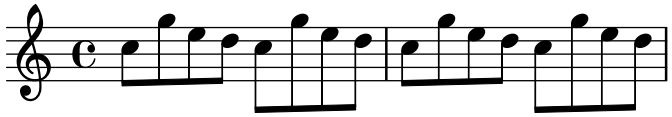


Dans cet exemple, les objets affinés étaient créés directement à partir du code saisi : la commande \breathe était une instruction explicite pour renvoyer un objet BreathingSign. Puisque la cible de la commande était sans ambiguïté, point n'était besoin de spécifier le nom de l'objet. Cependant, lorsqu'un objet est créé *indirectement*, mention du nom de l'objet devient requise. Il en va de même pour la commende \tweak.

Dans l'exemple qui suit, l'objet Beam est abaissé de deux espaces de portée par application de \offset à la propriété positions.

La première application de \offset requiert mention du nom de l'objet puisque rien dans le code ne crée explicitement de ligature. Dans la seconde application, la ligature est explicitement créée par l'expression musicale [, ce qui dispense de mentionner le nom de l'objet. Cette deuxième application comporte par ailleurs un raccourci : un unique number s'appliquera aux deux membres d'un number-pair.

```
{
  c''8 g'' e'' d''
  \offset Beam.positions #'(-2 . -2)
  c''8 g'' e'' d''
  c''8 g'' e'' d''
  c''8-\offset positions #-2 [ g'' e'' d'']
}
```



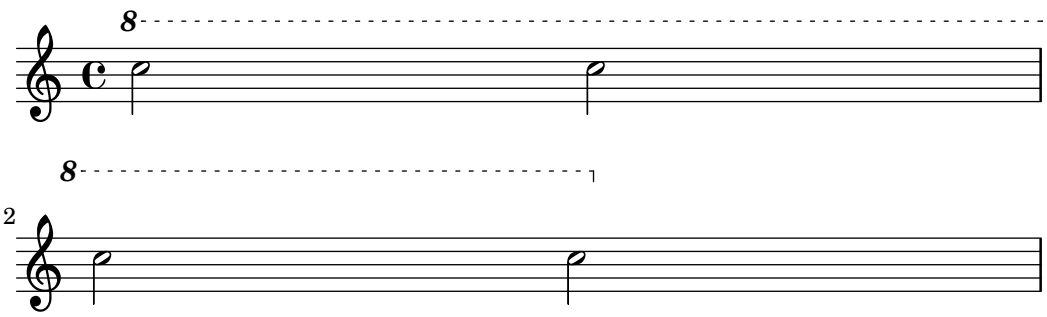
`\offset` *et les bandeaux avec rupture*

Il est aussi possible de modifier indépendamment les segments d'un objet étendu rencontrant des sauts de ligne. Dans ce cas, *décalages* est consitué d'une liste de valeurs pour le type de donnée requis par la propriété.

Utilisée de telle manière, la commande `\offset` est similaire à la commande `\alterBroken` – voir Section 5.5.5 [Modification de bandeaux avec rupture], page 694. Cependant, et contrairement à la commande `\alterBroken`, les valeurs fournies à `\offset` sont relatives.

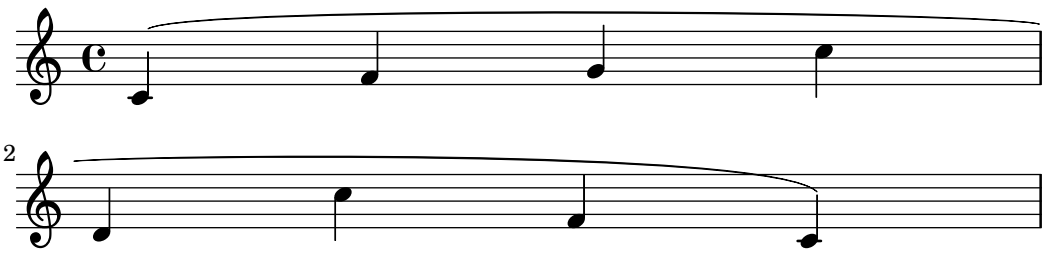
Dans l'exemple suivant est déplacé l'objet « segmenté » `OttavaBracket` au travers de sa propriété `staff-padding`. Puisque cette propriété est affectée d'un `number`, *décalages* est alimenté d'une liste de `number`s afin de prendre en compte les deux segments créés par le saut de ligne. La portion de crochet de la première ligne n'est en fait pas touchée puisque `0` est ajouté à la valeur par défaut de `staff-padding`. Le segment de la deuxième ligne est haussé de trois espaces de portée par rapport à sa hauteur par défaut. La hauteur par défaut est de `2`, bien qu'il ne soit pas nécessaire de le savoir.

```
{
 \offset staff-padding #'(0 3) Staff.OttavaBracket
  \ottava #1
  c'''2 c'''
  \break
  c'''2 c'''
}
```



L'exemple ci-dessous reproduit les effets de la commande `\shape` en décalant la propriété `control-points` de l'objet `Slur`. Ici, *décalages* est constitué d'une liste de `number-pair-list`s, une pour chaque segment de la liaison. Cet exemple produit un résultat identique à ce qui est illustré dans Section 5.5.4 [Modification de l'allure des éléments], page 690.

```
{
  c'4-\offset control-points #'(
              ((0 . 0) (0 . 0) (0 . 0) (0 . 1))
              ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
             ) ( f'4 g' c''
  \break
  d'4 c'' f' c')
}
```



5.3.7 Modification de listes associatives

Certaines propriétés configurables par l'utilisateur se présentent en interne comme étant des listes associatives – les puristes diront des *alists*. Une *alist* est en fait constituée de plusieurs paires de *clés* et *valeurs*. La structure d'un liste associative ressemble à :

```
'((clé1 . valeur1)
  (clé2 . valeur2)
  (clé3 . valeur3)
  ...)
```

Dans le cas où cette liste représente les propriétés d'un objet graphique ou bien l'une des variables du bloc `\paper`, chaque clé peut être modifiée individuellement sans que cela affecte les autres.

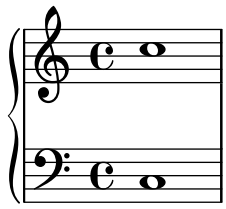
Par exemple, pour réduire l'espacement entre deux portées adjacentes d'un même système, on utilisera la propriété `staff-staff-spacing` qui est attachée à l'objet graphique `StaffGrouper`. Cette propriété est constituée d'une liste de quatre clés : `basic-distance`, `minimum-distance`, `padding` et `stretchability`. Ses réglages par défaut tels que mentionnés à la rubrique *Backend* de la référence des propriétés internes – voir Section "StaffGrouper" dans *Référence des propriétés internes* – sont :

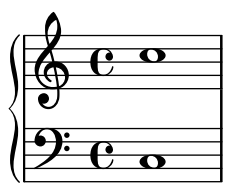
```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

Afin de rapprocher nos deux portées, il suffit de réduire la valeur (`9`) de la clé `basic-distance` au niveau de celle de la clé `minimum-distance` (`7`). La modification d'une seule clé individuellement peut se réaliser sous la forme d'une *déclaration imbriquée* :

```
% default space between staves
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>

% reduced space between staves
\new PianoStaff \with {
  % this is the nested declaration
  \override StaffGrouper.staff-staff-spacing.basic-distance = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```

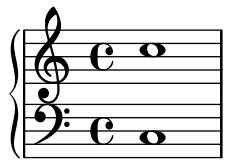




Le recours à une déclaration imbriquée touchera la clé indiquée (`basic-distance` dans l'exemple ci-dessus) sans pour autant modifier les autres clés de la propriété considérée.

Considérons maintenant que nous souhaitions que les portées soient le plus proche possible les unes des autres, à la limite du chevauchement. Il suffirait de mettre les quatre clés à zéro. Nous pourrions saisir quatre déclarations, chacune d'elles touchant une clé. Nous pouvons tout aussi bien redéfinir la propriété en une seule clause, sous la forme d'une liste associative :

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
       (minimum-distance . 0)
       (padding . 0)
       (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



N'oubliez pas que dès lors qu'une clé n'apparaît pas dans la liste, elle retourne à sa valeur *sauf-mention-contraire*. Autrement dit, dans le cas de `staff-staff-spacing` qui nous occupe, toutes les clés non mentionnées seront ramenées à zéro – à l'exception de `stretchability` qui prend par défaut la valeur de `basic-distance`. Les deux assertions suivantes sont donc équivalentes.

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
     (minimum-distance . 0)
     (padding . 0)
     (stretchability . 7))
```

L'une des conséquences, parfois involontaire, de ceci est la suppression de réglages standards effectués par un fichier d'initialisation chargé à chaque compilation d'un fichier source. Dans l'exemple précédent, les réglages standards de `padding` et `minimum-distance`, tels que déterminés par `scm/define-grobs.scm`, se voient ramenés à leur valeur *si-non-définie* ; autrement dit, les deux clés sont mises à zéro. La définition d'une propriété ou d'une variable sous forme de liste associative, quelle qu'en soit la taille, réinitialisera toujours les clés non mentionnées à leur valeur *si-non-définie*. Si telle n'est pas votre intention, nous vous recommandons alors de régler la valeur des clés individuellement par des déclarations imbriquées.

> **Note :** Les déclarations imbriquées ne sont pas fonctionnelles dans le cas des listes associatives des propriétés de contexte – telles `beamExceptions`, `keyAlterations`, `timeSignatureSettings`, etc. Ces propriétés ne sont modifiables qu'au travers d'une complète redéfinition de leur liste associative.

5.4 Propriétés et contextes utiles

5.4.1 Modes de saisie

La manière dont sera interprétée la notation contenue dans un fichier source dépend du mode affecté à la saisie. Il existe, en règle générale, deux façons de spécifier le mode : une forme développée – par exemple `\chordmode` – et une forme abrégée – par exemple `\chords`. La forme développée s'utilise particulièrement lorsque la saisie fait l'objet d'une variable ou se trouve dans un contexte explicitement créé. La forme abrégée crée implicitement un contexte du type adéquate à la saisie et la lui transmet directement. Cette forme abrégée est tout à fait indiquée aux situations simples pour lesquelles nul n'est besoin de créer explicitement le contexte qui prendra en charge la saisie.

Mode accords

Ce mode, activé par la commande `\chordmode`, permet d'interpréter les saisies comme étant des accords, qui seront imprimés sous forme de notes sur une portée – voir Section 2.7 [Notation des accords], page 444. La musique entrée en mode accords est rendue soit sous forme d'accords sur une portée pour un contexte `Staff`, soit sous forme de noms d'accord pour un contexte `ChordNames` ou sous forme de diagrammes pour un contexte `FretBoards`.

Le mode accords s'active aussi par la commande `\chords`, qui créera implicitement un nouveau contexte `ChordNames`. Le code saisi selon la syntaxe dévolue aux accords, sera interprété comme étant des accords nommés et sera alors rendu sous forme nominale dans ce contexte `ChordNames` – voir [Impression des noms d'accord], page 450.

Mode percussions

Ce mode, activé par la commande `\drummode`, permet d'interpréter les saisies comme étant de la notation pour percussions – voir [Notation de base pour percussions], page 423. Lorsqu'elle est entrée en mode percussions, la musique est rendue dans un contexte `DrumStaff`.

Le mode percussions s'active aussi par la commande `\drums`, qui créera implicitement un nouveau contexte `DrumStaff`. Le code saisi selon la syntaxe dévolue aux percussions, sera interprété comme étant de la notation pour percussions et alors rendu sous forme symbolique sur une portée de percussions – voir [Notation de base pour percussions], page 423.

Mode figures

Ce mode, activé par la commande `\figuremode`, permet d'interpréter les saisies comme étant de la basse chiffrée (ou figurée) – voir [Saisie de la basse chiffrée], page 460. Lorsqu'elle est entrée en mode figures, la musique est rendue sous forme de basse figurée dans un contexte `FiguredBass` ou dans un contexte `Staff`.

Le mode figures s'active aussi par la commande `\figures`, qui créera implicitement un nouveau contexte `FiguredBass`. Le code, saisi selon la syntaxe dévolue à la basse chiffrée, sera interprété comme étant des indication de basse chiffrée et sera alors rendu sous forme symbolique dans le contexte `FiguredBass` – voir [Introduction à la basse chiffrée], page 459.

Modes frets et tablatures

Il n'existe pas de mode spécifique pour saisir des symboles de fret ou de tablature.

Notes ou accords saisis en mode note puis affectés à un contexte `TabStaff` seront rendus sous forme de diagramme de tablature – voir [Tablatures par défaut], page 371.

Des diagrammes de fret viendront se positionner en surplomb d'une portée dès lors que les notes ou accords auront été saisis en mode note ou accord puis rendus dans un contexte `FretBoards` – voir [Tablatures automatiques], page 414. Ils peuvent aussi se gérer sous forme

de *markups* créés par la commande `\fret-diagram` – voir [Tablatures sous forme d'étiquette], page 393.

Mode paroles

Ce mode, activé par la commande `\lyricmode`, permet d'interpréter les saisies comme étant des syllabes, ayant éventuellement une durée, et des indications habituelles aux paroles – voir Section 2.1 [Musique vocale], page 287. Lorsqu'il est entré en mode paroles, le texte est rendu sous forme de syllabes dans un contexte `Lyrics`.

Le mode paroles s'active aussi par la commande `\lyrics`, qui créera implicitement un nouveau contexte `Lyrics`. Le code saisi sera interprété comme étant des paroles et sera alors rendu sous forme de syllabes dans le contexte `Lyrics`.

Le mode paroles s'active aussi par la commande `\addlyrics`, qui créera un contexte `Lyrics` et ajoutera implicitement une commande `\lyricsto` afin d'associer les paroles qui suivent à la musique précédemment saisie – voir [Durée automatique des syllabes], page 291.

Mode markup

Ce mode, activé par la commande `\markup`, permet d'interpréter les saisies comme étant des *markups* (annotations ou étiquettes) – voir Section A.11 [Commandes pour *markup*], page 746.

Mode notes

Le mode notes est le mode par défaut dans LilyPond. Il peut aussi s'activer par la commande `\notemode`. Les saisies seront interprétées comme étant des hauteurs, durées, *markups*, etc. qui seront rendues sous forme de notation musicale sur une portée.

Nul n'est besoin de spécifier le mode notes de manière explicite, hormis dans certaines situations particulières, notamment lorsque vous êtes en mode paroles, accords, ou tout autre mode, et que vous deviez insérer un élément qui ne serait disponible que grâce à la syntaxe du mode notes.

5.4.2 Direction et positionnement

En matière de typographie musicale, l'orientation et le positionnement de nombreux éléments est affaire de goût. Par exemple, les hampes peuvent être ascendantes ou descendantes, les paroles, nuances ou autres indications d'expression peuvent apparaître au-dessus ou en dessous de la portée, les indications textuelles s'alignent tantôt par la gauche, tantôt par la droite, ou être centrées. La plupart de ces choix peuvent être laissés à l'appréciation de LilyPond. Il peut être préférable, dans certains cas, d'imposer l'orientation ou le positionnement des éléments.

Indicateurs de position d'une articulation

Certains positionnements sont opérés par défaut – toujours au-dessus ou toujours en dessous (nuances ou points d'orgue) – alors que d'autres alterneront selon l'orientation des hampes (liaisons ou accents).
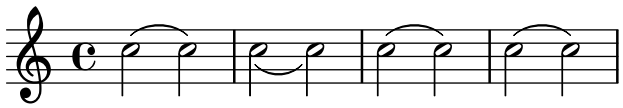
Le positionnement par défaut peut être outrepassé à l'aide d'un *indicateur de positionnement*, qui vient s'insérer juste avant l'articulation. LilyPond met à votre disposition trois indicateurs de positionnement : `^` (pour « au-dessus »), `_` (pour « au-dessous »), et `-` (pour « appliquer le positionnement par défaut »). L'indicateur de positionnement n'est pas obligatoire ; LilyPond considère alors qu'il y a un `-`. Un indicateur de positionnement est cependant **obligatoire** dans les cas suivants :

- une commande `\tweak`,
- une commande `\markup`,
- une commande `\tag`,

- les indications de corde, par exemple `-"corde"`,
- les indications de doigté, par exemple `-1`,
- les raccourcis d'articulation, par exemple `-.`, `->` ou `--`.

Les indicateurs de positionnement n'affectent que la note qui suit :

```
\relative {
  c''2( c)
  c2_( c)
  c2( c)
  c2^( c)
}
```



La propriété `direction`

Le positionnement ou l'orientation de nombreux objets de rendu sont gérés par la propriété `direction`.

La propriété `direction` peut prendre la valeur `1`, qui signifie « ascendant » ou « au-dessus », ou `-1`, qui signifie « descendant » ou « au-dessous ». Les symboliques `UP` et `DOWN` peuvent remplacer respectivement `1` et `-1`. Les valeurs `0` ou `CENTER` permettent de réaffecter à la propriété `direction` son comportement par défaut. Certaines commandes prédéfinies permettent par ailleurs de spécifier un comportement en matière d'orientation ou positionnement ; elles sont de la forme

`\`*xxx*`Up, \`*xxx*`Down et \`*xxx*`Neutral`

auquel cas `\`*xxx*`Neutral` signifie « retour au comportement par défaut » – voir Section "Objets inclus dans la portée" dans *Manuel d'initiation*.

Dans quelques cas particuliers, comme l'indication d'un *arpeggio*, la valeur affectée à la propriété `direction` déterminera si l'objet doit se placer à gauche ou à droite de son parent. Un `-1` ou `LEFT` signifiera alors « sur la gauche », et un `1` ou `RIGHT` « sur la droite ». Comme de bien entendu, un `0` ou `CENTER` signifiera « appliquer le positionnement par défaut ».

Notez que ces commandes resteront effectives jusqu'à ce qu'elles soient annulées.

```
\relative {
  c''2( c)
  \slurDown
  c2( c)
  c2( c)
  \slurNeutral
  c2( c)
}
```



En matière de musique polyphonique, il est souvent plus judicieux d'utiliser des contextes `Voice` explicites que de modifier l'orientation des objets. Pour de plus amples informations, voir Section 1.5.2 [Plusieurs voix], page 179.

Voir aussi

Manuel d'initiation : Section "Objets inclus dans la portée" dans *Manuel d'initiation*.

Manuel de notation : Section 1.5.2 [Plusieurs voix], page 179.

5.4.3 Distances et unités de mesure

LilyPond considère deux types de distances : les distances absolues et les distances relatives ou extensibles.

Les distances absolues permettent de spécifier les marges, indentations et autres détails de mise en page ; elles s'expriment par défaut en millimètres. Vous pouvez utiliser d'autres systèmes de mesure dès lors que la quantité est suivie de la mesure : `\mm`, `\cm`, `\in` (pouces) ou `\pt` (points, 1/72,27 pouce). Les mesures de mise en page peuvent aussi s'exprimer en unité extensible de portée `\staff-space` (voir ci-après). Pour plus d'information concernant la mise en page, voir la rubrique Section 4.1 [Mise en forme de la page], page 571.

Les distances relatives ou extensibles s'expriment toujours en « espace de portée » ou, plus rarement, en « demi espace de portée ». L'espace de portée (*staff-space*) correspond à la distance qui sépare deux lignes adjacentes d'une portée. Sa valeur par défaut est déterminée globalement par la taille de portée. Elle peut aussi s'ajuster ponctuellement en jouant sur la propriété `staff-space` de l'objet `StaffSymbol`. Les distances relatives s'ajustent automatiquement dès qu'une modification de la taille globale de portée ou bien de la propriété `staff-space` du `StaffSymbol` intervient. Cependant, les tailles de fonte ne s'ajusteront automatiquement que si la modification touche la taille globale des portées. La taille globale de portée permet ainsi de gérer l'aspect général de la partition – voir Section 4.2.2 [Définition de la taille de portée], page 585.

Lorsque seulement une portion de partition doit apparaître dans une taille, comme par exemple une portée d'ossia ou une note de bas de page, influer sur la taille globale de portée affecterait l'intégralité de la partition. Il convient donc dans ce cas de modifier à la fois la propriété `staff-space` du `StaffSymbol` et la taille des fontes. La fonction Scheme `magstep` est tout spécialement chargée d'adapter une modification du `staff-space` aux fontes. Pour de plus amples informations, reportez-vous à la rubrique Section "Longueur et épaisseur des objets" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Longueur et épaisseur des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 4.2.2 [Définition de la taille de portée], page 585, Section 4.1 [Mise en forme de la page], page 571.

5.4.4 Dimensions

Les dimensions d'un objet graphique spécifient la position des bords droit et gauche ainsi que des bords supérieur et inférieur de la boîte englobante de ces objets, en tant que distance par rapport au point de référence de l'objet et en unité d'espace de portée. Ces positions sont normalement codées sous la forme de deux paires Scheme. Par exemple, la commande de *markup* `\with-dimensions` prend trois arguments, les deux premiers étant des paires Scheme donnant la position des bords gauche et droit et celle des bords inférieur et supérieur :

```
\with-dimensions #'(-5 . 10) #'(-3 . 15) argument3
```

Ceci spécifie une boîte englobante pour *argument3* dont le bord gauche est à −5, le bord droit à 10, le bord inférieur à −3 et le bord supérieur à 15 espaces de portée du point de référence de cet objet.

Voir aussi
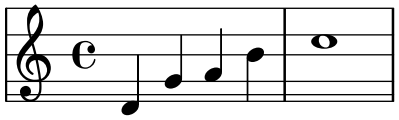
Manuel de notation : Section 5.4.3 [Distances et unités de mesure], page 669.

5.4.5 Propriétés des symboles de la portée

L'emplacement vertical et le nombre de lignes d'une portée se définissent conjointement. Comme l'illustre l'exemple suivant, le positionnement des notes n'est en rien influencé par le positionnement des lignes de la portée.
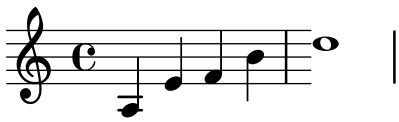
> **Note :** La propriété `'line-positions` écrase la propriété `'line-count`. Le nombre de lignes d'une portée est implicitement défini par le nombre d'éléments dans la liste des valeurs de `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol.line-positions = #'(7 3 0 -4 -6 -7)
}
\relative { a4 e' f b | d1 }
```



La largeur d'une portée, exprimée en espace de portée, peut être figée. L'espacement des objets inclus dans cette portée ne sera en rien affecté par ce réglage.

```
\new Staff \with {
  \override StaffSymbol.width = #23
}
\relative { a4 e' f b | d1 }
```



5.4.6 Extenseurs et prolongateurs

De nombreux objets de notation musicale s'étendent sur plusieurs notes, voire même sur plusieurs mesures. Il en va ainsi des liaisons, ligatures, crochets de n-olet, crochets de reprise, crescendos, trilles ou glissandos. Ces objets, que l'on englobe sous l'appellation « d'extenseurs », sont pourvus de propriétés spécifiques destinées à contrôler leur apparence et leur comportement. Un certain nombre de ces propriétés sont communes à tous les extenseurs, d'autres n'affectent que certains d'entre eux.

Tout extenseur dispose de la `spanner-interface`. Quelques uns, tout particulièrement ceux chargés de dessiner une ligne droite entre deux objets, disposent aussi de la `line-spanner-interface`.

```
+
```

Utilisation de `spanner-interface`
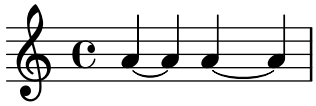
Cette interface fournit deux propriétés qui s'appliquent à certains extenseurs.

La propriété `minimum-length`

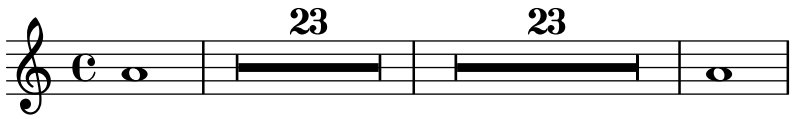
La longueur minimale d'un extenseur est déterminée par la propriété `minimum-length`. Au plus sa valeur est élevée, au plus l'espacement des notes qui le bornent sera grand. Forcer sa valeur restera néanmoins sans effet pour un certain nombre d'extenseurs dont la longueur dépend d'autres considérations. Voici quelques exemples de mise en œuvre de cette propriété.

```
a'~ a'
```
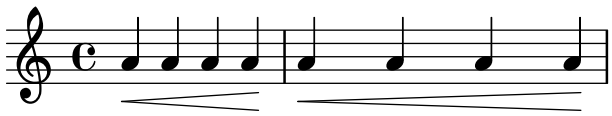
```
a'
% increase the length of the tie
-\tweak minimum-length #5
~ a'
```



```
\relative \compressMMRests {
  a'1
  R1*23
  % increase the length of the rest bar
  \once \override MultiMeasureRest.minimum-length = #20
  R1*23
  a1
}
```




```
\relative {
  a' \< a a a \!
  % increase the length of the hairpin
  \override Hairpin.minimum-length = #20
  a \< a a a \!
}
```



Cette propriété permet aussi de jouer sur l'envergure d'une liaison d'articulation ou de phrasé.

```
\relative {
  a'( g)
  a
  -\tweak minimum-length #5
  ( g)

  a\( g\)
  a
  -\tweak minimum-length #5
  \( g\)
}
```
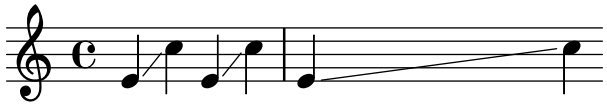


Certains objets requièrent un appel explicite à la procédure `set-spacing-rods` pour que la propriété `minimum-length` produise ses effets. La propriété `set-spacing-rods` doit alors

prendre pour valeur `ly:spanner::set-spacing-rods`. Par exemple, la longueur minimale d'un glissando ne pourra être forcée tant que la propriété `springs-and-rods` n'aura pas été définie :

```
% default
e' \glissando c''

% not effective alone
\once \override Glissando.minimum-length = #20
e' \glissando c''
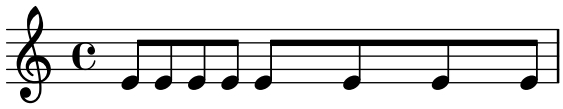
% effective only when both overrides are present
\once \override Glissando.minimum-length = #20
\once \override Glissando.springs-and-rods =
                  #ly:spanner::set-spacing-rods
e' \glissando c''
```



Il en va de même pour l'objet `Beam` (ligature) :

```
% not effective alone
\once \override Beam.minimum-length = #20
e'8 e' e' e'
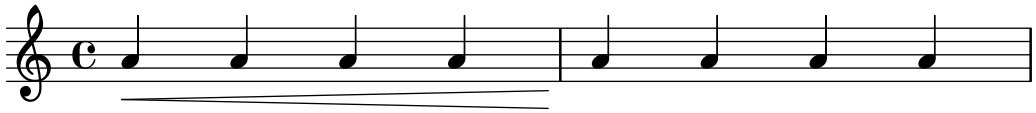
% effective only when both overrides are present
\once \override Beam.minimum-length = #20
\once \override Beam.springs-and-rods =
                  #ly:spanner::set-spacing-rods
e'8 e' e' e'
```

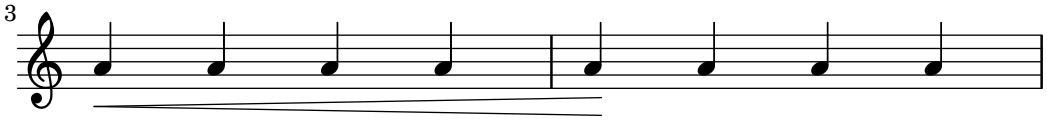


La propriété `to-barline`

La seconde propriété fournie par la `spanner-interface` est `to-barline`. Elle est activée par défaut, raison pour laquelle les soufflets et autres extenseurs finissant sur la première note d'une mesure s'arrêtent visuellement au niveau de la barre de mesure qui la précède. Le fait de désactiver la propriété `to-barline` aura pour effet de prolonger l'extenseur au delà de la barre de mesure et jusqu'à la note qui le borne :

```
\relative {
  a' \< a a a a \! a a a \break
  \override Hairpin.to-barline = ##f
  a \< a a a a \! a a a
}
```





Cette propriété n'est pas opérationnelle pour tous les extenseurs. Il serait en effet quelque peu surprenant de l'activer (lui affecter `#t`) dans le cas d'une liaison d'articulation ou de phrasé !
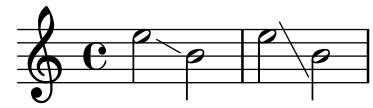
Utilisation de `line-spanner-interface`

Un certain nombre d'objets disposent de la `line-spanner-interface`, entre autres :

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

La routine en charge de dessiner le stencil de ces extenseurs est `ly:line-spanner::print`. Elle va déterminer les deux points extrêmes et dessiner entre eux une ligne du style requis. Bien que la localisation des deux bornes de l'extenseur soit calculée à la volée, vous pouvez cependant forcer leur ordonnée (coordonnée-Y). Les propriétés que vous devrez ajuster résident au deuxième niveau dans la hiérarchie, mais la syntaxe de la commande `\override` nécessaire demeure relativement simple :

```
e''2 \glissando b'
\once \override Glissando.bound-details.left.Y = #3
\once \override Glissando.bound-details.right.Y = #-2
e''2 \glissando b'
```

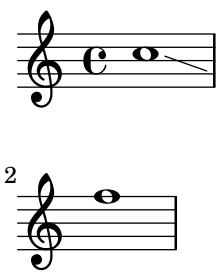


La propriété `Y` est valorisée en unités de `staff-space`, la ligne médiane de la portée correspondant au point zéro. Pour le glissando qui nous occupe, il s'agit du `Y` à l'aplomb (coordonnée-X) du centre de la tête de chacune des deux notes, si tant est que la ligne doive s'étendre entre ces deux points.

Si le `Y` n'est pas défini, sa valeur sera calculée en fonction de la position verticale du point d'attachement de l'extenseur.

Dans le cas où l'extenseur est interrompu par un saut de ligne, les terminaisons à cet endroit se gèrent à l'aide des sous-clés `left-broken` et `right-broken` de `bound-details`, comme ci-dessous :

```
\override Glissando.breakable = ##t
\override Glissando.bound-details.right-broken.Y = #-3
c''1 \glissando \break
f''1
```



Les sous-propriétés `left` et `right` du `bound-details` disposent d'autres clés modifiables de la même manière que `Y` :

`Y` Détermine l'ordonnée (coordonnée-Y) de la terminaison, avec un décalage en `staff-space` par rapport à la ligne médiane de la portée. Il s'agit par défaut du centre de l'objet d'attachement, qui est le centre vertical de la tête de note pour un glissando.

En ce qui concerne les extenseurs horizontaux, tels ceux comportant du texte ou les trilles, il est fixé à `0`.
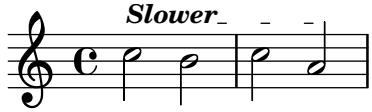
`attach-dir` Détermine le début et la fin de la ligne sur l'axe des abscisses, relativement à l'objet de rattachement. Une valeur de `-1` (ou `LEFT`) aura pour effet de commencer ou terminer la ligne sur la gauche de la tête de note de rattachement.

`X` Il s'agit de l'abscisse (coordonnée-X) absolue de la terminaison. Elle se calcule à la volée, et son forçage n'apporte rien de plus.

`stencil` Les extenseurs linéaires peuvent commencer ou finir par un symbole, enregistré dans cette sous-propriété. Elle est conçue pour un usage interne, aussi nous vous conseillons de plutôt recourir à `text`.

`text` Il s'agit d'un *markup* qui se poursuivra par l'extenseur. C'est la sous-propriété utilisée pour ajouter *cresc.*, *tr* ou autre texte à un extenseur horizontal.

```
\override TextSpanner.bound-details.left.text
   = \markup { \small \bold Slower }
\relative { c''2\startTextSpan b c a\stopTextSpan }
```
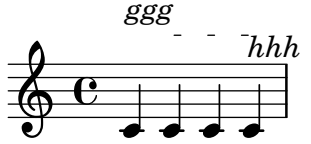


`stencil-align-dir-y`
`stencil-offset`

Lorsqu'aucune de ces deux sous-propriétés n'est définie, le stencil est simplement positionné à l'extrémité, centré sur la ligne telle que définie par les sous-propriétés `X` et `Y`. L'utilisation de `stencil-align-dir-y` ou `stencil-offset` permettra d'aligner le symbole verticalement par rapport au coin de la ligne :

```
\override TextSpanner.bound-details
            .left.stencil-align-dir-y = #-2
\override TextSpanner.bound-details
            .right.stencil-align-dir-y = #UP

\override TextSpanner.bound-details.left.text = "ggg"
\override TextSpanner.bound-details.right.text = "hhh"

\relative { c'4^\startTextSpan c c c \stopTextSpan }
```



Vous n'aurez pas manqué de constater qu'une valeur négative place le texte *en haut* – contrairement à ce que l'on serait en droit d'attendre. Ceci est dû au fait
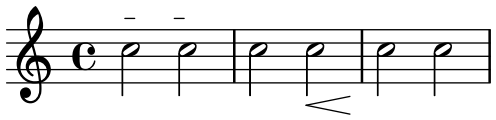
que la valeur `-1` ou `DOWN` signifie « aligner le bord *inférieur* du texte sur la ligne d'extension ». Une valeur égale à `1` ou `UP` alignera le sommet du texte sur cette ligne d'extension.

`arrow` L'activation de cette sous-propriété (lui affecter `#t`) ajoutera à l'extenseur une terminaison en flèche.

`padding` Cette sous-propriété contrôle l'espace qui doit séparer l'extrémité de la ligne et la fin réelle de l'extenseur. Sans ce « décalage », le trait indiquant un glissando commencerait et finirait au beau milieu de chacune des têtes de note.

La fonction `\endSpanners` permet d'interrompre l'extenseur qui vient dès la note suivante. Autrement dit, il ne s'étendra que sur une seule note, ou jusqu'à la prochaine barre de mesure si `to-barline` a été activé et que survient une barre avant la note suivante.

```
\relative c'' {
  \endSpanners
  c2 \startTextSpan c2 c2
  \endSpanners
  c2 \< c2 c2
}
```



L'utilisation de `\endSpanners` permet de s'affranchir d'insérer un `\stopTextSpan` pour clôturer un `\startTextSpan` ou un `\!` pour terminer un soufflet.

Voir aussi

Référence des propriétés internes : Section “Glissando” dans *Référence des propriétés internes*, Section “line-spanner-interface” dans *Référence des propriétés internes*, Section “TextSpanner” dans *Référence des propriétés internes*, Section “TrillSpanner” dans *Référence des propriétés internes*, Section “VoiceFollower” dans *Référence des propriétés internes*.

5.4.7 Visibilité des objets

La visibilité des objets de rendu se contrôle de quatre façons différentes : vous pouvez supprimer leur stencil, les rendre transparents, les coloriser en blanc ou bien encore forcer leur propriété `break-visibility`. Les trois premières options peuvent s'appliquer à tous les objets, la dernière étant réservée aux objets *changeables*. Le Manuel d'initiation contient une introduction à ces quatre techniques, à la rubrique Section “Visibilité et couleur des objets” dans *Manuel d'initiation*.

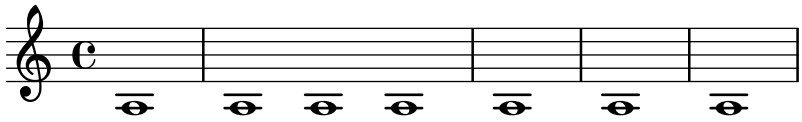
LilyPond met en œuvre quelques techniques particulières adaptées à certains objets ; elles sont couvertes par une rubrique spécifique.

Suppression des stencils

Tout objet de rendu se voit attribuer une propriété `stencil`. Elle est par défaut définie par la fonction chargée de dessiner cet objet. Lorsque cette propriété est désactivée de force – en lui attribuant la valeur `#f` – aucune fonction ne sera appelée ; l'objet ne sera donc pas dessiné. Le retour au comportement par défaut s'opère à l'aide d'un `\revert`.

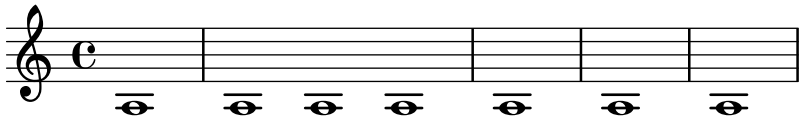
```
a1 a
\override Score.BarLine.stencil = ##f
a a
```

```
\revert Score.BarLine.stencil
a a a
```



Cette opération relativement courante fait l'objet du racourci `\omit` :
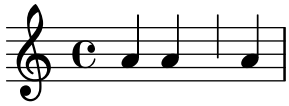
```
a1 a
\omit Score.BarLine
a a
\undo \omit Score.BarLine
a a a
```



Transparence des objets
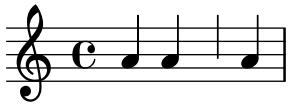
Tout objet de rendu dispose d'une propriété de transparence, qui est par défaut définie à `#f`. Le fait de l'activer rendra l'objet transparent tout en préservant la place qu'il occupe.

```
a'4 a'
\once \override NoteHead.transparent = ##t
a' a'
```



Cette opération relativement courante fait l'objet du racourci `\hide` :
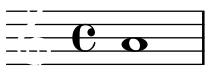
```
a'4 a'
\once \hide NoteHead
a' a'
```



Blanchiment des objets

Tout objet de rendu dispose d'une propriété couleur, par défaut définie à `black` (noir). Le fait de la forcer à `white` (blanc) rendra l'objet indistinct du fond blanc. Néanmoins, lorsque cet objet en recouvre d'autres, la couleur de leurs points de jonction dépendra de l'ordre dans lequel ils sont dessinés, ce qui peut laisser apparaître un fantôme de l'objet blanchi comme ci-dessous :

```
\override Staff.Clef.color = #white
a'1
```



Cet inconvénient peut être évité en modifiant l'ordre dans lequel les objets sont dessinés. Chaque objet de rendu dispose d'une propriété `layer` (calque ou niveau) à laquelle est affecté un nombre entier. Les objets ayant la plus faible valeur sont dessinés en premier, puis les autres, de telle sorte qu'un objet ayant une valeur plus élevée les recouvrira. La plupart des objet ont un `layer` valorisé à `1` – quelques uns, dont `StaffSymbol` et `BarLine`, ont une valeur à `0`. L'ordre d'impression d'objets ayant une même valeur de `layer` est indéterminé.

La clef de l'exemple précédent a par défaut un `layer` à `1` ; elle est donc dessinée après les lignes de la portée – `layer` valorisé par défaut à `0` – et donc les recouvre. Pour changer cet état de fait, l'objet `Clef` doit avoir un `layer` de valeur inférieure, disons `-1`, pour pouvoir être dessiné avant.

```
\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1
a'1
```
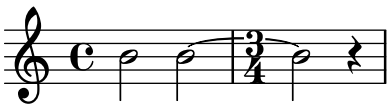


Morceaux choisis

Utilisation de la propriété `whiteout`

Tout objet graphique peut s'imprimer sur un fond blanc afin de masquer une partie des objets qu'il recouvre. Ceci trouve toute son utilité pour améliorer certaines collisions, notamment dans des situations où un repositionnement d'objets est irréaliste. Il faut alors explicitement définir la propriété `layer` afin de contrôler quels objets seront masqués par le fond blanc.

Dans l'exemple ci-dessous, la liaison est en collision avec la métrique ; la situation est améliorée dès lors que la portion de liaison qui traverse la métrique est masquée par l'affectation de la propriété `whiteout` à l'objet `TimeSignature`. Pour ce faire, `TimeSignature` est déplacé sur un calque au-dessus de celui de `Tie` – il reste sur le calque par défaut (1) –, puis le `StaffSymbol` est placé sur un calque supérieur à celui de `TimeSignature` pour ne pas être masqué.

```
{
  \override Score.StaffSymbol.layer = #4
  \override Staff.TimeSignature.layer = #3
  b'2 b'~
  \once \override Staff.TimeSignature.whiteout = ##t
  \time 3/4
  b' r4
}
```



Utilisation de break-visibility

La plupart des objets de rendu ne sont imprimés qu'une seule fois ; certains cependant, tels les barres de mesure, clefs, métriques ou armures, apparaîtront deux fois lors d'un saut de ligne – une première fois en fin de ligne, puis à nouveau au début de la ligne suivante. Ces objets, que l'on peut traiter de *changeables* (*breakable* en anglais) disposent de la propriété `break-visibility` spécialement chargée de contrôler leur visibilité aux trois endroits où il sont susceptibles d'apparaître : en début de ligne, en cours de ligne ou en fin de ligne – si tant est qu'un changement s'y produise.

Par exemple, la métrique est imprimée par défaut au début de la première ligne, et nulle part ailleurs. En cas de modification, une nouvelle métrique sera imprimée à l'endroit du changement. Dans le cas où ce changement intervient en fin de ligne, la nouvelle métrique s'imprime au début de la ligne suivante, et une métrique « de précaution » viendra se placer au bout de la ligne précédente.

Ce comportement est géré par la propriété `break-visibility`, dont vous trouverez une explication à la rubrique Section "Visibilité et couleur des objets" dans *Manuel d'initiation*. Cette propriété est constituée d'un vecteur de trois booléens qui, dans l'ordre, déterminent si l'objet sera imprimé à la fin, en cours, et au début d'une ligne – on pourrait aussi dire avant un saut de ligne, là où il n'y a pas de saut de ligne, et après un saut de ligne.

Les huit combinaisons possibles sont aussi disponibles sous la forme de fonctions prédéfinies, regroupées dans le fichier `scm/output-lib.scm`. Le tableau suivant vous les présente ; les trois dernières colonnes indiquent l'endroit où l'objet sera visible.

Forme fonctionnelle	**Forme vectorielle**	**Avant saut**	**Hors saut**	**Après saut**
`all-visible`	`#(#t #t #t)`	oui	oui	oui
`begin-of-line-visible`	`#(#f #f #t)`	non	non	oui
`center-visible`	`#(#f #t #f)`	non	oui	non
`end-of-line-visible`	`#(#t #f #f)`	oui	non	non
`begin-of-line-invisible`	`#(#t #t #f)`	oui	oui	non
`center-invisible`	`#(#t #f #t)`	oui	non	oui
`end-of-line-invisible`	`#(#f #t #t)`	non	oui	oui
`all-invisible`	`#(#f #f #f)`	non	non	non

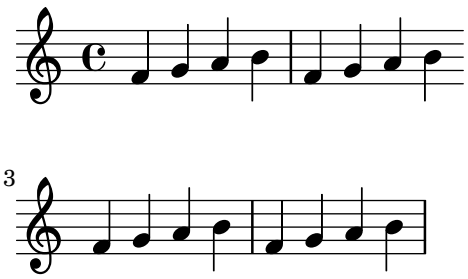
Les réglages par défaut de la propriété `break-visibility` diffèrent selon l'objet de rendu. Le tableau suivant présente, pour la plupart des objets comportant la propriété `break-visibility`, ces réglages par défaut.

Objet de rendu	**Contexte habituel**	**Réglage par défaut**
`BarLine`	`Score`	calculé
`BarNumber`	`Score`	`begin-of-line-visible`
`BreathingSign`	`Voice`	`begin-of-line-invisible`
`Clef`	`Staff`	`begin-of-line-visible`
`Custos`	`Staff`	`end-of-line-visible`
`DoublePercentRepeat`	`Voice`	`begin-of-line-invisible`
`KeySignature`	`Staff`	`begin-of-line-visible`
`ClefModifier`	`Staff`	`begin-of-line-visible`
`RehearsalMark`	`Score`	`end-of-line-invisible`
`TimeSignature`	`Staff`	`all-visible`

Voici un exemple d'utilisation de la forme vectorielle pour contrôler la visibilité des barres de mesure :

```
\relative {
  f'4 g a b
  f4 g a b
  % Remove bar line at the end of the current line
  \once \override Score.BarLine.break-visibility = ##(#f #t #t)
  \break
```

```
  f4 g a b
  f4 g a b
}
```



Lors d'un forçage de `break-visibility` sous une forme vectorielle, les trois éléments doivent impérativement être mentionnés. Ces formes vectorielles ne sont d'ailleurs pas prises en charge par tous les objets de rendu, et certaines combinaisons peuvent entraîner des erreurs ; nous citerons entre autres :
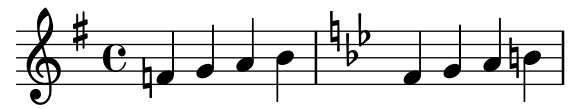
- Une barre de mesure ne peut s'imprimer en début de ligne.
- Un numéro de mesure ne peut apparaître au début de la première ligne, à moins d'être différent de 1.
- Clef – voir ci-après.
- Les répétitions en pourcentage sont soit toutes imprimées, soit aucune. Vous devrez utiliser `begin-of-line-invisible` pour les voir et `all-invisible` pour les masquer.
- Armure – voir ci-après.
- Modificateur de clef – voir ci-après.
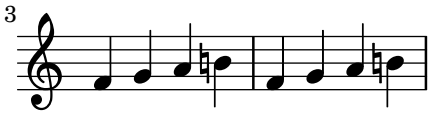
Considérations spécifiques

Visibilité après changement explicite

La propriété `break-visibility` contrôle la visibilité des armures ou changements de clef en début de ligne uniquement, donc après un saut. Elle ne produit aucun effet sur la visibilité d'une armure ou d'une clef après un changement explicite de tonalité ou de clef, ni en cours, ni en fin de ligne. Dans l'exemple suivant, l'armure est présente même après le passage en si bémol majeur malgré l'activation de `all-invisible` (*tous invisibles*).
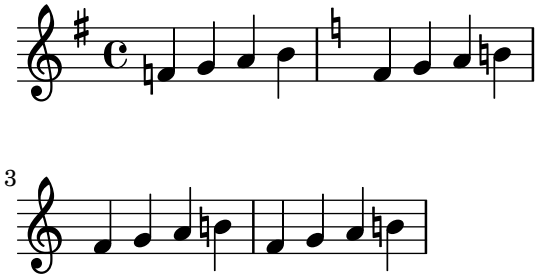
```
\relative {
  \key g \major
  f'4 g a b
  % Try to remove all key signatures
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b
  \break
  f4 g a b
}
```





La visibilité lors de ces changements explicites d'armure ou de clef est géré respectivement par les propriétés `explicitKeySignatureVisibility` et `explicitClefVisibility`. Leur fonctionnement est en tout point identique à celui de la propriété `break-visibility` – forme vectorielle à trois éléments ou forme fonctionnelle comme indiqué ci-avant. Toutes deux sont attachées au contexte `Staff` (la portée) et non directement aux objets de rendu ; elles sont de ce fait introduites par une instruction `\set`. Leur valeur par défaut est de toujours imprimer les objets – réglage sur `all-visible`. Ces deux propriétés gèrent uniquement la visibilité des armures et clefs lors d'un changement explicite, et en dehors d'un début de ligne ; il faudra en pareil cas forcer la `break-visibility` de ces objets pour les supprimer.
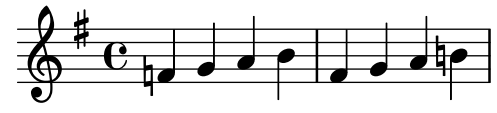
```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```
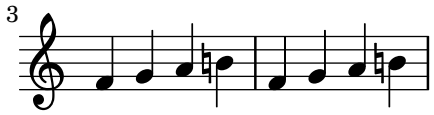


Visibilité des bécarres de précaution

L'impression d'altérations de précaution au moment d'un changement explicite de tonalité sera annulée dès lors que vous aurez désactivé la propriété `printKeyCancellation` du contexte `Staff` :

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```

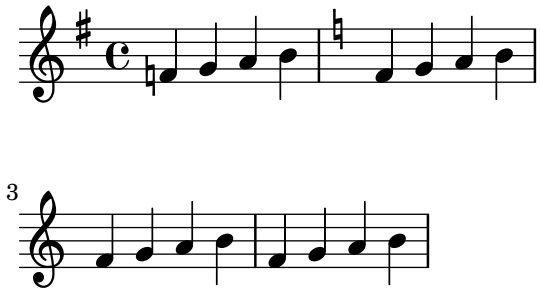




Avec de tels réglages particuliers, seules les altérations accidentelles permettront d'indiquer le changement de tonalité.
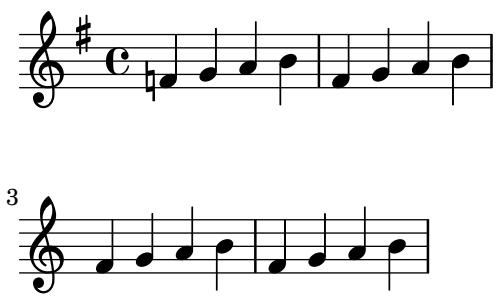
Notez bien que lors d'une bascule en do majeur ou la mineur, seuls les « bécarres d'annulation » permettent d'identifier le changement de tonalité. En pareil cas, désactiver `printKeyCancellation` sera sans effet :

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



La suppression des bécarres d'annulation même lors d'un passage en do majeur ou la mineur n'interviendra qu'après modification de la visibilité de l'objet `KeyCancellation` :

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeyCancellation.break-visibility = #all-invisible
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Barres de mesure automatiques

La désactivation de la propriété `automaticBars`, qui réside dans le contexte `Score`, permet de s'affranchir d'imprimer automatiquement les barres de mesure ; seules seront imprimées les barres explicitées à l'aide de la commande `\bar`. Néanmoins, et contrairement à ce qui se passe avec la commande `\cadenzaOn`, le compteur de numéro de mesure continuera de s'incrémenter. Les barres s'imprimeront à nouveau, au niveau où en est le compteur, dès que la propriété `automaticBars` sera réactivée. Gardez à l'esprit que les sauts de ligne, lorsque cette propriété est désactivée, ne peuvent intervenir qu'à l'occasion d'un `\bar` explicite.

Clefs transposées

L'indication de transposition d'une clef est produite par l'objet de rendu `ClefModifier`. Sa visibilité étant gérée par héritage direct de l'objet `Clef`, nul n'est besoin de forcer un quelconque `break-visibility` au niveau des objets `ClefModifier` pour éliminer une indication de transposition lorsque la clef est invisible.

Lors d'un changement explicite de clef, la propriété `explicitClefVisibility` gère à la fois le symbole de la clef et l'indication de transposition qui lui est attachée.

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*
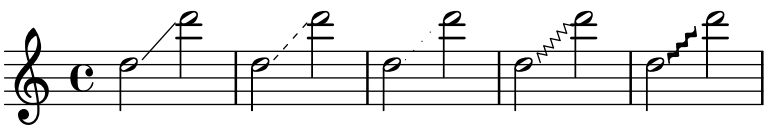
5.4.8 Styles de ligne

Certaines indications portées à l'attention de l'exécutant – tels *rallentando*, *accelerando* et *trilles* – apparaissent sous la forme d'un texte qui peut s'étendre sur plusieurs mesures à l'aide d'une ligne parfois pointillée ou ondulée.

En matière de dessin du texte et des lignes, ces indications font appel aux mêmes routines que le glissando ; leur comportement peut donc être affiné selon les mêmes préceptes, au moyen de la routine `ly:line-spanner::print` qui est tout spécialement chargée de dessiner les extenseurs. Cette routine détermine l'emplacement exact des deux points extrêmes de l'extenseur, puis trace une ligne du style demandé entre ces deux points.

L'exemple ci-dessous indique les différents styles de ligne disponibles, ainsi que la manière de les spécifier.
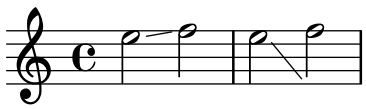
```
\relative {
  d''2 \glissando d'2
  \once \override Glissando.style = #'dashed-line
  d,2 \glissando d'2
  \override Glissando.style = #'dotted-line
  d,2 \glissando d'2
  \override Glissando.style = #'zigzag
  d,2 \glissando d'2
  \override Glissando.style = #'trill
  d,2 \glissando d'2
}
```



Les points d'ancrage de l'extension sont calculés à la volée pour chaque objet graphique, mais rien ne vous empêche de les forcer :

```
\relative {
```

```
  e''2 \glissando f
  \once \override Glissando.bound-details.right.Y = #-2
  e2 \glissando f
}
```



La valeur de `Y` est ainsi fixée à `-2` en ce qui concerne la borne droite. Il en irait de même pour la borne gauche en spécifiant `left` (gauche) au lieu de `right` (droite).

En l'absence de réglage du `Y`, celui-ci est calculé à partir de l'emplacement vertical des points d'attache gauche et droit de l'extenseur.

De plus amples informations quant à l'ajustement des extenseurs font l'objet de la rubrique Section 5.4.6 [Extenseurs et prolongateurs], page 670.

5.4.9 Rotation des objets

Qu'il s'agisse des objets de rendu ou d'éléments textuels sous forme de *markup*, vous pouvez les faire pivoter selon vos désirs et à partir de n'importe quel point. La méthode diffère cependant selon ce que vous désirez manipuler.

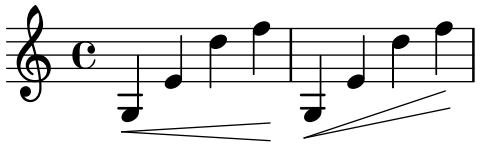
Rotation des objets de mise en forme

Tout objet de rendu disposant de la `grob-interface` est susceptible de pivoter, grâce à la propriété `rotation`. Celle-ci prend en argument une liste de trois éléments : l'angle de rotation – dans le sens inverse des aiguilles d'une montre – ainsi que les coordonnées `x` et `y` du point appartenant à l'objet en question et à partir duquel doit s'effectuer cette rotation. L'angle est exprimé en degrés, les coordonnées en espaces de portée.

L'angle et les coordonnées ne peuvent se déterminer que par tâtonnement.

Il existe assez peu de situation où faire pivoter un objet de mise en forme soit réellement opportun ; en voici une :

```
g4\< e' d'' f''\!
\override Hairpin.rotation = #'(15 -1 0)
g4\< e' d'' f''\!
```
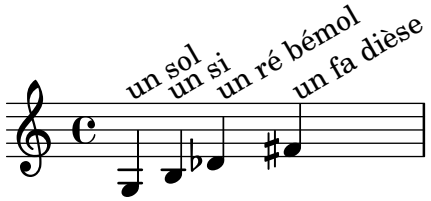


Rotation des étiquettes

Tout texte faisant l'objet d'un *markup* peut pivoter selon n'importe quel angle, à l'aide de la commande `\rotate`. Celle-ci prend deux arguments : l'angle de rotation exprimé en degrés – dans le sens inverse des aiguilles d'une montre – et le texte à basculer. Il ne s'agit pas ici de faire pivoter les extrémités du texte ; celles-ci récupéreront leurs coordonnées x et y du *markup* pivoté. Dans l'exemple ci-dessous, la propriété `outside-staff-priority` à été fixée à `#f` afin de désactiver l'évitement automatique des collisions qui pourrait repousser certains textes trop haut.

```
\override TextScript.outside-staff-priority = ##f
g4^\markup { \rotate #30 "un sol" }
b^\markup { \rotate #30 "un si" }
```

```
des'^\markup { \rotate #30 "un ré bémol" }
fis'^\markup { \rotate #30 "un fa dièse" }
```



5.5 Retouches avancées

Nous allons voir, au fil des paragraphes qui suivent, différentes approches permettant de fignoler l'apparence d'une partition.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "Retouche de partition" dans *Manuel d'initiation*.

Manuel de notation : Section 5.2 [En quoi consiste la référence des propriétés internes], page 648, Section 5.3 [Modification de propriétés], page 651.

Manuel d'extension : Section "Interfaces pour programmeurs" dans *Extension de LilyPond*.

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Retouches" dans *Morceaux choisis*.

Référence des propriétés internes : Section "All layout objects" dans *Référence des propriétés internes*.

5.5.1 Alignement des objets

Les objets graphiques disposant des interfaces `self-alignment-interface` ou `side-position-interface` peuvent s'aligner par rapport à un objet précédemment positionné, ce de différentes manières. Ces objets sont référencés aux rubriques Section "self-alignment-interface" dans *Référence des propriétés internes* et Section "side-position-interface" dans *Référence des propriétés internes*.

Tous les objets graphiques ont un point de référence, une étendue horizontale et une étendue verticale. L'étendue horizontale est représentée par une paire de nombres indiquant l'écart entre le point de référence et les bords gauche et droit – l'écart à gauche étant négatif. L'étendue verticale est représentée par une paire de nombres indiquant l'écart entre le point de référence et les bords inférieur et supérieur – l'écart vers le bas étant négatif.

La position d'un objet sur la portée est donnée par la valeur des propriétés `X-offset` et `Y-offset`. La valeur de `X-offset` indique l'écart en abscisse (coordonnée X) par rapport au point de référence de l'objet parent ; la valeur de `Y-offset` indique l'écart par rapport à la ligne médiane de la portée. Les valeurs de `X-offset` et `Y-offset` peuvent être fournies arbitrairement, ou bien être calculé par des procédures spécifiques qui détermineront l'alignement par rapport à l'objet parent.

> **Note :** Nombre d'objets sont affectés par des considérations spécifiques en matière de positionnement ; jouer sur les valeurs de `X-offset` ou `Y-offset` se révélera inefficace en pareil cas, même si l'objet dispose de la `self-alignment-interface`. Fixer arbitrairement les propriétés `X-offset` ou `Y-offset` annihilera alors les effets de la propriété `self-alignment` correspondante.

Par exemple, une altération peut se repositionner verticalement grâce à son `Y-offset` ; toute modification de son `X-offset` restera par contre sans effet.

Les indications de repère s'alignent sur des objets de rupture – tels les barres de mesure, clefs, métriques et armures. Certaines propriétés spécifiques – dépendant de la `break-aligned-interface` – permettent de gérer le positionnement des indications de repère sur ces objets.

Voir aussi
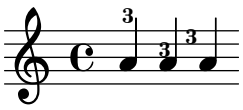
Manuel de notation : [Utilisation de `break-aligned-interface`], page 687.

Manuel d'extension : Section "Fonctions de rappel" dans *Extension de LilyPond*.

Détermination directe de `X-offset` et `Y-offset`

Vous pouvez fournir, pour de nombreux objets, des valeurs numériques aux propriétés `X-offset` et `Y-offset`. Voici par exemple une note avec indication du doigté tout d'abord avec un positionnement par défaut, puis repositionnement par modification successive du `X-offset` et du `Y-offset`.

```
a'-3
a'
-\tweak X-offset #0
-\tweak Y-offset #0
-3
a'
-\tweak X-offset #-1
-\tweak Y-offset #1
-3
```



Utilisation de `side-position-interface`

Un objet disposant de la `side-position-interface` peut se voir accolé à son voisin de telle sorte que les bords des deux objets se touchent. Un tel objet peut se positionner au-dessus, en dessous, à droite ou à gauche de son parent. Ce parent ne saurait être stipulé ; il est déterminé par l'ordre d'apparition des éléments dans le flux des saisies. La plupart de ces objets ont pour parent la tête de note qui leur est associée.

Les valeurs des propriétés `side-axis` et `direction` détermineront l'endroit où viendra se positionner l'objet, selon les préceptes suivants :

Propriété `side-axis`	**Propriété `direction`**	**Positionnement**
`0`	`-1`	gauche
`0`	`1`	droite
`1`	`-1`	en dessous
`1`	`1`	au-dessus

Pour un `side-axis` à `0`, le `X-offset` devrait engager la procédure `ly:side-position-interface::x-aligned-side`. Celle-ci renverra la valeur adéquate de `X-offset` permettant d'accoler l'objet sur la droite ou sur la gauche de son parent, selon la valeur de `direction`.

Pour un `side-axis` à `1`, le `Y-offset` devrait engager la procédure `ly:side-position-interface::y-aligned-side`. Celle-ci renverra la valeur adéquate de `Y-offset` permettant d'accoler l'objet au-dessus ou en dessous de son parent, selon la valeur de `direction`.

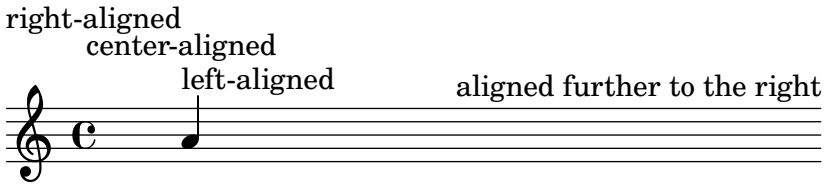
Utilisation de `self-alignment-interface`

Réalignement d'objets horizontalement

L'alignement horizontal d'un objet disposant de la `self-alignment-interface` dépend de la valeur de sa propriété `self-alignment-X`, si tant est que la propriété `X-offset` de cet objet engage la procédure `ly:self-alignment-interface::x-aligned-on-self`. La propriété `self-alignment-X` peut contenir un nombre réel, l'unité de base étant la moitié de l'étendue horizontale de l'objet. Une valeur négative décalera l'objet vers la droite, une valeur positive vers la gauche. La valeur `0` permet de centrer l'objet sur le point de référence de son parent. Une valeur de `-1` alignera le bord gauche de l'objet sur le point de référence de son parent, et une valeur de `1` alignera le bord droit de l'objet sur le point de référence de son parent. Les valeurs symboliques `LEFT`, `CENTER` et `RIGHT` correspondent respectivement à `-1`, `0` et `1`.

En règle générale, la valeur de `self-alignment-X` se modifie à l'aide d'une commande `\override`. Le recours à la commande `\tweak` permet de traiter séparément plusieurs annotations affectées à une même note :

```
a'
-\tweak self-alignment-X #-1
^"left-aligned"
-\tweak self-alignment-X #0
^"center-aligned"
-\tweak self-alignment-X #RIGHT
^"right-aligned"
-\tweak self-alignment-X #-2.5
^"aligned further to the right"
```



Réalignement d'objets verticalement

L'alignement vertical suit le même principe : la propriété `Y-offset` doit alors engager la procédure `ly:self-alignment-interface::y-aligned-on-self`. Toutefois, il arrive bien souvent que d'autres mécanismes interviennent dans l'alignement vertical. La valeur de `Y-offset` n'étant que l'une des variables qui seront prises en compte, l'ajustement pour certains objets peut se révéler fastidieux. L'unité de base est relativement réduite, puisqu'elle est de la moitié de l'étendue verticale de l'objet ; le nombre à fournir en argument pourrait donc être relativement élevé. Une valeur de `-1` alignera le bord inférieur de l'objet sur le point de référence de son parent, et une valeur de `1` alignera le bord supérieur de l'objet sur le point de référence de son parent. La valeur `0` permet de centrer l'objet sur le point de référence de son parent. Les valeurs symboliques `DOWN`, `CENTER` et `UP` correspondent respectivement à `-1`, `0` et `1`.
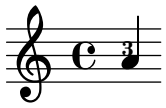
Réalignement d'objets sur les deux axes

Définir à la fois `X-offset` et `Y-offset` permet de réaligner un objet sur les deux axes.

Dans l'exemple ci-dessous, nous ajustons l'indication de doigté de telle sorte qu'elle se place au plus près de la tête de note.

```
a'
-\tweak self-alignment-X #0.5  % move horizontally left
-\tweak Y-offset #ly:self-alignment-interface::y-aligned-on-self
```

```
-\tweak self-alignment-Y #-1  % move vertically up
-3  % third finger
```

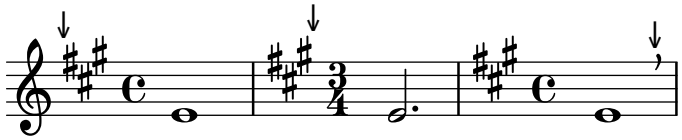


Utilisation de `break-aligned-interface`

Indications de repère et numéros de mesure peuvent s'aligner sur des objets de notation autres qu'une barre de mesure. Parmi ces objets, nous citerons `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature`, et `time-signature`.

Chaque objet possède son propre point de référence par défaut, sur lequel viendront s'aligner les indications de repère :

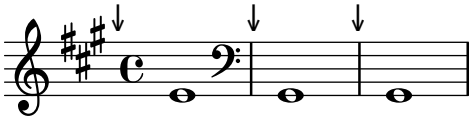
```
% The rehearsal mark will be aligned
% to the right edge of the Clef
\override Score.RehearsalMark.break-align-symbols =
            #'(clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned
% to the left edge of the Time Signature
\override Score.RehearsalMark.break-align-symbols =
            #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark "↓"
e'2.
% The rehearsal mark will be centered
% above the Breath Mark
\override Score.RehearsalMark.break-align-symbols =
            #'(breathing-sign)
\key a \major
\clef treble
\time 4/4
e'1
\breathe
\mark "↓"
```



Les différents objets sur lesquels l'alignement pourrait intervenir seront regroupés dans une liste. Si l'un des objets est invisible à l'endroit voulu, en raison d'un réglage de `break-visibility` ou bien par forçage de la visibilité des armures et clefs, le repère ou le numéro de mesure viendra s'aligner sur le premier élément de cette liste qui soit visible. Dans

le cas où aucun objet de la liste n'est visible, l'alignement se fera sur la barre de mesure ou, dans le cas où la barre de mesure est invisible, à l'endroit même où la barre prendrait place.

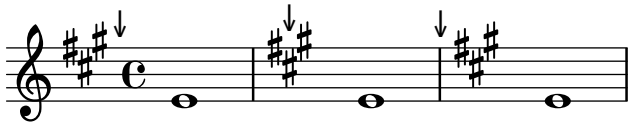
```
% The rehearsal mark will be aligned
% to the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols =
           #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned
% to the right edge of the Clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols =
           #'(key-signature clef)
\key a \major
\clef bass
\mark "↓"
gis,1
% The rehearsal mark will be centered
% above the Bar Line
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols =
           #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
```



L'alignement d'un repère sur un objet de notation peut se modifier, comme l'illustre l'exemple suivant. Toutefois, si la partition comporte plusieurs portées, ce réglage devra apparaître dans chacune des portées.

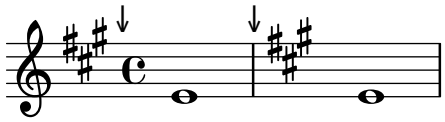
```
% The RehearsalMark will be aligned
% with the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols =
           #'(key-signature)
\key a \major
\clef treble
\time 4/4
\mark "↓"
e'1
% The RehearsalMark will be centered
% above the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment =
                 #CENTER
\mark "↓"
```

```
\key a \major
e'1
% The RehearsalMark will be aligned
% with the left edge of the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment =
                  #LEFT
\key a \major
\mark "↓"
e'1
```



Le bord gauche d'un repère peut se décaler arbitrairement sur la gauche ou sur la droite. La valeur est exprimée en espaces de portée.

```
% The RehearsalMark will be aligned
% with the left edge of the Key Signature
% and then shifted right by 3.5 staff-spaces
\override Score.RehearsalMark.break-align-symbols =
           #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark "↓"
e'1
% The RehearsalMark will be aligned
% with the left edge of the Key Signature
% and then shifted left by 2 staff-spaces
\once \override Score.KeySignature.break-align-anchor = #-2
\key a \major
\mark "↓"
e'1
```



5.5.2 Regroupement vertical d'objets graphiques

Les objets `VerticalAlignment` et `VerticalAxisGroup` travaillent de concert. Comme leur nom anglais l'indiquent, `VerticalAxisGroup` regroupe différents objets tels que les portées (`Staff`), les paroles (`Lyrics`) et ainsi de suite ; puis `VerticalAlignment` synchronise verticalement ces différents groupes. En général, il n'y a qu'un seul `VerticalAlignment` pour l'ensemble de la partition, mais chaque contexte `Staff`, `Lyrics`, etc. possède son propre `VerticalAxisGroup`.

5.5.3 Modification des stencils

Tout objet de rendu dispose d'une propriété `stencil` attachée à la `grob-interface`. En règle générale, cette propriété référence par défaut une fonction spécifique à l'objet et taillée sur mesure pour fournir le symbole qui va le représenter dans l'output. Par exemple, le réglage standard de la propriété `stencil` de l'objet `MultiMeasureRest` est `ly:multi-measure-rest::print`.
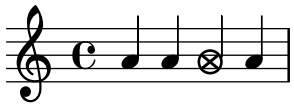
Le symbole standard d'un objet quel qu'il soit peut être remplacé à partir du moment où la propriété `stencil` référence une procédure différente et écrite à cet effet. Ceci requiert une

bonne maîtrise du fonctionnement interne de LilyPond, mais est grandement facilité dans bien des cas et permet d'obtenir le résultat escompté.

En effet, rien ne nous interdit, à partir de la propriété `stencil`, d'appeler la procédure qui génère du texte, `ly:text-interface::print` en l'occurrence, et d'adjoindre à l'objet une propriété `text` qui contiendra, sous forme de *markup*, le symbole à dessein. Grâce à l'extrême flexibilité des *markups*, vous pourrez parvenir à bien des choses – voir à ce sujet [Éléments graphiques dans du texte formaté], page 273.

C'est la technique employée ici, où l'une des têtes de note est remplacée par une croix inscrite dans un cercle :

```
XinO = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph "noteheads.s2cross"
  }
}
\relative {
  a' a \XinO a a
}
```



Tous les glyphes `Feta` de la fonte Emmentaler sont accessibles à l'aide de la commande de *markup* `\musicglyph` – voir Section A.8 [La fonte Emmentaler], page 718.

L'insertion de fichier `EPS` ou d'instructions Postscript sont accessibles par les commandes de *markup* `\epsfile` et `\postscript` respectivement – voir l'annexe Section "Graphisme" dans *Manuel de notation*.

Voir aussi

Manuel de notation : Section A.11 [Commandes pour *markup*], page 746, [Éléments graphiques dans du texte formaté], page 273, Section "Graphisme" dans *Manuel de notation*, Section A.8 [La fonte Emmentaler], page 718, Section 1.8.2 [Mise en forme du texte], page 263.

5.5.4 Modification de l'allure des éléments

Modification des liaisons

Les liaisons, qu'elles soient de prolongation (`Tie`), d'articulation (`Slur`), de phrasé (`PhrasingSlur`), de laisser-vibrer (`LaissezVibrerTie`) ou de reprise (`RepeatTie`), sont dessinées sous la forme de courbes de Bézier de degré trois. Lorsque l'aspect de la liaison automatiquement calculé n'est pas satisfaisant, il peut être modifié manuellement de deux manières différentes :

1. en spécifiant l'ajustement qui doit être apporté aux points de contrôle de la courbe calculée automatiquement, ou
2. en fournissant explicitement les quatre points de contrôle qui permettront de définir cette courbe.

Ces deux méthodes sont expliquées ci-dessous. La première convient mieux dans le cas d'une légère adaptation de la courbe ; la seconde sera plus efficace lorsqu'il s'agira de créer une courbe sur une seule et unique note.
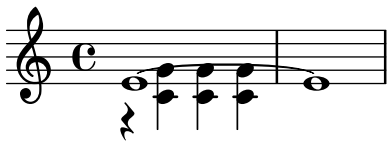
Courbes de Bézier cubiques

Quatre points définissent une courbe de Bézier cubique. Les premier et quatrième points sont les points de départ et d'arrivée de la courbe ; les deux autres points de contrôle – P1 et P2 – en détermineront l'allure. La courbe se trace en partant du point P0, en se dirigeant vers P1 et en arrivant au point P3 selon la direction P2-P3. La courbe est à l'intérieur de l'enveloppe convexe des points de contrôle. Tout déplacement (translation, rotation, échelonnement) des points de contrôle sera répercuté sur le dessin de la courbe.

Spécification de l'ajustement des points de contrôle

Voici par exemple une liaison de prolongation dont l'allure n'est pas des plus heureuses, même en optant pour un `\tieDown`.

```
<<
  { e'1~ 1 }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```



L'ajustement des points de contrôle de cette liaison de tenue à l'aide de `\shape` permet d'éviter les collisions.

L'instruction `\shape` obéit à la syntaxe

`[-]\shape` *`déplacements élément`*

Ceci aura pour effet de repositionner les points de contrôle de *élément* des différents montants fournis par *déplacements*. L'argument *déplacements* est constitué d'une liste de paires de nombres ou bien d'une liste de telles listes. Chacun des membres de l'une des paires indique l'ajustement de la coordonnée d'un point de contrôle. Lorsque *élément* est textuel, il en résulte une dérogation particulière appliquée au type d'objet considéré, alors que dans le cas d'une expression musicale sera appliqué un affinage approprié.
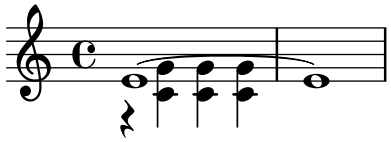
En d'autres termes, la fonction `\shape` se comporte soit comme un `\once \override`, soit comme un `\tweak` selon que l'argument *élément* est un nom d'objet – tel « Slur » – ou une expression musicale tel un « (». L'argument *déplacements* spécifie les ajustements à apporter aux quatre points de contrôle, sous la forme d'une liste de paires `(dx . dy)` dont les valeurs sont exprimées en espace de portée ; on utilisera une liste de listes de paires dans le cas où la courbe comporte plusieurs segments.

La fonction sera précédée d'un tiret si et seulement si elle doit s'appliquer sous forme de `\tweak`.

Pour l'exemple qui nous occupe, l'adaptation sous forme dérogatoire – recours à `\once\override` – de la fonction `\shape`, nous pouvons remonter la liaison d'un demi espace de portée :
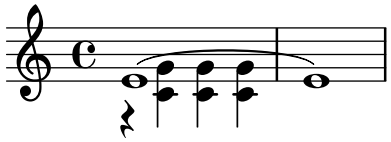
```
<<
  {
    \shape #'((0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie
    e'1~ 1
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
```
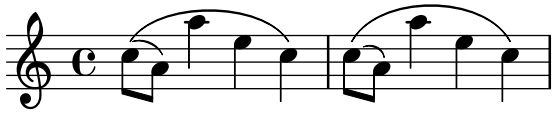
```
>>
```



La liaison est maintenant mieux positionnée ; mais sa partie centrale pourrait être un peu plus relevée, en procédant comme ci-dessous, cette fois en utilisant la formulation d'affinage – la forme `\tweak` :

```
<<
  {
    e'1-\shape #'((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e'
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```
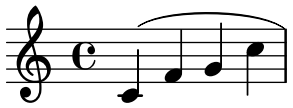


L'adaptation du positionnement horizontal des points de contrôle se réalise de la même manière, ce qui permet de gérer le galbe de deux courbes débutant au même instant musical :
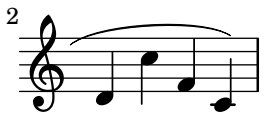
```
\relative {
  c''8(\( a) a'4 e c\)
  \shape #'((0.7 . -0.4) (0.5 . -0.4) (0.3 . -0.3) (0 . -0.2)) Slur
  \shape #'((0 . 0) (0 . 0.5) (0 . 0.5) (0 . 0)) PhrasingSlur
  c8(\( a) a'4 e c\)
}
```



La fonction `\shape` permet aussi d'adapter les points de contrôle d'une courbe qui se prolonge après un saut de ligne. Chaque portion de la courbe peut se voir appliquer sa propre liste d'ajustements. Lorsque l'un des segments ne nécessite pas de retouche, il suffit de lui fournir une liste vide. Dans l'exemple suivant, le saut de ligne laisse à croire qu'il y a non pas une seule mais deux liaisons :
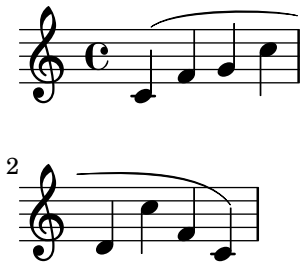
```
\relative {
  c'4( f g c
  \break
  d,4 c' f, c)
}
```
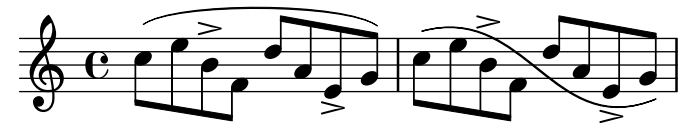




Regalber les deux moitiés de la liaison rend plus évident le fait qu'elle s'étend par delà le saut de ligne :

```
% () may be used as a shorthand for ((0 . 0) (0 . 0) (0 . 0) (0 . 0))
% if any of the segments does not need to be changed
\relative c' {
  \shape #'(
             (( 0 . 0) (0 . 0) (0 . 0) (0 . 1))
             ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
           ) Slur
  c4( f g c
  \break
  d,4 c' f, c)
}
```



La présence d'une courbe en esse requiert obligatoirement d'ajuster manuellement les points de contrôle – LilyPond n'optera jamais automatiquement pour un tel galbe.

```
\relative c'' {
  c8( e b-> f d' a e-> g)
  \shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5))
         PhrasingSlur
  c8\( e b-> f d' a e-> g\)
}
```



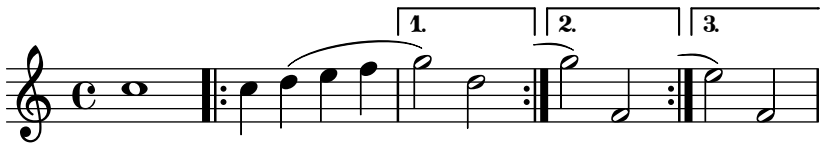
Déclaration explicite des points de contrôle

Les coordonnées des points de contrôle sont données en unités d'espace de portée. L'abscisse est relative au point de référence de la note de départ de la liaison ; l'ordonnée est relative à la ligne médiane de la portée. Les différentes coordonnées sont entrées sous la forme d'une liste de quatre paires de nombres décimaux (ou nombres réels). L'une des manières de procéder consiste à tout d'abord estimer les coordonnées des deux extrémités puis, par tâtonnement, déterminer les deux points intermédiaires. Gardez à l'esprit que ces valeurs pourront devoir être revues si la musique ou sa mise en forme sont modifées.

L'une des situation où spécifier explicitement les points de contrôle se révèle être tout à fait appropriée est lorsqu'ils se réfèrent à une seule et unique note. L'exemple suivant illustre l'un des moyens d'indiquer une liaison qui se prolonge sur les alternatives d'une répétition.

```
\relative {
  c''1
```

```
  \repeat volta 3 { c4 d( e f }
  \alternative {
    \volta 1 { g2) d }
    \volta 2 {
      g2
        % create a slur and move it to a new position
        % the <> is just an empty chord to carry the slur termination
        -\tweak control-points
                  #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <> )
      f,
    }
    \volta 3 {
      e'2
        % create a slur and move it to a new position
        -\tweak control-points
                  #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <> )
      f,
    }
  }
}
```



Problèmes connus et avertissements

Lorsque plusieurs liaisons, quelle qu'en soit la nature, commencent au même moment, jouer sur la propriété `control-points` est impossible, et la commande `\tweak` inefficace. Vous pouvez néanmoins influer sur la propriété `tie-configuration` de l'objet `TieColumn` pour déterminer la ligne de départ et l'orientation.

Voir aussi

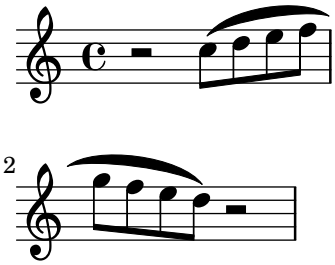
Référence des propriétés internes : Section "TieColumn" dans *Référence des propriétés internes*.

5.5.5 Modification de bandeaux avec rupture

Utilisation de `\alterBroken`

Lorsqu'un bandeau ou l'extension d'un objet rencontre un saut de ligne ou une rupture, chacun de ses tronçons hérite des attributs de l'objet originel. Par voie de conséquence, la modification d'une extension avec rupture produira les mêmes effets sur chacun de ses segments. Dans l'exemple ci-dessous, la modification apportée à `thickness` s'applique aussi bien avant qu'après le saut de ligne.

```
\relative c'' {
  r2
  \once\override Slur.thickness = 10
  c8( d e f
  \break
  g8 f e d) r2
}
```



La commande `\alterBroken` permet de modifier indépendamment l'apparence de chacune des parties d'un bandeau. Selon le cas, cette commande génèrera soit un `\override`, soit un `\tweak` qui s'appliquera à la propriété du bandeau.

La commande `\alterBroken` répond à la syntaxe :

`[-]\alterBroken` *`propriété valeurs élément`*

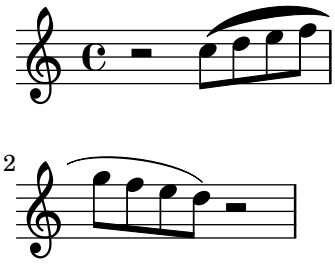
L'argument *valeurs* est constitué d'une liste de valeurs, une pour chaque tronçon. Lorsque *élément* est un nom d'objet graphique, tels `Slur` ou `Staff.PianoPedalBracket`, il en résulte un `\override` du type de *grob* spécifié. Lorsque *élément* est une expression musicale comme « (» ou « [», en résulte cette même expression musicale à laquelle s'applique un `\tweak`.

Le tiret introduisant la commande `\alterBroken` est impératif dans le cadre d'un `\tweak` ; il est superflu pour un `\override`.

Dans sa variante `\override`, la commande `\alterBroken` peut se préfixer d'un `\once` ou d'un `\temporary` qui seront annulés par un `\revert` suivi de la *propriété* – voir Section "Fonctions de substitution intermédiaires" dans *Extension de LilyPond*.

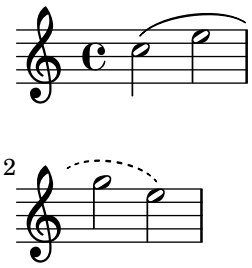
Le code ci-dessous applique un `\override` indépendant à chacun des segments du phrasé de l'exemple précédent :

```
\relative c'' {
  r2
  \alterBroken thickness #'(10 1) Slur
  c8( d e f
  \break
  g8 f e d) r2
}
```



La commande `\alterBroken` peut s'utiliser avec tout objet étendu, y compris `Tie`, `PhrasingSlur`, `Beam` et `TextSpanner`. Par exemple, un éditeur préparant une édition critique pourrait faire ressortir l'absence d'une partie de liaison de phrasé dans l'une des sources, en optant pour un tracé pointillé du seul segment ajouté. L'exemple ci-dessous illustre la manière de procéder, ici avec la variante `\tweak` de la commande :

```
% The empty list is conveniently used below, because it is the
% default setting of dash-definition, resulting in a solid curve.
\relative {
  c''2-\alterBroken dash-definition #'(() ((0 1.0 0.4 0.75))) \(e
  \break
  g2 e\)
}
```



Il est important de considérer que `\alterBroken` affectera à chaque portion d'un bandeau interrompu la valeur correspondante de *valeurs*. Si d'aventure il y a moins de valeurs que de tronçons, toute portion additionnelle se verra assigner une liste vide. Ceci peut conduire à des résultats inattendus dans le cas où la propriété de rendu ne bascule pas sur une liste vide par défaut. En pareil cas, chaque segment devrait se voir assigner une valeur appropriée.

Problèmes connus et avertissements

Les sauts de ligne peuvent intervenir à différents endroits pour répondre à des modifications de mise en forme. Les réglages adoptés par `\alterBroken` peuvent devenir inadaptés si le bandeau n'est plus rompu ou est découpé en plus de segments que prévu. L'introduction explicite d'un `\break` peut alors pallier ces situations.

La commande `\alterBroken` est inopérante sur les propriétés d'un bandeau qui sont traitées avant la procédure de saut de ligne, comme `direction`.

Voir aussi

Manuel d'extension : Section "Retouches complexes" dans *Extension de LilyPond*.

5.5.6 Conteneurs requalifiants

Les conteneurs requalifiants permettent de faciliter le calcul des espacements en cas de modification du *Y-axis* – plus particulièrement les composantes `Y-offset` et `Y-extent` – à l'aide d'une fonction scheme en lieu et place de valeurs.

L'envergure verticale (`Y-extent`) de certains objets dépend de la propriété `stencil` ; jouer sur leur stencil requiert alors une intervention supplémentaire au niveau du `Y-extent` à l'aide d'un conteneur transitoire. Lorsqu'une fonction affecte un `Y-offset` ou un `Y-extent`, cela déclenche la détermination des sauts de ligne de manière anticipée dans la séquence des traitements. Il en résulte que cette opération n'est en fait pas exécutée ; elle renvoie habituellement `0` ou `'(0 . 0)`, ce qui peut engendrer des collisions. Une fonction « pure » évitera d'avorter la construction des propriétés ou objets, qui de ce fait verront leurs arguments liés à la verticalité (`Y-axis`) correctement évalués.

Il existe actuellement une trentaine de fonctions que l'on peut qualifier de « pures ». Le recours à un conteneur transitoire permet de requalifier une fonction de telle sorte qu'elle soit reconnue comme « pure » et soit donc évaluée **avant** détermination des sauts de ligne – l'espacement horizontal sera de fait ajusté en temps et en heure. La fonction « impure » sera ensuite évaluée **après** le positionnement des sauts de ligne.

> **Note :** Il n'est pas toujours facile d'avoir l'assurance qu'une fonction soit qualifiée de « pure » ; aussi nous vous recommandons d'éviter d'utiliser les objets `Beam` ou `VerticalAlignment` lorsque vous désirez en créer une.

Un conteneur requalifiant se construit selon la syntaxe

```
(ly:make-unpure-pure-container f0 f1)
```

où `f0` est une fonction prenant *n* arguments (*n*>=1), le premier devant être l'objet en question ; il s'agit de la fonction dont le résultat sera réutilisé. *f1* est la fonction qui sera qualifiée de « pure ».

Elle prend *n_+_2* arguments, le premier devant être lui aussi l'objet en question, et les second et troisième étant respectivement les « point de départ » (*start*) et « point d'arrivée » (*end*).

start et *end* sont dans tous les cas des valeurs fictives qui trouveront leur utilité dans le cas d'objets de type `Spanner`, tels les soufflets (`Hairpin`) ou barres de ligature (`Beam`), en retournant les différentes estimations de hauteur basées sur leurs début et fin d'extension.

Viennent ensuite les autres arguments de la fonction initiale `f0` – autrement dit aucun si *n_=_1*.

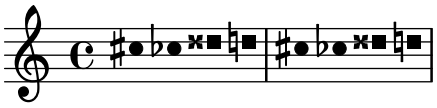
Les résultats de la deuxième fonction (`f1`) permettent une approximation des valeurs qui seront ensuite utilisées par la fonction initiale aux fins d'ajustement lors des phases ultérieures d'espacement.

```
#(define (square-line-circle-space grob)
(let* ((pitch (ly:event-property (ly:grob-property grob 'cause)
                                 'pitch))
       (notename (ly:pitch-notename pitch)))
 (if (= 0 (modulo notename 2))
     (make-circle-stencil 0.5 0.0 #t)
     (make-filled-box-stencil '(0 . 1.0)
                              '(-0.5 . 0.5)))))

squareLineCircleSpace = {
  \override NoteHead.stencil = #square-line-circle-space
}

smartSquareLineCircleSpace = {
  \squareLineCircleSpace
  \override NoteHead.Y-extent =
   #(ly:make-unpure-pure-container
      ly:grob::stencil-height
      (lambda (grob start end) (ly:grob::stencil-height grob)))
}

\new Voice \with { \remove "Stem_engraver" }
\relative c'' {
  \squareLineCircleSpace
  cis4 ces disis d
  \smartSquareLineCircleSpace
  cis4 ces disis d
}
```



La première mesure de l'exemple ci-dessus ne fait pas appel à un conteneur requalifiant ; le moteur d'espacement n'a donc aucune connaissance de la largeur des têtes de note et ne peut empêcher qu'elles chevauchent les altérations. Dans la deuxième mesure, par contre, le recours à un conteneur requalifiant informe le moteur d'espacement de la largeur des têtes de note ; les collisions sont alors évitées du fait de l'espace réservé à chacune des têtes.

Lorsqu'il s'agit de calculs simples, les fonctions, tant pour la partie « pure » que pour la partie « impure », peuvent être identiques au détail près du nombre d'arguments utilisés ou du domaine d'intervention. Ce cas de figure étant relativement répandu, `ly:make-unpure-pure-container`

construira d'elle même cette deuxième lorsqu'il ne sera fait appel qu'à une seule fonction en argument.

> **Note :** Le fait de qualifier une fonction de « pure » alors qu'elle ne l'est pas peut générer des résultats imprévisibles.

5.6 Utilisation de fonctions musicales

Une adaptation ou un affinage qui devient récurrent parce que doit s'appliquer à différentes expressions musicales peut faire l'objet d'une *fonction musicale*. Nous ne traiterons ici que des fonctions de *substitution*, dont le but est de substituer une variable en un bout de code LilyPond. D'autres fonctions, plus complexes, sont abordées au chapitre Section "Fonctions musicales" dans *Extension de LilyPond*.

5.6.1 Syntaxe d'une fonction de substitution

La rédaction d'une fonction chargée de substituer du code LilyPond à une variable est chose relativement aisée. Une telle fonction est de la forme

```
fonction =
#(define-music-function
     (arg1 arg2...)
     (type1? type2?...)
   #{
     ...musique...
   #})
```

où

argN	*n*ième argument.
typeN?	un *type de prédicat* Scheme pour lequel *argN* doit renvoyer `#t`.
...musique...	du code LilyPond tout ce qu'il y a de plus ordinaire, avec des `$` (là où seule une construction LilyPond est autorisée) et des `#` (lorsqu'il s'agit d'une valeur en Scheme, d'un argument de fonction musicale ou de musique faisant partie d'une liste) pour référencer les arguments (par ex. '`#arg1`').

La liste des types de prédicat est aussi obligatoire. Voici quelques uns des types de prédicat les plus utilisés dans les fonctions musicales :

```
boolean?
cheap-list?  (au lieu de « list? », pour accélérer le traitement)
ly:duration?
ly:music?
ly:pitch?
markup?
number?
pair?
string?
symbol?
```

Une liste plus fournie est disponible à l'annexe Section A.21 [Types de prédicats prédéfinis], page 860. Vous pouvez par ailleurs définir vos propres types de prédicat.

Voir aussi

Manuel de notation : Section A.21 [Types de prédicats prédéfinis], page 860.

Manuel d'extension : Section "Fonctions musicales" dans *Extension de LilyPond*.

Fichiers d'initialisation : `lily/music-scheme.cc`, `scm/c++.scm`, `scm/lily.scm`.
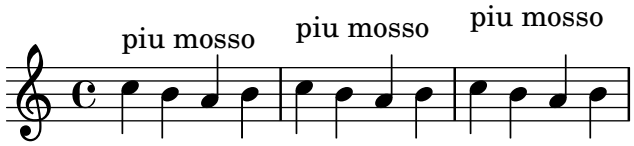
5.6.2 Exemples de fonction de substitution

La présente rubrique regroupe quelques exemples de fonction substitutive. Le propos est ici d'illustrer les possibilités qu'offrent les fonctions de substitution simple.

Dans ce premier exemple, nous définissons une fonction dans le but de simplifier le réglage du décalage d'une annotation (un `TextScript`).

```
padText =
#(define-music-function
     (padding)
     (number?)
   #{
     \once \override TextScript.padding = #padding
   #})
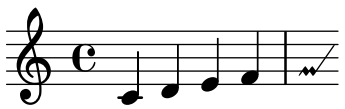
\relative {
  c''4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" b a b
  \padText #2.6
  c4^"piu mosso" b a b
}
```



Nous pouvons utiliser autre chose que des nombres au sein d'une fonction, y compris une expression musicale :

```
custosNote =
#(define-music-function
     (note)
     (ly:music?)
   #{
     \tweak NoteHead.stencil #ly:text-interface::print
     \tweak NoteHead.text
        \markup \musicglyph "custodes.mensural.u0"
     \tweak Stem.stencil ##f
     #note
   #})
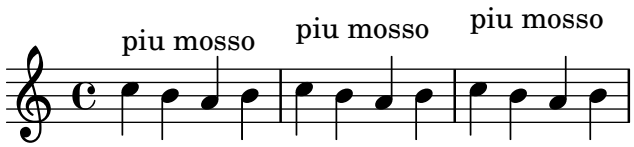
\relative { c'4 d e f \custosNote g }
```



Ces fonctions sont toutes deux des expressions uniques simples dans lesquelles seul le dernier élément d'un appel à une fonction ou une dérogation est absent. Dans ce cas particulier de définition d'une fonction, une syntaxe alternative et plus simple autorise à se cantonner à écrire la partie constante de l'expression et remplacer son dernier élément, absent, par `\etc` :
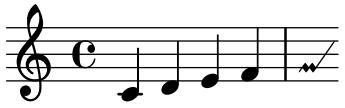
```
padText =
  \once \override TextScript.padding = \etc

\relative {
  c''4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" b a b
  \padText #2.6
  c4^"piu mosso" b a b
}
```




```
custosNote =
  \tweak NoteHead.stencil #ly:text-interface::print
  \tweak NoteHead.text
     \markup \musicglyph "custodes.mensural.u0"
  \tweak Stem.stencil ##f
  \etc

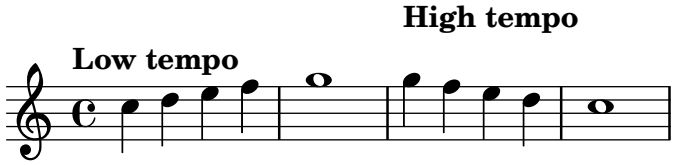
\relative { c'4 d e f \custosNote g }
```



Une fonction de substitution peut traiter plusieurs arguments :

```
tempoPadded =
#(define-music-function
     (padding tempotext)
     (number? markup?)
   #{
     \once \override Score.MetronomeMark.padding = #padding
     \tempo \markup { \bold #tempotext }
   #})

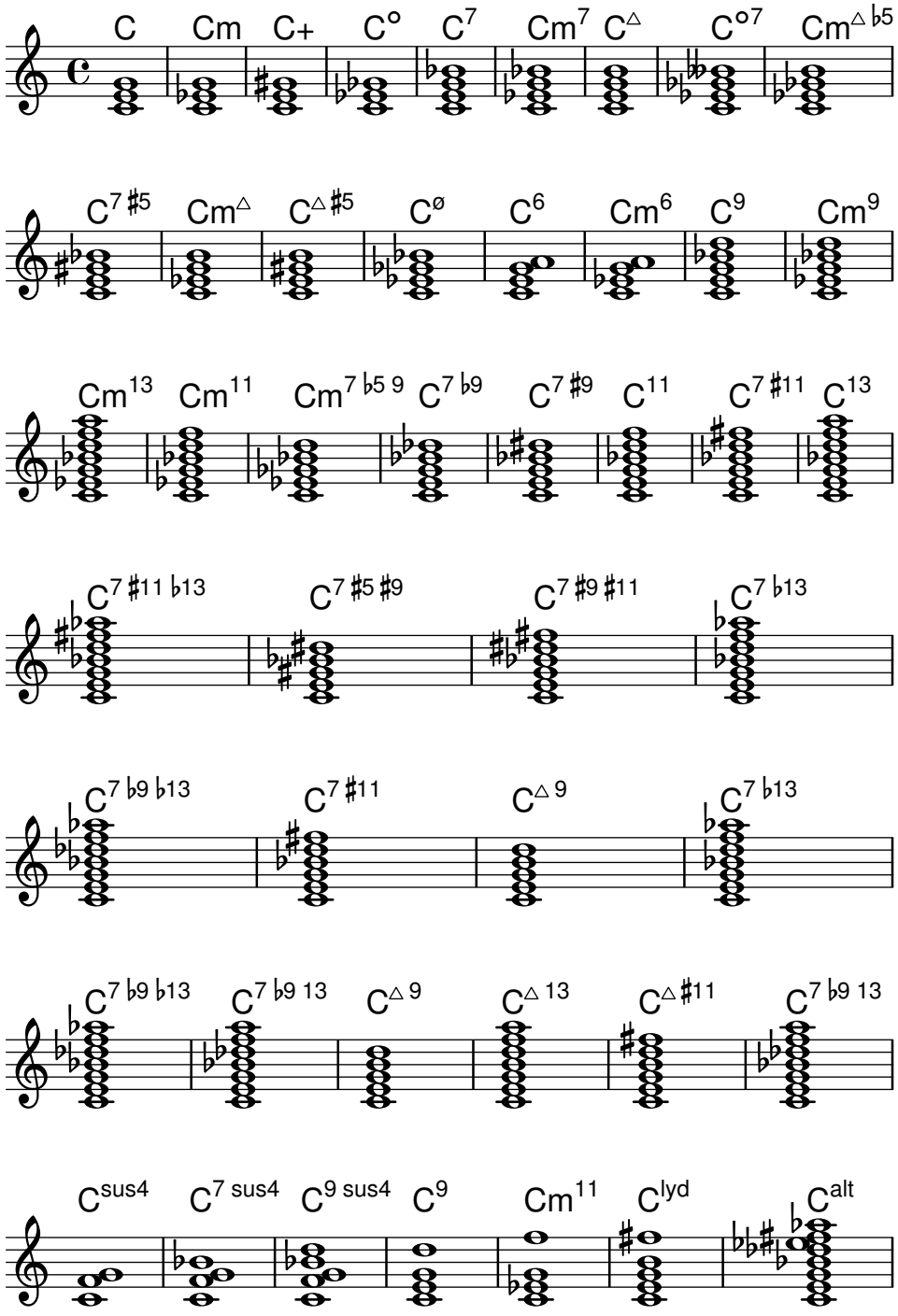
\relative {
  \tempo \markup { "Low tempo" }
  c''4 d e f g1
  \tempoPadded #4.0 "High tempo"
  g4 f e d c1
}
```


High tempo
Low tempo

Annexe A Tables du manuel de notation
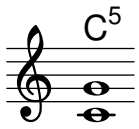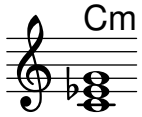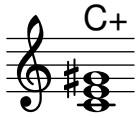
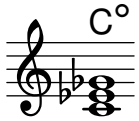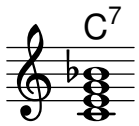
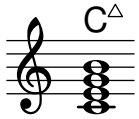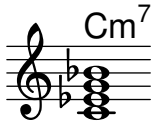
A.1 Table des noms d'accord

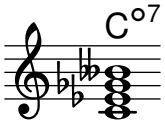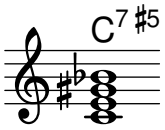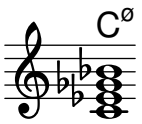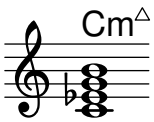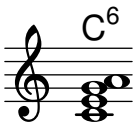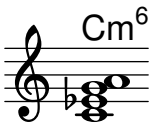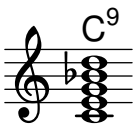
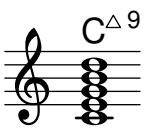
La liste suivante répertorie les noms des accords selon la dénomination standard de LilyPond ainsi que ce qu'ils représentent. D'autres systèmes de nommage, non pris en charge, sont indiqués dans le morceau choisi « Noms d'accords alternatifs » (*chord-names-alternative*) dans Section "Accords" dans *Morceaux choisis*, qui inclut une notation inspirée par Harald Banter (1982) utilisée par défaut dans LilyPond jusqu'à sa version 1.7.
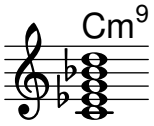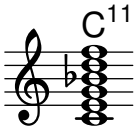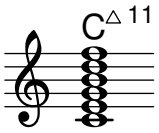
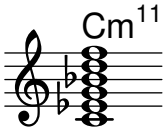
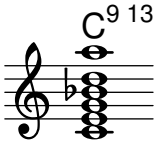
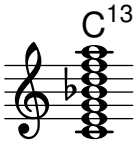
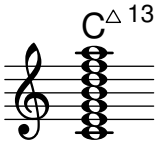
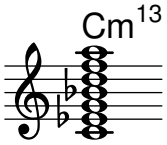


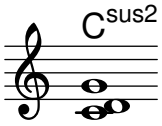
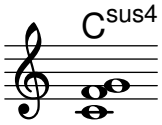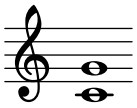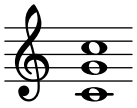
A.2 Modificateurs d'accord usuels

Le tableau suivant indique les différents modificateurs qui permettent d'obtenir les structures habituelles d'un certain nombre d'accords.

Type	**Intervalle**	**Modificateur**	**Exemple**	**Résultat**
Accord parfait majeur (triade majeure)	Tierce majeure et quinte juste	`5` ou rien	`c1:5`	
Accord parfait mineur (triade mineure)	Tierce mineure et quinte juste	`m` ou `m5`	`c1:m`	
Triade augmentée	Tierce majeure et quinte augmentée	`aug`	`c1:aug`	
Triade diminuée	Tierce mineure et quinte diminuée	`dim`	`c1:dim`	
Septième de dominante	Triade majeure et septième mineure	`7`	`c1:7`	
Septième majeure	Triade majeure et septième majeure	`maj7` ou `maj`	`c1:maj7`	
Septième mineure	Triade mineure et septième mineure	`m7`	`c1:m7`	

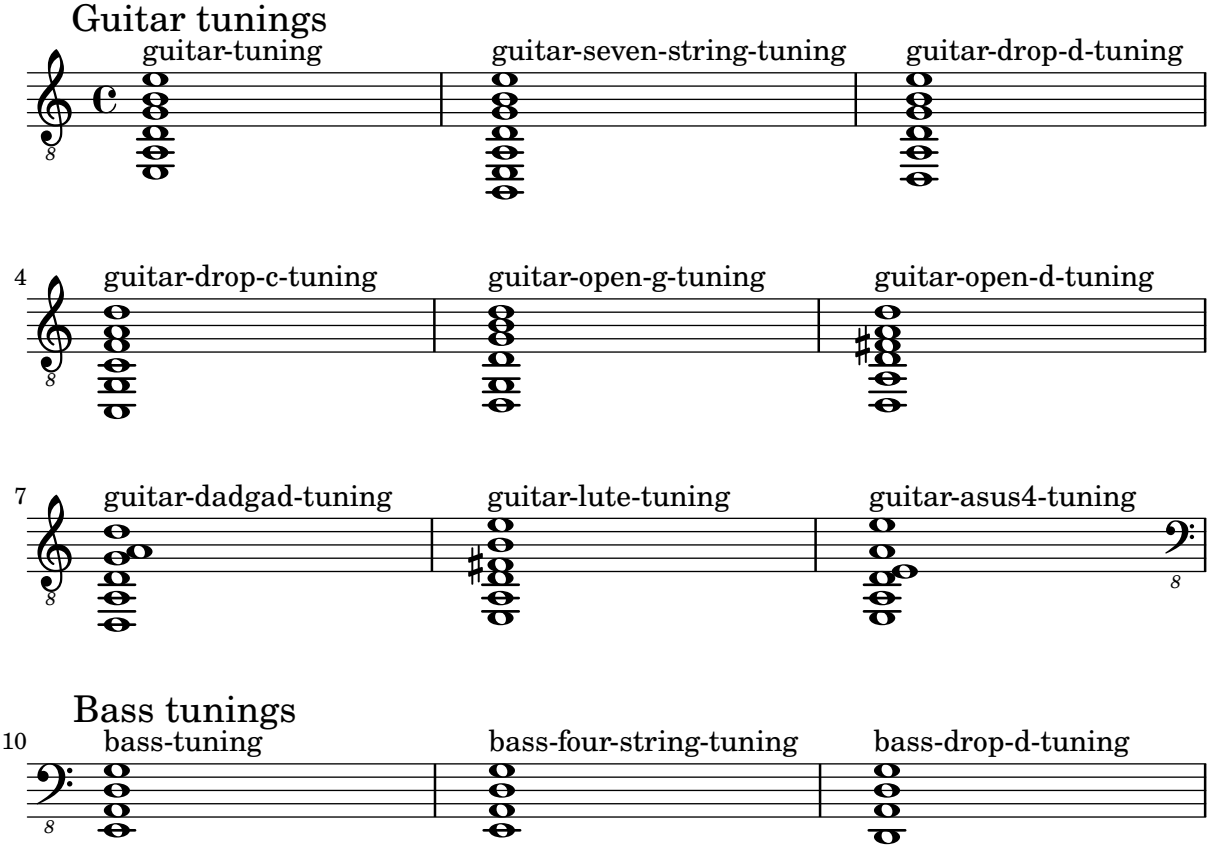
Septième diminuée	Triade diminuée et septième diminuée	`dim7`	`c1:dim7`	
Septième augmentée	Triade augmentée et septième mineure	`aug7`	`c1:aug7`	
Septième semi-diminuée	Triade diminuée et septième mineure	`m7.5-`	`c1:m7.5-`	
Accord mineur avec septième majeure	Triade mineure et septième majeure	`m7+`	`c1:m7+`	
Sixte ajoutée (majeur)	Triade majeure et sixte majeure	`6`	`c1:6`	
Sixte ajoutée (mineur)	Triade mineure et sixte majeure	`m6`	`c1:m6`	
Neuvième de dominante	Septième de dominante et neuvième majeure	`9`	`c1:9`	
Neuvième (majeur)	Septième majeure et neuvième majeure	`maj9`	`c1:maj9`	

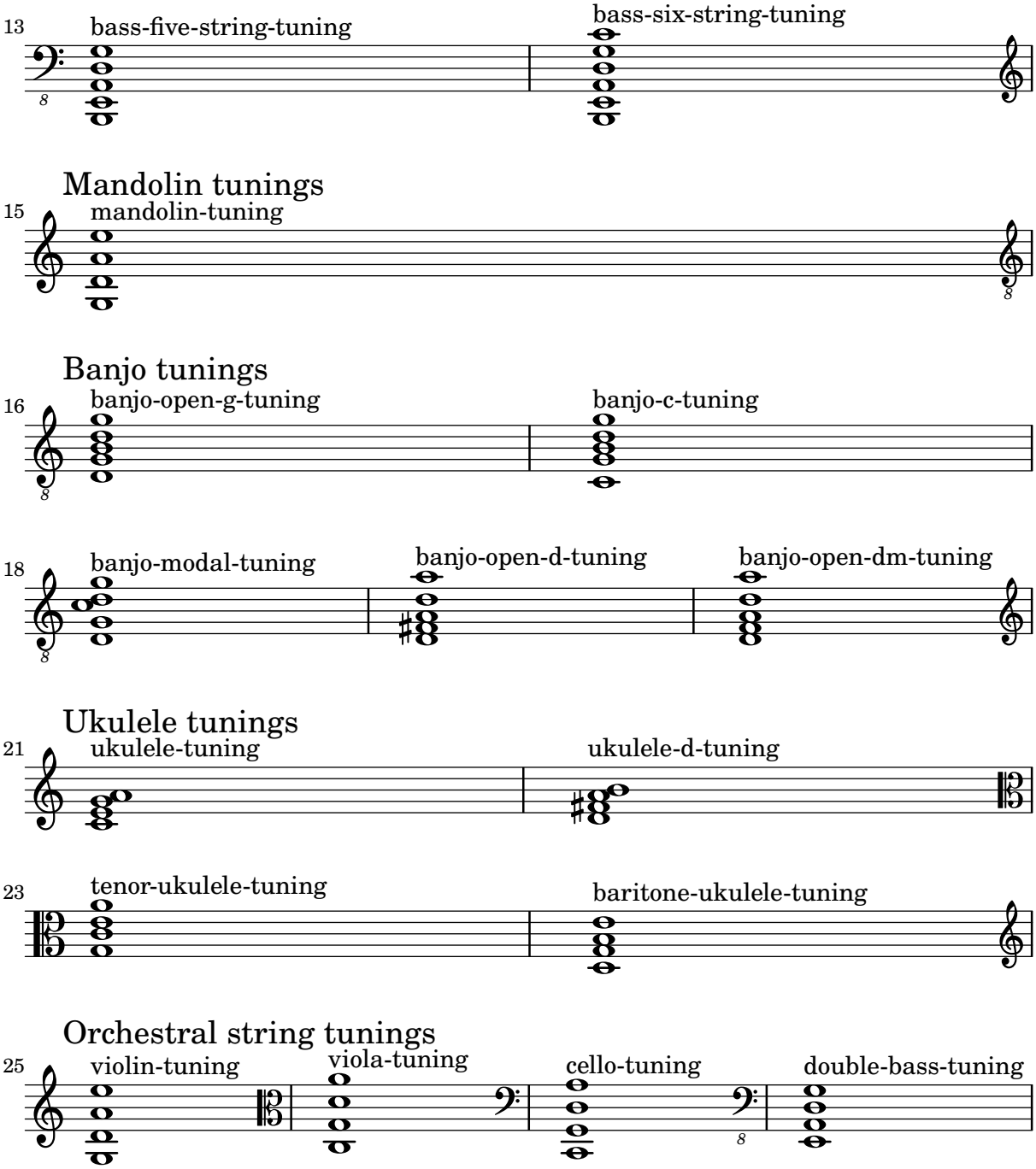
Neuvième (mineur)	Septième mineure et neuvième majeure	`m9`	`c1:m9`	
Onzième de dominante	Neuvième de dominante et onzième juste	`11`	`c1:11`	
Onzième (majeur)	Septième majeure, neuvième majeure et onzième juste	`maj11`	`c1:maj11`	
Onzième (mineur)	Septième mineure, neuvième majeure et onzième juste	`m11`	`c1:m11`	
Treizième de dominante	Neuvième de dominante et treizième majeure	`13`	`c1:13`	
Treizième de dominante	Onzième de dominante et treizième majeure	`13.11`	`c1:13.11`	
Treizième (majeur)	Septième majeure, neuvième, onzième et treizième majeures	`maj13.11`	`c1:maj13.11`	
Treizième (mineur)	Septième mineure, neuvième, onzième et treizième majeures	`m13.11`	`c1:m13.11`	

Seconde suspendue	Seconde majeure et quinte juste	`sus2`	`c1:sus2`	
Quarte suspendue	Quarte juste et quinte juste	`sus4`	`c1:sus4`	
Power chord (deux voix)	Quinte juste	`1.5`	`c1:5`	
Power chord (trois voix)	Quinte juste et octave	`1.5.8`	`c1:5.8`	

A.3 Accordages prédéfinis

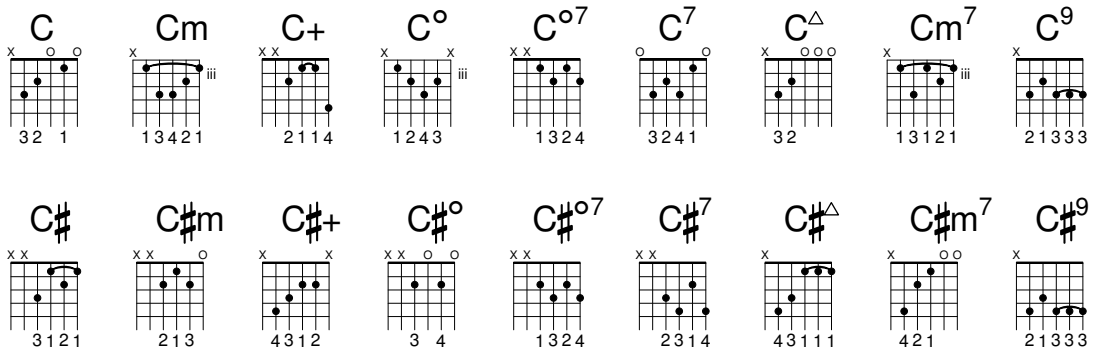
La liste suivante répertorie les différents accordages dont LilyPond dispose.

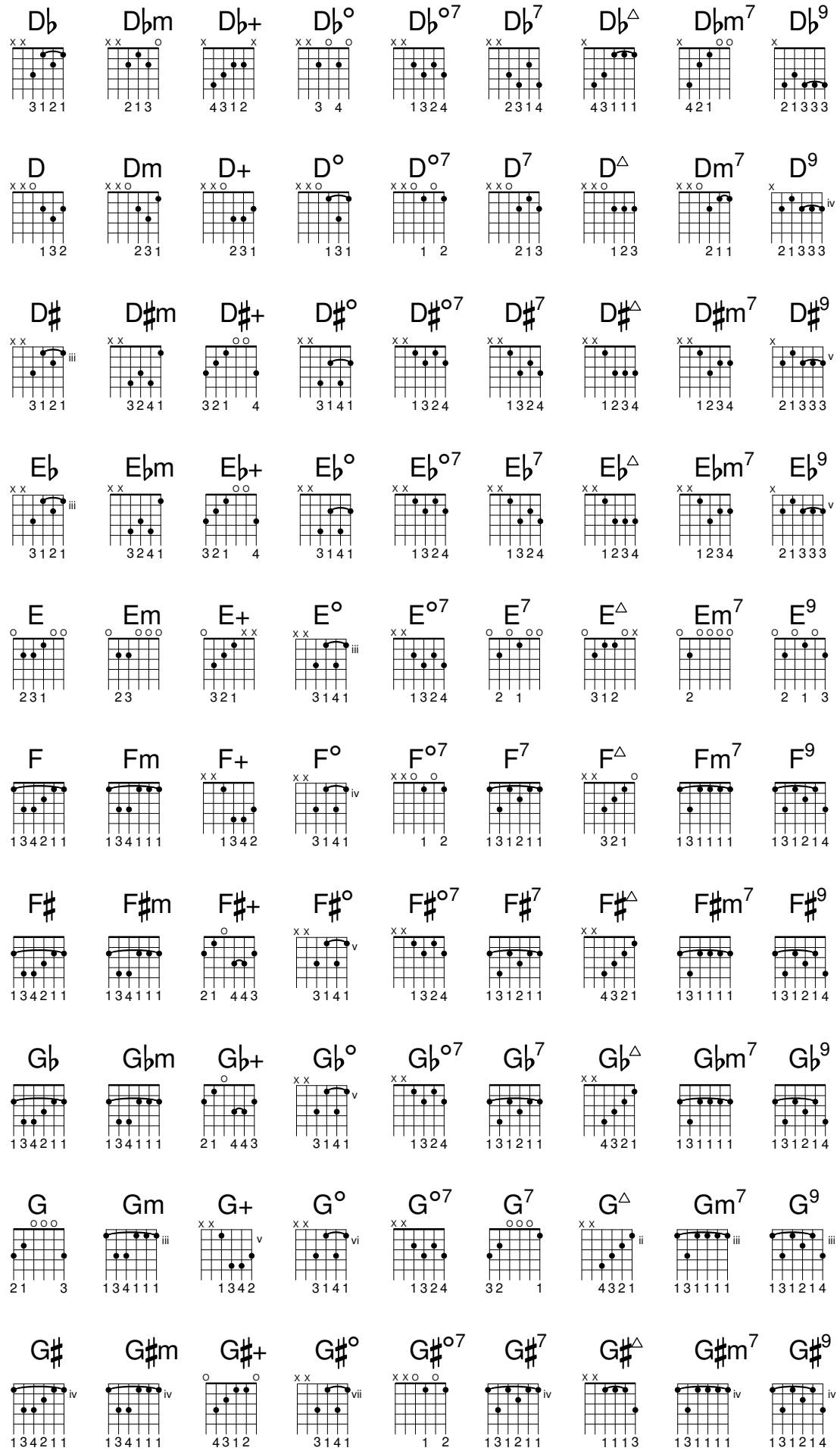


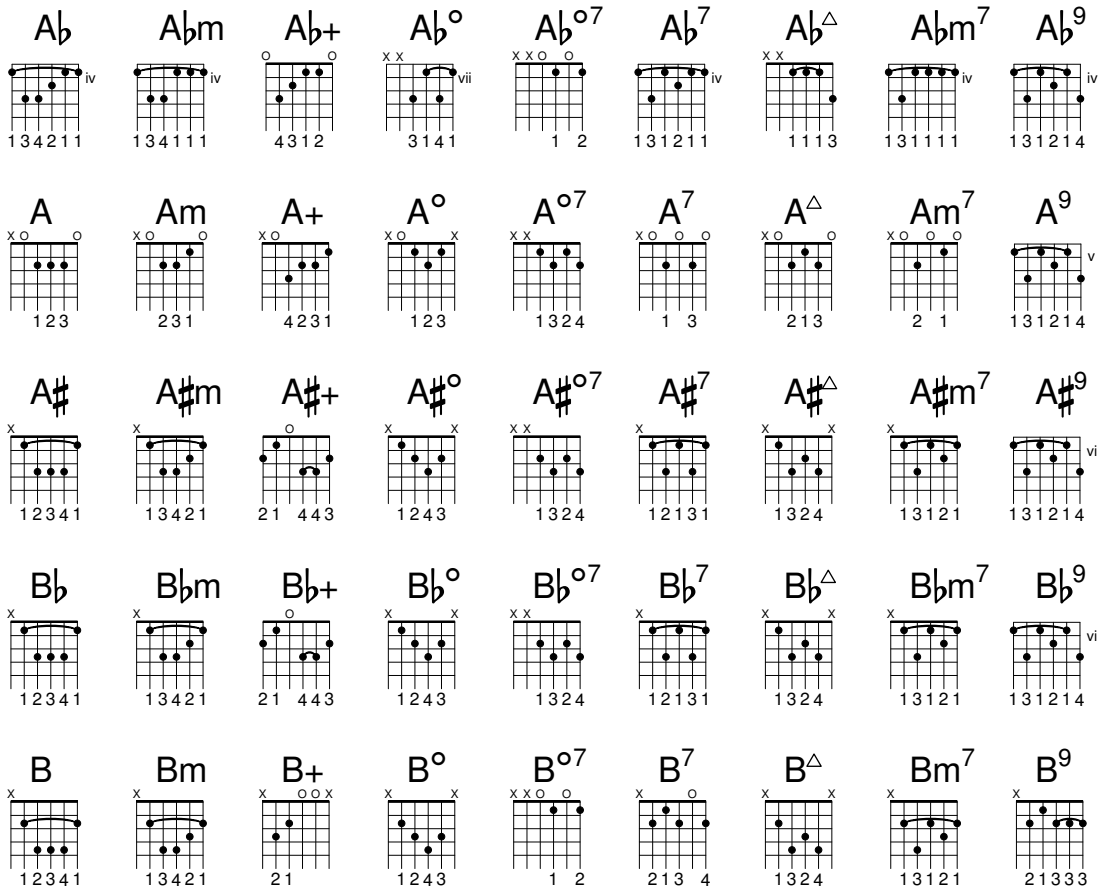


A.4 Diagrammes d'accord prédéfinis

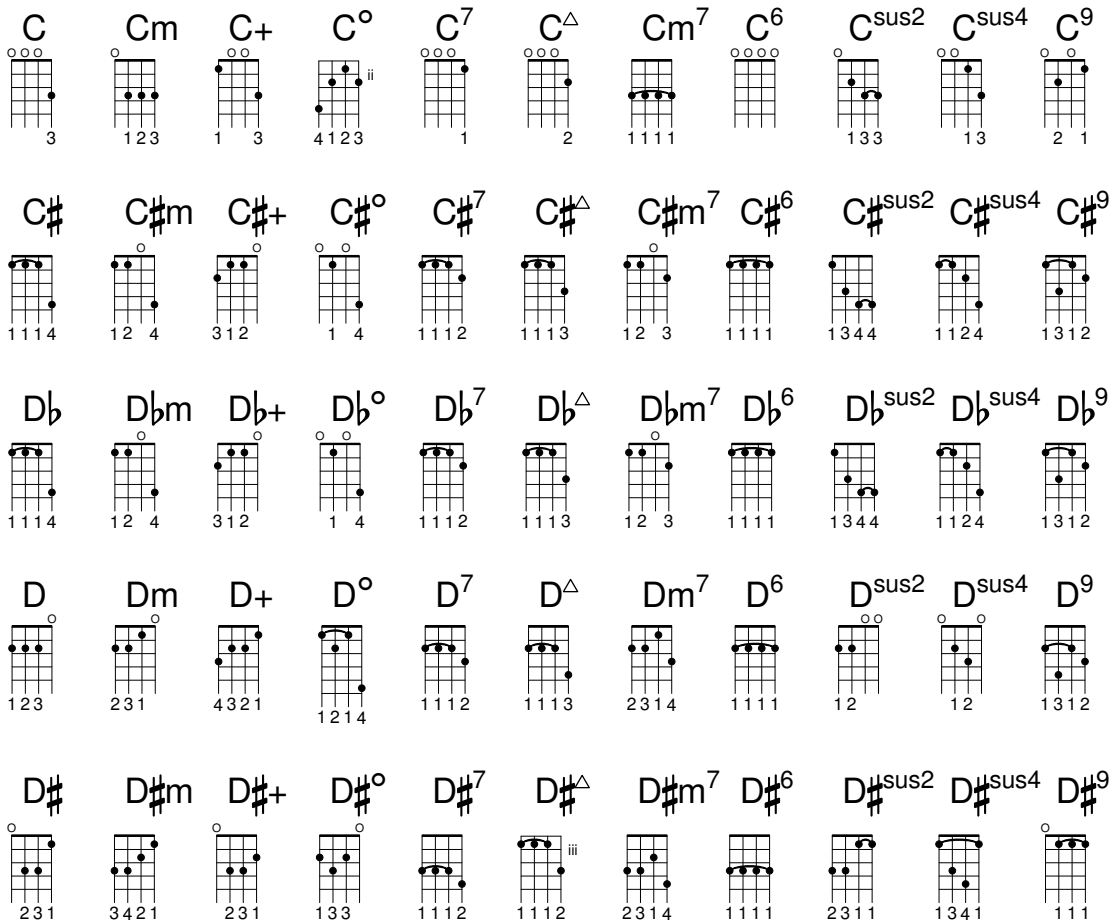
Diagrammes pour guitare

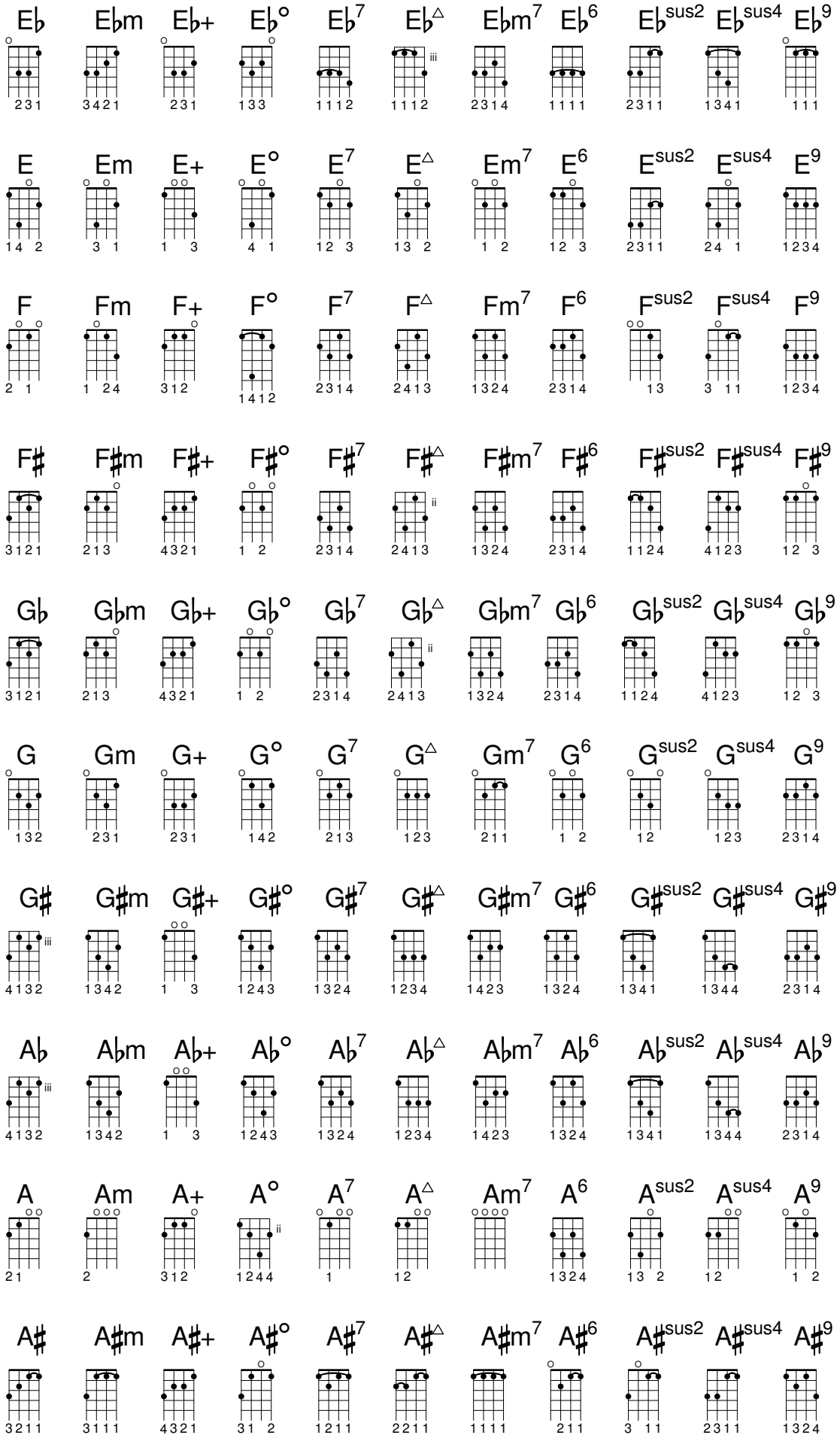


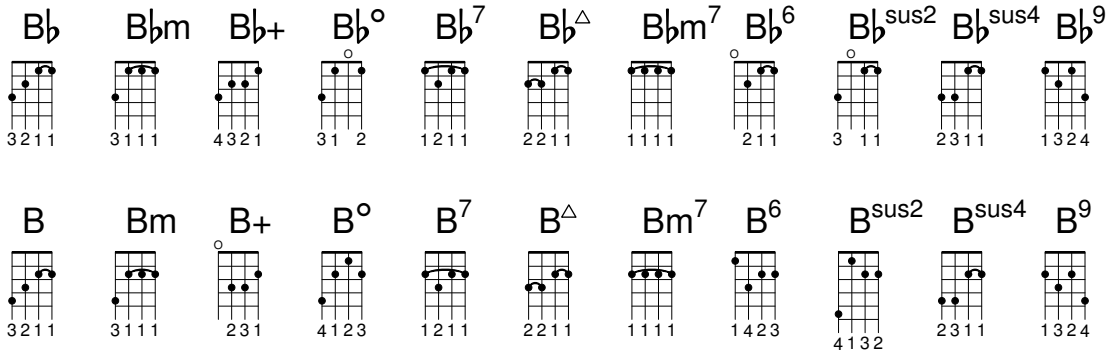

D♭ D♭m D♭+ D♭° D♭°7 D♭7 D♭△ D♭m7 D♭9
D Dm D+ D° D°7 D7 D△ Dm7 D9
D♯ D♯m D♯+ D♯° D♯°7 D♯7 D♯△ D♯m7 D♯9
E♭ E♭m E♭+ E♭° E♭°7 E♭7 E♭△ E♭m7 E♭9
E Em E+ E° E°7 E7 E△ Em7 E9
F Fm F+ F° F°7 F7 F△ Fm7 F9
F♯ F♯m F♯+ F♯° F♯°7 F♯7 F♯△ F♯m7 F♯9
G♭ G♭m G♭+ G♭° G♭°7 G♭7 G♭△ G♭m7 G♭9
G Gm G+ G° G°7 G7 G△ Gm7 G9
G♯ G♯m G♯+ G♯° G♯°7 G♯7 G♯△ G♯m7 G♯9



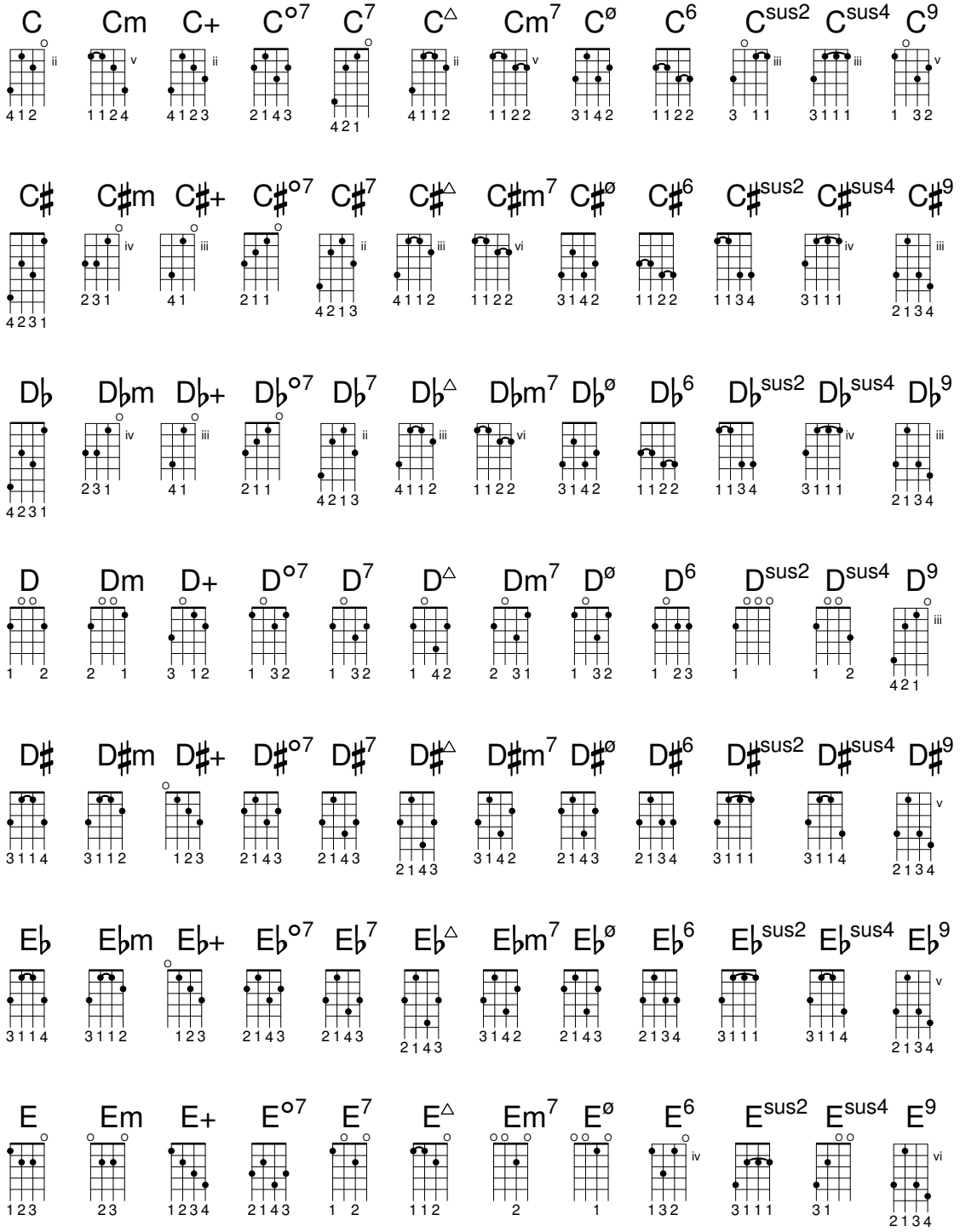
Diagrammes pour ukulele

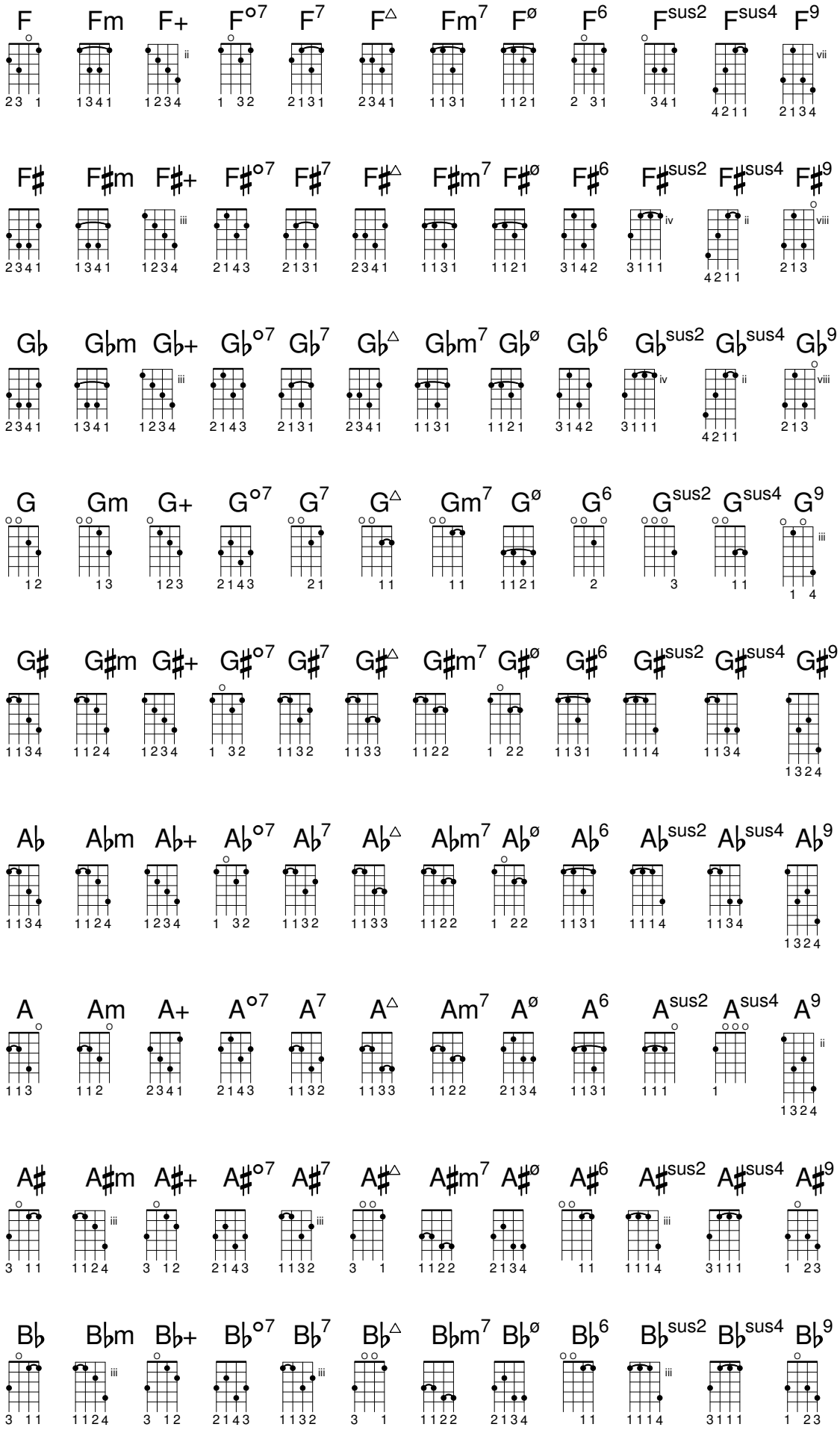


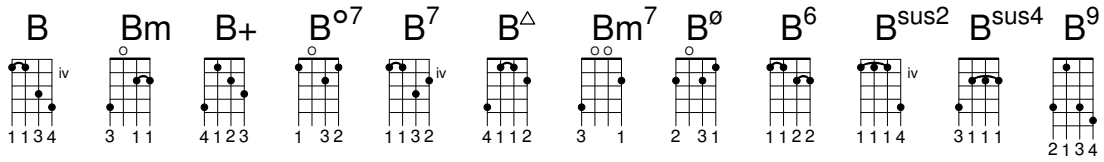

E♭ E♭m E♭+ E♭° E♭7 E♭△ E♭m7 E♭6 E♭sus2 E♭sus4 E♭9
2 3 1 3 4 2 1 2 3 1 1 3 3 1 1 1 2 1 1 1 2 2 3 1 4 1 1 1 2 3 1 1 1 3 4 1 1 1
E Em E+ E° E7 E△ Em7 E6 Esus2 Esus4 E9
1 4 2 3 1 1 3 4 1 1 2 3 1 3 2 1 2 1 2 3 2 3 1 1 2 4 1 1 2 3 4
F Fm F+ F° F7 F△ Fm7 F6 Fsus2 Fsus4 F9
2 1 1 2 4 3 1 2 1 4 1 2 2 3 1 4 2 4 1 3 1 3 2 4 2 3 1 4 1 3 3 1 1 1 2 3 4
F♯ F♯m F♯+ F♯° F♯7 F♯△ F♯m7 F♯6 F♯sus2 F♯sus4 F♯9
3 1 2 1 2 1 3 4 3 2 1 1 2 2 3 1 4 2 4 1 3 1 3 2 4 2 3 1 4 1 1 2 4 4 1 2 3 1 2 3
G♭ G♭m G♭+ G♭° G♭7 G♭△ G♭m7 G♭6 G♭sus2 G♭sus4 G♭9
3 1 2 1 2 1 3 4 3 2 1 1 2 2 3 1 4 2 4 1 3 1 3 2 4 2 3 1 4 1 1 2 4 4 1 2 3 1 2 3
G G m G+ G° G7 G△ Gm7 G6 Gsus2 Gsus4 G9
1 3 2 2 3 1 2 3 1 1 4 2 2 1 3 1 2 1 1 2 1 2 3 2 3 1 4
G♯ G♯m G♯+ G♯° G♯7 G♯△ G♯m7 G♯6 G♯sus2 G♯sus4 G♯9
4 1 3 2 1 3 4 2 1 3 1 2 4 3 1 3 2 4 1 2 3 4 1 4 2 3 1 3 2 4 1 3 4 1 1 3 4 4 2 3 1 4
A♭ A♭m A♭+ A♭° A♭7 A♭△ A♭m7 A♭6 A♭sus2 A♭sus4 A♭9
4 1 3 2 1 3 4 2 1 3 1 2 4 3 1 3 2 4 1 2 3 4 1 4 2 3 1 3 2 4 1 3 4 1 1 3 4 4 2 3 1 4
A A m A+ A° A7 A△ Am7 A6 Asus2 Asus4 A9
2 1 2 3 1 2 1 2 4 4 1 2 1 3 2 4 1 3 2 1 2
A♯ A♯m A♯+ A♯° A♯7 A♯△ A♯m7 A♯6 A♯sus2 A♯sus4 A♯9
3 2 1 1 3 1 1 4 3 2 1 3 1 2 1 2 1 1 2 2 1 1 1 1 2 1 1 3 1 1 2 3 1 1 1 3 2 4



Diagrammes pour mandoline




F Fm F+ F°7 F7 F△ Fm7 Fø F6 Fsus2 Fsus4 F9
F♯ F♯m F♯+ F♯°7 F♯7 F♯△ F♯m7 F♯ø F♯6 F♯sus2 F♯sus4 F♯9
G♭ G♭m G♭+ G♭°7 G♭7 G♭△ G♭m7 G♭ø G♭6 G♭sus2 G♭sus4 G♭9
G Gm G+ G°7 G7 G△ Gm7 Gø G6 Gsus2 Gsus4 G9
G♯ G♯m G♯+ G♯°7 G♯7 G♯△ G♯m7 G♯ø G♯6 G♯sus2 G♯sus4 G♯9
A♭ A♭m A♭+ A♭°7 A♭7 A♭△ A♭m7 A♭ø A♭6 A♭sus2 A♭sus4 A♭9
A Am A+ A°7 A7 A△ Am7 Aø A6 Asus2 Asus4 A9
A♯ A♯m A♯+ A♯°7 A♯7 A♯△ A♯m7 A♯ø A♯6 A♯sus2 A♯sus4 A♯9
B♭ B♭m B♭+ B♭°7 B♭7 B♭△ B♭m7 B♭ø B♭6 B♭sus2 B♭sus4 B♭9



A.5 Formats de papier prédéfinis

Les formats de page sont définis dans le fichier `scm/paper.scm`.

La série A « ISO 216 »

`"a10"`	(26 x 37 mm)
`"a9"`	(37 x 52 mm)
`"a8"`	(52 x 74 mm)
`"a7"`	(74 x 105 mm)
`"a6"`	(105 x 148 mm)
`"a5"`	(148 x 210 mm)
`"a4"`	(210 x 297 mm)
`"a3"`	(297 x 420 mm)
`"a2"`	(420 x 594 mm)
`"a1"`	(594 x 841 mm)
`"a0"`	(841 x 1189 mm)

Deux tailles étendues, définies par la « DIN 476 »

`"4a0"`	(1682 x 2378 mm)
`"2a0"`	(1189 x 1682 mm)

La série B « ISO 216 »

`"b10"`	(31 x 44 mm)
`"b9"`	(44 x 62 mm)
`"b8"`	(62 x 88 mm)
`"b7"`	(88 x 125 mm)
`"b6"`	(125 x 176 mm)
`"b5"`	(176 x 250 mm)
`"b4"`	(250 x 353 mm)
`"b3"`	(353 x 500 mm)
`"b2"`	(500 x 707 mm)
`"b1"`	(707 x 1000 mm)
`"b0"`	(1000 x 1414 mm)

La série C standard « ISO 269 »

`"c10"`	(28 x 40 mm)
`"c9"`	(40 x 57 mm)
`"c8"`	(57 x 81 mm)
`"c7"`	(81 x 114 mm)
`"c6"`	(114 x 162 mm)
`"c5"`	(162 x 229 mm)
`"c4"`	(229 x 324 mm)
`"c3"`	(324 x 458 mm)
`"c2"`	(458 x 648 mm)
`"c1"`	(648 x 917 mm)
`"c0"`	(917 x 1297 mm)

Formats nord américains

`"junior-legal"`	(8.0 x 5.0 in)
`"legal"`	(8.5 x 14.0 in)
`"ledger"`	(17.0 x 11.0 in)
`"letter"`	(8.5 x 11.0 in)

`"tabloid"`	(11.0 x 17.0 in)
`"11x17"`	(11.0 x 17.0 in)
`"17x11"`	(17.0 x 11.0 in)

Government-letter, défini par le *IEEE Printer Working Group*, à l'usage des enfants

`"government-letter"`	(8 x 10.5 in)
`"government-legal"`	(8.5 x 13.0 in)
`"philippine-legal"`	(8.5 x 13.0 in)

Formats ANSI

`"ansi a"`	(8.5 x 11.0 in)
`"ansi b"`	(17.0 x 11.0 in)
`"ansi c"`	(17.0 x 22.0 in)
`"ansi d"`	(22.0 x 34.0 in)
`"ansi e"`	(34.0 x 44.0 in)
`"engineering f"`	(28.0 x 40.0 in)

Formats nord américains pour l'architecture

`"arch a"`	(9.0 x 12.0 in)
`"arch b"`	(12.0 x 18.0 in)
`"arch c"`	(18.0 x 24.0 in)
`"arch d"`	(24.0 x 36.0 in)
`"arch e"`	(36.0 x 48.0 in)
`"arch e1"`	(30.0 x 42.0 in)

Formats anciens, toujours en vigueur dans le Royaume Uni

`"statement"`	(5.5 x 8.5 in)
`"half letter"`	(5.5 x 8.5 in)
`"quarto"`	(8.0 x 10.0 in)
`"octavo"`	(6.75 x 10.5 in)
`"executive"`	(7.25 x 10.5 in)
`"monarch"`	(7.25 x 10.5 in)
`"foolscap"`	(8.27 x 13.0 in)
`"folio"`	(8.27 x 13.0 in)
`"super-b"`	(13.0 x 19.0 in)
`"post"`	(15.5 x 19.5 in)
`"crown"`	(15.0 x 20.0 in)
`"large post"`	(16.5 x 21.0 in)
`"demy"`	(17.5 x 22.5 in)
`"medium"`	(18.0 x 23.0 in)
`"broadsheet"`	(18.0 x 24.0 in)
`"royal"`	(20.0 x 25.0 in)
`"elephant"`	(23.0 x 28.0 in)
`"double demy"`	(22.5 x 35.0 in)
`"quad demy"`	(35.0 x 45.0 in)
`"atlas"`	(26.0 x 34.0 in)
`"imperial"`	(22.0 x 30.0 in)
`"antiquarian"`	(31.0 x 53.0 in)

Formats de base PA4

`"pa0"`	(840 x 1120 mm)
`"pa1"`	(560 x 840 mm)
`"pa2"`	(420 x 560 mm)
`"pa3"`	(280 x 420 mm)
`"pa4"`	(210 x 280 mm)

`"pa5"`	(140 x 210 mm)
`"pa6"`	(105 x 140 mm)
`"pa7"`	(70 x 105 mm)
`"pa8"`	(52 x 70 mm)
`"pa9"`	(35 x 52 mm)
`"pa10"`	(26 x 35 mm)

Format utilisé en Asie du Sudest et en Australie

`"f4"`	(210 x 330 mm)

Format spécifique aux courts exemples `@lilypond` de la documentation, basé sur un A8 à l'italienne.

`"a8landscape"`	(74 x 52 mm)

A.6 Instruments MIDI

La liste suivante répertorie les différentes dénominations que vous pouvez affecter à la propriété `midiInstrument`. L'ordre dans lequel ils sont rangés, par colonne, correspond aux 128 programmes du standard *General MIDI*.

```
acoustic grand
bright acoustic
electric grand
honky-tonk
electric piano 1
electric piano 2
harpsichord
clav
celesta
glockenspiel
music box
vibraphone
marimba
xylophone
tubular bells
dulcimer
drawbar organ
percussive organ
rock organ
church organ
reed organ
accordion
harmonica
concertina
acoustic guitar (nylon)
acoustic guitar (steel)
electric guitar (jazz)
electric guitar (clean)
electric guitar (muted)
overdriven guitar
distorted guitar
guitar harmonics
acoustic bass
electric bass (finger)
contrabass
tremolo strings
pizzicato strings
orchestral harp
timpani
string ensemble 1
string ensemble 2
synthstrings 1
synthstrings 2
choir aahs
voice oohs
synth voice
orchestra hit
trumpet
trombone
tuba
muted trumpet
french horn
brass section
synthbrass 1
synthbrass 2
soprano sax
alto sax
tenor sax
baritone sax
oboe
english horn
bassoon
clarinet
piccolo
flute
recorder
pan flute
blown bottle
lead 7 (fifths)
lead 8 (bass+lead)
pad 1 (new age)
pad 2 (warm)
pad 3 (polysynth)
pad 4 (choir)
pad 5 (bowed)
pad 6 (metallic)
pad 7 (halo)
pad 8 (sweep)
fx 1 (rain)
fx 2 (soundtrack)
fx 3 (crystal)
fx 4 (atmosphere)
fx 5 (brightness)
fx 6 (goblins)
fx 7 (echoes)
fx 8 (sci-fi)
sitar
banjo
shamisen
koto
kalimba
bagpipe
fiddle
shanai
tinkle bell
agogo
steel drums
woodblock
taiko drum
melodic tom
synth drum
reverse cymbal
```

```
electric bass (pick)      shakuhachi          guitar fret noise
fretless bass             whistle             breath noise
slap bass 1               ocarina             seashore
slap bass 2               lead 1 (square)     bird tweet
synth bass 1              lead 2 (sawtooth)   telephone ring
synth bass 2              lead 3 (calliope)   helicopter
violin                    lead 4 (chiff)      applause
viola                     lead 5 (charang)    gunshot
cello                     lead 6 (voice)
```

A.7 Liste des couleurs

Couleurs de base

La syntaxe appropriée à la gestion des couleurs est traitée au chapitre [Coloration d'objets], page 242.

```
black       white          red           green
blue        cyan           magenta       yellow
grey        darkred        darkgreen     darkblue
darkcyan    darkmagenta    darkyellow
```

Noms de couleur CSS

Les noms de couleur CSS peuvent s'utiliser tels quels en argument textuel.

```
aliceblue       darkturquoise       lightsalmon         papayawhip
antiquewhite    darkviolet          lightseagreen       peachpuff
aqua            deeppink            lightskyblue        peru
aquamarine      deepskyblue         lightslategray      pink
azure           dimgray             lightslategrey      plum
beige           dimgrey             lightsteelblue      powderblue
bisque          dodgerblue          lightyellow         purple
black           firebrick           lime                rebeccapurple
blanchedalmond  floralwhite         limegreen           red
blue            forestgreen         linen               rosybrown
blueviolet      fuchsia             magenta             royalblue
brown           gainsboro           maroon              saddlebrown
burlywood       ghostwhite          mediumaquamarine    salmon
cadetblue       gold                mediumblue          sandybrown
chartreuse      goldenrod           mediumorchid        seagreen
chocolate       gray                mediumpurple        seashell
coral           green               mediumseagreen      sienna
cornflowerblue  greenyellow         mediumslateblue     silver
cornsilk        grey                mediumspringgreen   skyblue
crimson         honeydew            mediumturquoise     slateblue
cyan            hotpink             mediumvioletred     slategray
darkblue        indianred           midnightblue        slategrey
darkcyan        indigo              mintcream           snow
darkgoldenrod   ivory               mistyrose           springgreen
darkgray        khaki               moccasin            steelblue
darkgreen       lavender            navajowhite         tan
darkgrey        lavenderblush       navy                teal
arkkhaki       lawngreen           oldlace             thistle
darkmagenta     lemonchiffon        olive               tomato
```

darkolivegreen	lightblue	olivedrab	turquoise
darkorange	lightcoral	orange	violet
darkorchid	lightcyan	orangered	wheat
darkred	lightgoldenrodyellow	orchid	white
darksalmon	lightgray	palegoldenrod	whitesmoke
darkseagreen	lightgreen	palegreen	yellow
darkslateblue	lightgrey	paleturquoise	yellowgreen
darkslategray	lightpink	palevioletred	
darkslategrey			

La définition CSS diffère des noms de couleur X pour les couleurs suivantes : `green`, `grey`, `maroon`, `purple`.

Noms de couleur X

Les noms de couleur X 11 (`https://en.wikipedia.org/wiki/X11_color_names`) offrent un choix plus large que les noms CSS. Ils peuvent s'employer de différentes manières.

- Un nom de couleur peut s'écrire sous la forme d'un mot composé et sans espace auquel chaque élément prend une initiale capitalisée (par ex. `LightSlateBlue`). Il peut aussi s'exprimer sous la forme d'une suite de mots, auquel cas les initiales ne sont pas capitalisées (par ex. `light slate blue`).
- Le *gris* accepte aussi bien l'orthographe « grey » que « gray » (par ex. `DarkSlateGray`).
- Certains noms peuvent prendre un suffixe numérique, comme `LightSalmon4`.

Les listes suivantes présentent tous les noms de couleur disponibles, sans adjonction de suffixe numérique, puis celles acceptant un suffixe.

Noms de couleur sans suffixe numérique

AliceBlue	LawnGreen	OrangeRed	firebrick
AntiqueWhite	LemonChiffon	PaleGoldenrod	gainsboro
BlanchedAlmond	LightBlue	PaleGreen	gold
BlueViolet	LightCoral	PaleTurquoise	goldenrod
CadetBlue	LightCyan	PaleVioletRed	green
CornflowerBlue	LightGoldenrod	PapayaWhip	grey
DarkBlue	LightGoldenrodYellow	PeachPuff	honeydew
DarkCyan	LightGreen	PowderBlue	ivory
DarkGoldenrod	LightGrey	RosyBrown	khaki
DarkGreen	LightPink	RoyalBlue	lavender
DarkGrey	LightSalmon	SaddleBrown	linen
DarkKhaki	LightSeaGreen	SandyBrown	magenta
DarkMagenta	LightSkyBlue	SeaGreen	maroon
DarkOliveGreen	LightSlateBlue	SkyBlue	moccasin
DarkOrange	LightSlateGrey	SlateBlue	navy
DarkOrchid	LightSteelBlue	SlateGrey	orange
DarkRed	LightYellow	SpringGreen	orchid
DarkSalmon	LimeGreen	SteelBlue	peru
DarkSeaGreen	MediumAquamarine	VioletRed	pink
DarkSlateBlue	MediumBlue	WhiteSmoke	plum
DarkSlateGrey	MediumOrchid	YellowGreen	purple
DarkTurquoise	MediumPurple	aquamarine	red
DarkViolet	MediumSeaGreen	azure	salmon
DeepPink	MediumSlateBlue	beige	seashell
DeepSkyBlue	MediumSpringGreen	bisque	sienna

```
DimGrey          MediumTurquoise       black          snow
DodgerBlue       MediumVioletRed       blue           tan
FloralWhite      MidnightBlue          brown          thistle
ForestGreen      MintCream             burlywood      tomato
GhostWhite       MistyRose             chartreuse     turquoise
reenYellow      NavajoWhite           chocolate      violet
HotPink          NavyBlue              coral          wheat
+IndianRed        OldLace               cornsilk       white
LavenderBlush    OliveDrab             cyan           yellow
```

Noms de couleur avec suffixe numérique

Les couleurs suivantes acceptent un suffixe entier numérique N compris entre 1 et 4.

```
AntiqueWhiteN    LightSkyBlueN    SteelBlueN    khakiN
CadetBlueN       LightSteelBlueN  VioletRedN    magentaN
DarkGoldenrodN   LightYellowN     aquamarineN   maroonN
DarkOliveGreenN  MediumOrchidN    azureN        orangeN
DarkOrangeN      MediumPurpleN    bisqueN       orchidN
DarkOrchidN      MistyRoseN       blueN         pinkN
DarkSeaGreenN    NavajoWhiteN     brownN        plumN
DeepPinkN        OliveDrabN       burlywoodN    purpleN
DeepSkyBlueN     OrangeRedN       chartreuseN   redN
DodgerBlueN      PaleGreenN       chocolateN    salmonN
HotPinkN         PaleTurquoiseN   coralN        seashellN
IndianRedN       PaleVioletRedN   cornsilkN     siennaN
LavenderBlushN   PeachPuffN       cyanN         snowN
LemonChiffonN    RosyBrownN       firebrickN    tanN
LightBlueN       RoyalBlueN       goldN         thistleN
LightCyanN       SeaGreenN        goldenrodN    tomatoN
LightGoldenrodN  SkyBlueN         greenN        turquoiseN
LightPinkN       SlateBlueN       honeydewN     wheatN
LightSalmonN     SpringGreenN     ivoryN        yellowN
```

Échelle de gris

Une échelle de gris s'obtient en utilisant

```
greyN
```

où N est compris entre 0 et 100.

A.8 La fonte Emmentaler

La fonte Emmentaler est constituée de deux *jeux* de glyphes : « Feta » est utilisé pour la notation classique, et « Parmesan » pour la notation ancienne.

Les différents symboles – ou glyphes – de la fonte Emmentaler peuvent s'inclure directement dans un objet *markup*. Pour ce faire, il suffit d'employer le nom du glyphe (voir les tables ci-après) comme ceci :

```
g^\markup { \musicglyph "scripts.segno" }
```

ou

```
\markup { \musicglyph "five" }
```

Pour de plus amples informations, reportez-vous au chapitre Section 1.8.2 [Mise en forme du texte], page 263.

Glyphes de clef

clefs.C		clefs.C_change	
clefs.varC		clefs.varC_change	
clefs.F		clefs.F_change	
clefs.G		clefs.G_change	
clefs.GG		clefs.GG_change	
clefs.tenorG		clefs.tenorG_change	
clefs.percussion		clefs.percussion_change	
clefs.varpercussion		clefs .varpercussion_change	
clefs.tab		clefs.tab_change	

Glyphes de métrique

timesig.C44		timesig.C22	

Glyphes de chiffre

plus	+	comma	,
hyphen	-	period	.

zero		one	
two		three	
four		five	
six		seven	
eight		nine	

Glyphes d'altération

accidentals.sharp		accidentals .sharp.arrowup	
accidentals .sharp.arrowdown		accidentals .sharp.arrowboth	
accidentals.sharp .slashslash.stem		accidentals.sharp .slashslashslash.stemstem	
accidentals.sharp .slashslashslash.stem		accidentals .sharp.slash.stem	
accidentals.sharp .slashslash.stemstemstem		accidentals.doublesharp	
accidentals.natural		accidentals .natural.arrowup	
accidentals .natural.arrowdown		accidentals .natural.arrowboth	
accidentals.flat		accidentals.flat.arrowup	
accidentals .flat.arrowdown		accidentals .flat.arrowboth	
accidentals.flat.slash		accidentals.flat .slashslash	

accidentals.mirroredflat.flat 

accidentals.mirroredflat 

accidentals.mirroredflat.backslash 

accidentals.flatflat

accidentals.flatflat.slash 

accidentals.rightparen)

accidentals.leftparen (

Glyphes de tête de note par défaut

noteheads.uM2 

noteheads.dM2 

noteheads.sM1 

noteheads.s0 

noteheads.s1 

noteheads.s2 

Glyphes de tête de note spéciale

noteheads.sM1double 

noteheads.s0diamond 

noteheads.s1diamond 

noteheads.s2diamond 

noteheads.s0triangle 

noteheads.s1triangle 

noteheads.s2triangle 

noteheads.s0slash 

noteheads.s1slash 

noteheads.s2slash 

noteheads.s0cross 

noteheads.s1cross 

noteheads.s2cross 

noteheads.s2xcircle 

noteheads.s0harmonic 

noteheads.s2harmonic 

Glyphes de tête de note à forme variable

noteheads.s0do		noteheads.s1do	
noteheads.s2do		noteheads.s0doThin	
noteheads.s1doThin		noteheads.s2doThin	
noteheads.s0re		noteheads.s1re	
noteheads.s2re		noteheads.s0reThin	
noteheads.s1reThin		noteheads.s2reThin	
noteheads.s0mi		noteheads.s1mi	
noteheads.s2mi		noteheads.s0miMirror	
noteheads.s1miMirror		noteheads.s2miMirror	
noteheads.s0miThin		noteheads.s1miThin	
noteheads.s2miThin		noteheads.u0fa	
noteheads.d0fa		noteheads.u1fa	
noteheads.d1fa		noteheads.u2fa	
noteheads.d2fa		noteheads.u0faThin	

noteheads.d0faThin		noteheads.u1faThin	
noteheads.d1faThin		noteheads.u2faThin	
noteheads.d2faThin		noteheads.s0sol	
noteheads.s1sol		noteheads.s2sol	
noteheads.s0la		noteheads.s1la	
noteheads.s2la		noteheads.s0laThin	
noteheads.s1laThin		noteheads.s2laThin	
noteheads.s0ti		noteheads.s1ti	
noteheads.s2ti		noteheads.s0tiThin	
noteheads.s1tiThin		noteheads.s2tiThin	
noteheads.u0doFunk		noteheads.d0doFunk	
noteheads.u1doFunk		noteheads.d1doFunk	
noteheads.u2doFunk		noteheads.d2doFunk	
noteheads.u0reFunk		noteheads.d0reFunk	
noteheads.u1reFunk		noteheads.d1reFunk	

noteheads.u2reFunk		noteheads.d2reFunk	
noteheads.u0miFunk		noteheads.d0miFunk	
noteheads.u1miFunk		noteheads.d1miFunk	
noteheads.s2miFunk		noteheads.u0faFunk	
noteheads.d0faFunk		noteheads.u1faFunk	
noteheads.d1faFunk		noteheads.u2faFunk	
noteheads.d2faFunk		noteheads.s0solFunk	
noteheads.s1solFunk		noteheads.s2solFunk	
noteheads.s0laFunk		noteheads.s1laFunk	
noteheads.s2laFunk		noteheads.u0tiFunk	
noteheads.d0tiFunk		noteheads.u1tiFunk	
noteheads.d1tiFunk		noteheads.u2tiFunk	
noteheads.d2tiFunk		noteheads.s0doWalker	
noteheads.u1doWalker		noteheads.d1doWalker	
noteheads.u2doWalker		noteheads.d2doWalker	

noteheads.s0reWalker		noteheads.u1reWalker	
noteheads.d1reWalker		noteheads.u2reWalker	
noteheads.d2reWalker		noteheads.s0miWalker	
noteheads.s1miWalker		noteheads.s2miWalker	
noteheads.s0faWalker		noteheads.u1faWalker	
noteheads.d1faWalker		noteheads.u2faWalker	
noteheads.d2faWalker		noteheads.s0laWalker	
noteheads.s1laWalker		noteheads.s2laWalker	
noteheads.s0tiWalker		noteheads.u1tiWalker	
noteheads.d1tiWalker		noteheads.u2tiWalker	
noteheads.d2tiWalker			

Glyphes de silence

rests.0		rests.1	
rests.0o		rests.1o	
rests.M3		rests.M2	
rests.M1		rests.M1o	

rests.2 

rests.2classical 

rests.2z 

rests.3 
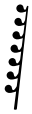
rests.4 

rests.5 

rests.6 

rests.7 

rests.8 

rests.9 

rests.10 

Glyphes de crochet de croche

flags.u3 

flags.u4 

flags.u5 

flags.u6 

flags.u7 

flags.u8 

flags.u9 

flags.u10 

flags.d3 

flags.d4 

flags.d5 

flags.d6 

flags.d7 

flags.d8 

flags.d9 

flags.d10 

flags.ugrace 

flags.dgrace 

Glyphes de point

dots.dot

Glyphes de nuance

space

f 

m

n

p

r

s

z

Glyphes de script

scripts.ufermata 

scripts.dfermata 

scripts
.uhenzeshortfermata 

scripts
.dhenzeshortfermata 

scripts
.uhenzelongfermata 

scripts
.dhenzelongfermata 

scripts.ushortfermata 

scripts.dshortfermata 

scripts
.uveryshortfermata 

scripts
.dveryshortfermata 

scripts.ulongfermata		scripts.dlongfermata	
scripts.uverylongfermata		scripts.dverylongfermata	
scripts.thumb		scripts.sforzato	
scripts.espr		scripts.staccato	
scripts.ustaccatissimo		scripts.dstaccatissimo	
scripts.tenuto		scripts.uportato	
scripts.dportato		scripts.umarcato	
scripts.dmarcato		scripts.open	
scripts.halfopen		scripts.halfopenvertical	
scripts.stopped		scripts.upbow	
scripts.downbow		scripts.reverseturn	
scripts.turn		scripts.slashturn	
scripts.haydnturn		scripts.trill	
scripts.upedalheel		scripts.dpedalheel	
scripts.upedaltoe		scripts.dpedaltoe	

scripts.flageolet 

scripts.segno 

scripts.varsegno 

scripts.coda 

scripts.varcoda 

scripts.rcomma

scripts.lcomma

scripts.rvarcomma 

scripts.lvarcomma 

scripts.arpeggio

scripts.trill_element

scripts.arpeggio
.arrow.M1

scripts.arpeggio.arrow.1 

scripts.trillelement

scripts.prall

scripts.mordent 

scripts.prallprall

scripts.prallmordent 

scripts.upprall 

scripts.upmordent 

scripts.prallup 

scripts.downprall 

scripts.downmordent 

scripts.pralldown 

scripts.lineprall 

scripts.caesura.curved 

scripts.caesura.straight 

scripts.tickmark

scripts.snappizzicato 

scripts.ictus

scripts.uaccentus

scripts.daccentus

scripts.usemicirculus

scripts.dsemicirculus

scripts.circulus

scripts.augmentum

scripts.usignumcongruentiae

scripts.dsignumcongruentiae

Glyphes de flèche

arrowheads.open.01

arrowheads.open.0M1

arrowheads.open.11

arrowheads.open.1M1

arrowheads.close.01

arrowheads.close.0M1

arrowheads.close.11

arrowheads.close.1M1

Glyphes d'extrémité d'accolade

brackettips.up

brackettips.down

Glyphes de pédale

pedal.*

pedal.M

pedal..

pedal.P

pedal.d

pedal.e

pedal.Ped

Glyphes d'accordéon

accordion.discant		accordion.dot	
accordion.freebass		accordion.stdbass	
accordion.bayanbass		accordion.oldEE	
accordion.push		accordion.pull	

Glyphes de liaison

ties.lyric.short		ties.lyric.default	

Glyphes de style vaticana

clefs.vaticana.do		clefs.vaticana.do_change	
clefs.vaticana.fa		clefs.vaticana.fa_change	
custodes.vaticana.u0		custodes.vaticana.u1	
custodes.vaticana.u2		custodes.vaticana.d0	
custodes.vaticana.d1		custodes.vaticana.d2	
accidentals.vaticanaM1		accidentals.vaticana0	
dots.dotvaticana		noteheads .svaticana.punctum	
noteheads.svaticana .punctum.cavum		noteheads.svaticana .linea.punctum	

noteheads.svaticana .linea.punctum.cavum		noteheads.svaticana .inclinatum	
noteheads.svaticana.lpes		noteheads .svaticana.vlpes	
noteheads.svaticana.upes		noteheads .svaticana.vupes	
noteheads .svaticana.plica		noteheads .svaticana.vplica	
noteheads .svaticana.epiphonus		noteheads.svaticana .vepiphonus	
noteheads.svaticana .reverse.plica		noteheads.svaticana .reverse.vplica	
noteheads.svaticana .inner.cephalicus		noteheads.svaticana .cephalicus	
noteheads .svaticana.quilisma			

Glyphes de style medicaea

clefs.medicaea.do		clefs.medicaea.do_change	
clefs.medicaea.fa		clefs.medicaea.fa_change	
custodes.medicaea.u0		custodes.medicaea.u1	
custodes.medicaea.u2		custodes.medicaea.d0	
custodes.medicaea.d1		custodes.medicaea.d2	
accidentals.medicaeaM1		noteheads.smedicaea .inclinatum	
noteheads .smedicaea.punctum		noteheads .smedicaea.rvirga	
noteheads .smedicaea.virga			

Glyphes de style Hufnagel

clefs.hufnagel.do 

clefs.hufnagel.do_change 

clefs.hufnagel.fa 

clefs.hufnagel.fa_change 

clefs.hufnagel.do.fa 

clefs.hufnagel.do.fa_change 

custodes.hufnagel.u0 

custodes.hufnagel.u1 

custodes.hufnagel.u2 

custodes.hufnagel.d0 

custodes.hufnagel.d1 

custodes.hufnagel.d2 

accidentals.hufnagelM1 

noteheads.shufnagel.punctum

noteheads.shufnagel.virga 

noteheads.shufnagel.lpes

Glyphes de style mensural

rests.M3mensural

rests.M2mensural

rests.M1mensural

rests.0mensural

rests.1mensural

rests.2mensural

rests.3mensural

rests.4mensural

clefs.mensural.c 

clefs.mensural.c_change 

clefs.blackmensural.c 

clefs.blackmensural.c_change 

clefs.mensural.f 

clefs.mensural.f_change 

clefs.mensural.g 

clefs.mensural.g_change 

custodes.mensural.u0 

custodes.mensural.u1 

custodes.mensural.u2 

custodes.mensural.d0

custodes.mensural.d1 

custodes.mensural.d2 

accidentals.mensural1

accidentals.mensuralM1

flags.mensuralu03

flags.mensuralu13

flags.mensuralu23

flags.mensurald03 

flags.mensurald13

flags.mensurald23

flags.mensuralu04

flags.mensuralu14

flags.mensuralu24

flags.mensurald04 

flags.mensurald14

flags.mensurald24 

flags.mensuralu05

flags.mensuralu15

flags.mensuralu25

flags.mensurald05 

flags.mensurald15

flags.mensurald25 

flags.mensuralu06

flags.mensuralu16

flags.mensuralu26

flags.mensurald06 

flags.mensurald16

flags.mensurald26 

timesig.mensural44 

timesig.mensural22 

timesig.mensural32 

timesig.mensural64 

timesig.mensural94 

timesig.mensural34 

timesig.mensural68 

timesig.mensural98 

timesig.mensural48 

timesig.mensural68alt 

timesig.mensural24 

noteheads.uM3mensural 

noteheads.dM3mensural 

noteheads.sM3ligmensural 

noteheads.uM2mensural 

noteheads.dM2mensural 

noteheads.sM2ligmensural 

noteheads.sM1mensural 

noteheads.urM3mensural 

noteheads.drM3mensural 

noteheads
.srM3ligmensural 

noteheads.urM2mensural 

noteheads.drM2mensural 

noteheads
.srM2ligmensural 

noteheads.srM1mensural 

noteheads .uM3semimensural 

noteheads .dM3semimensural 

noteheads .sM3semiligmensural 

noteheads .uM2semimensural 

noteheads .dM2semimensural 

noteheads .sM2semiligmensural 

noteheads .sM1semimensural 

noteheads .urM3semimensural 

noteheads .drM3semimensural 

noteheads .srM3semiligmensural 

noteheads .urM2semimensural 

noteheads .drM2semimensural 

noteheads .srM2semiligmensural 

noteheads .srM1semimensural 

noteheads .uM3blackmensural 

noteheads .dM3blackmensural 

noteheads .sM3blackligmensural 

noteheads .uM2blackmensural 

noteheads .dM2blackmensural 

noteheads .sM2blackligmensural 

noteheads .sM1blackmensural 

noteheads.s0mensural

noteheads.s1mensural

noteheads.s2mensural

noteheads .s0blackmensural

Glyphes de style néomensural

rests.M3neomensural		rests.M2neomensural	
rests.M1neomensural		rests.0neomensural	
rests.1neomensural		rests.2neomensural	
rests.3neomensural		rests.4neomensural	
clefs.neomensural.c		clefs.neomensural .c_change	
timesig.neomensural44		timesig.neomensural22	
timesig.neomensural32		timesig.neomensural64	
timesig.neomensural94		timesig.neomensural34	
timesig.neomensural68		timesig.neomensural98	
timesig.neomensural48		timesig.neomensural68alt	
timesig.neomensural24		noteheads.uM3neomensural	
noteheads.dM3neomensural		noteheads.uM2neomensural	
noteheads.dM2neomensural		noteheads.sM1neomensural	
noteheads .urM3neomensural		noteheads .drM3neomensural	

noteheads .urM2neomensural 

noteheads .drM2neomensural 

noteheads .srM1neomensural 

noteheads.s0neomensural

noteheads.s1neomensural 

noteheads.s2neomensural

Glyphes de style Petrucci

clefs.petrucci.c1 

clefs.petrucci.c1_change 

clefs.petrucci.c2 

clefs.petrucci.c2_change 

clefs.petrucci.c3 

clefs.petrucci.c3_change 

clefs.petrucci.c4 

clefs.petrucci.c4_change 

clefs.petrucci.c5 

clefs.petrucci.c5_change 

clefs.petrucci.f 

clefs.petrucci.f_change 

clefs.petrucci.g 

clefs.petrucci.g_change 

noteheads.s0petrucci 

noteheads.s1petrucci 

noteheads.s2petrucci 

noteheads .s0blackpetrucci 

noteheads .s1blackpetrucci 

noteheads .s2blackpetrucci 

Glyphes de style Solesmes

noteheads.ssolesmes.incl.parvum

noteheads.ssolesmes.auct.asc

noteheads.ssolesmes.auct.desc

noteheads.ssolesmes.incl.auctum

noteheads.ssolesmes.stropha

noteheads.ssolesmes.stropha.aucta

noteheads.ssolesmes.oriscus

Glyphes de style kiévien

clefs.kievan.do 

clefs.kievan.do_change 

accidentals.kievan1 

accidentals.kievanM1 

scripts.barline.kievan 

dots.dotkievan

noteheads.sM2kievan 

noteheads.sM1kievan 

noteheads.s0kievan 

noteheads.d2kievan 

noteheads.u2kievan 

noteheads.s1kievan 

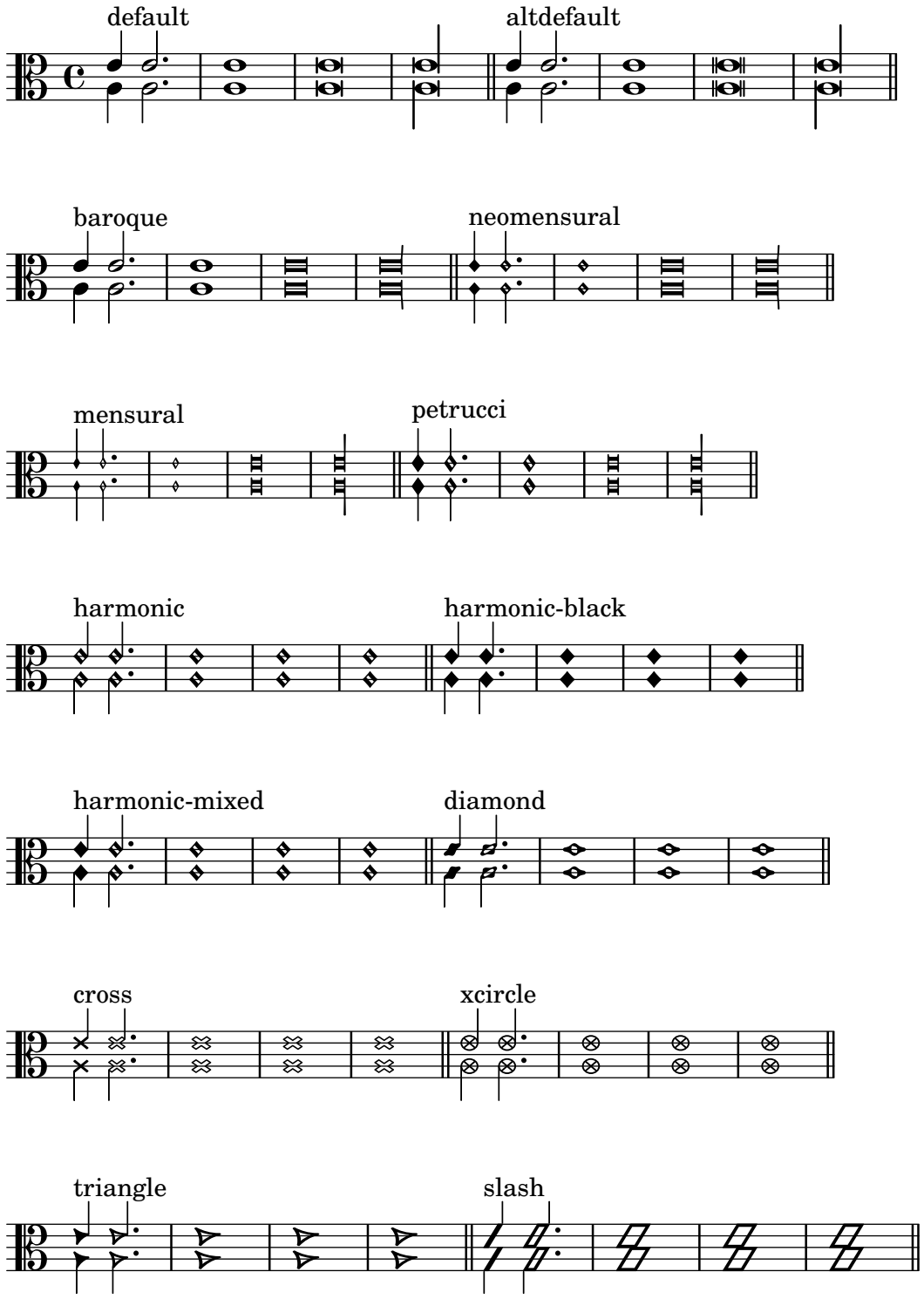
noteheads.sr1kievan 

noteheads.d3kievan 

noteheads.u3kievan 

A.9 Styles de tête de note
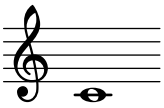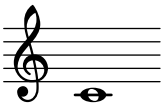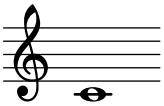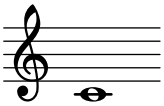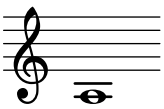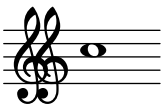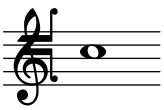
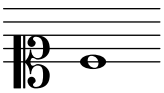
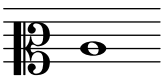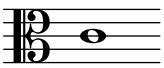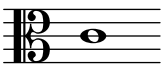
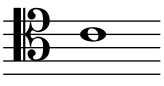
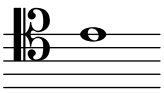
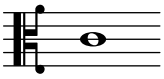
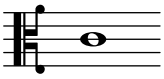
Voici les différents styles de tête de note disponibles.



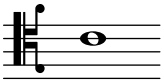
A.10 Styles de clef

Le tableau suivant répertorie tous les styles de clef disponibles ainsi que la position du *do médium* par rapport à la clef.
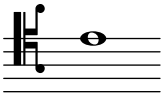
Clefs standards

Exemple	Résultat	Exemple	Résultat
`\clef G`		`\clef "G2"`	
`\clef treble`		`\clef violin`	
`\clef french`		`\clef GG`	
`\clef tenorG`			
`\clef soprano`		`\clef mezzosoprano`	
`\clef C`		`\clef alto`	
`\clef tenor`		`\clef baritone`	
`\clef varC`		`\clef altovarC`	

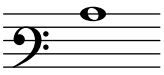
```
\clef tenorvarC
```



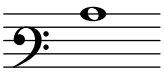
```
\clef baritonevarC
```
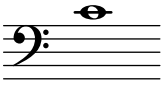


```
\clef varbaritone
```
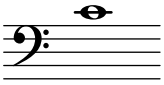


```
\clef baritonevarF
```



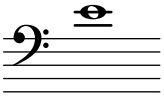
```
\clef F
```



```
\clef bass
```
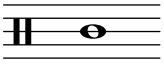


```
\clef subbass
```



Clefs pour portée de percussions
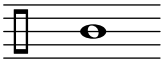
Exemple **Résultat**

```
\clef percussion
```



Exemple **Résultat**
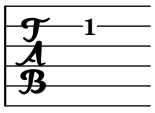
```
\clef varpercussion
```



Clefs pour tablatures

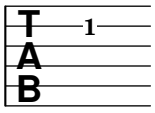
Exemple **Résultat**

```
 \new TabStaff {
   \clef tab
}
```



Exemple **Résultat**

```
 \new TabStaff {
   \clef moderntab
}
```

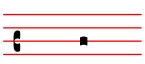


Clefs de musique ancienne

Grégorien

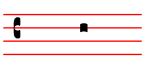
Exemple **Résultat**

```
\clef "vaticana-do1"
```



Exemple **Résultat**

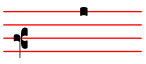
```
\clef "vaticana-do2"
```



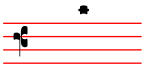
```
\clef "vaticana-do3"
```



```
\clef "vaticana-fa1"
```



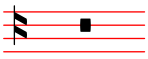
```
\clef "vaticana-fa2"
```



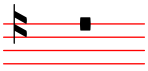
```
\clef "medicaea-do1"
```



```
\clef "medicaea-do2"
```



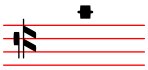
```
\clef "medicaea-do3"
```



```
\clef "medicaea-fa1"
```



```
\clef "medicaea-fa2"
```



```
\clef "hufnagel-do1"
```



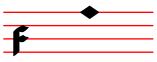
```
\clef "hufnagel-do2"
```



```
\clef "hufnagel-do3"
```



```
\clef "hufnagel-fa1"
```



```
\clef "hufnagel-fa2"
```
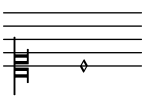


```
\clef "hufnagel-do-fa"
```



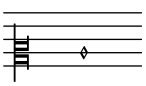
Mensural

Exemple **Résultat** **Exemple** **Résultat**

```
\clef "mensural-c1"
```
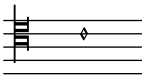


```
\clef "mensural-c2"
```
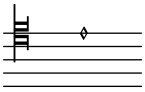


```
\clef "mensural-c3"
```



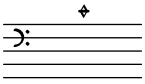
```
\clef "mensural-c4"
```



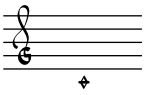
```
\clef "mensural-c5"
```



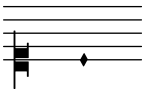
```
\clef "mensural-f"
```



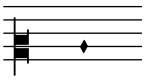
```
\clef "mensural-g"
```



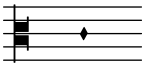
```
\clef "blackmensural-c1"
```



```
\clef "blackmensural-c2"
```



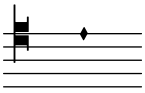
```
\clef "blackmensural-c3"
```
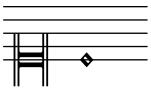


```
\clef "blackmensural-c4"
```
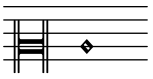


```
\clef "blackmensural-c5"
```



```
\clef "neomensural-c1"
```
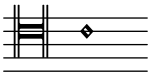


```
\clef "neomensural-c2"
```



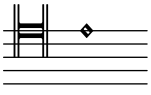
```
\clef "neomensural-c3"
```
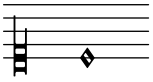


```
\clef "neomensural-c4"
```



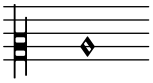
```
\clef "neomensural-c5"
```



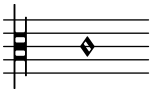
```
\clef "petrucci-c1"
```
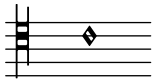


```
\clef "petrucci-c2"
```



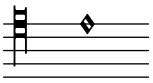
```
\clef "petrucci-c3"
```



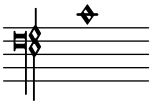
```
\clef "petrucci-c4"
```



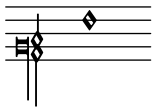
```
\clef "petrucci-c5"
```



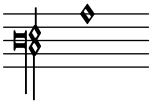
```
\clef "petrucci-f"
```
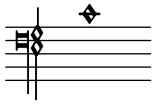


```
\clef "petrucci-f2"
```



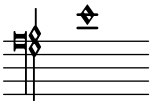
```
\clef "petrucci-f3"
```
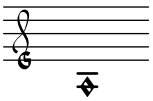


```
\clef "petrucci-f4"
```
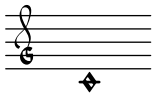


```
\clef "petrucci-f5"
```



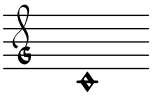
```
\clef "petrucci-g1"
```



```
\clef "petrucci-g2"
```



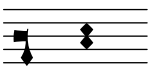
```
\clef "petrucci-g"
```



Kiévien

Exemple **Résultat**

```
\clef "kievan-do"
```



A.11 Commandes pour *markup*

Les commandes suivantes peuvent être utilisées dans un bloc `\markup { }`.

The following commands can all be used inside `\markup { }`.

A.11.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)
: Use *size* as the absolute font size (in points) to display *arg*. Adjusts `baseline-skip` and `word-space` accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```



Used properties:

- `baseline-skip` (3)
- `word-space` (0.6)

`\bold` *arg* (markup)
: Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```



`\box` *arg* (markup)
: Draw a box round *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box
  \line { V. S. }
}
```



Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps` *arg* (markup)
Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic` *arg* (markup)
Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like 'più **f**', the normal words (like 'più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

sfzp

`\finger` *arg* (markup)
Set *arg* as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

1 2 3 4 5

`\fontCaps` *arg* (markup)
Set `font-shape` to `caps`

Note: `\fontCaps` requires the installation and selection of fonts which support the `caps` font shape.

`\fontsize` *increment* (number) *arg* (markup)
Add *increment* to the font-size. Adjusts `baseline-skip` accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

default smaller

Used properties:

- `baseline-skip` (2)

- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

default huge

`\italic` *arg* (markup)
Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

default large

`\larger` *arg* (markup)
Increase the font size relative to the current setting.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

default larger

`\magnify` *sz* (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

```
A \magnify #1.1 { A } A
```

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

`\medium` *arg* (markup)
Switch to medium font-series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

some bold text medium font series **bold again**

`\normal-size-sub` *arg* (markup)
Set *arg* in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default $_{\text{subscript in standard size}}$

Used properties:

- `font-size` (0)

`\normal-size-super` *arg* (markup)
Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

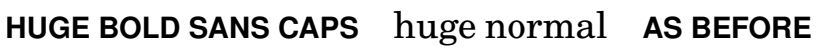
default $^{\text{superscript in standard size}}$

Used properties:

- `font-size` (0)

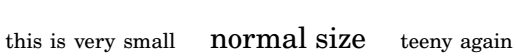
`\normal-text` *arg* (markup)
Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
    \hspace #2
    as before
  }
}
```



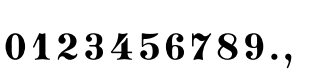
`\normalsize` *arg* (markup)
Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```



`\number` *arg* (markup)
Set font family to `number`, which yields the font used for time signatures and fingerings. This font contains numbers and some punctuation; it has no letters.

```
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
```



`\overtie` *arg* (markup)
Overtie *arg*.

```
\markup \line {
  \overtie "overtied"
  \override #'((offset . 5) (thickness . 1))
```

```
  \overtie "overtied"
  \override #'((offset . 1) (thickness . 5))
  \overtie "overtied"
}
```



Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\replace` *replacements* (list) *arg* (markup)

Used to automatically replace a string by another in the markup *arg*. Each pair of the alist *replacements* specifies what should be replaced. The `key` is the string to be replaced by the `value` string.

```
\markup \replace #'(("thx" . "Thanks!")) thx
```



Used properties:

- `replacement-alist`

`\roman` *arg* (markup)

Set font family to `roman`.

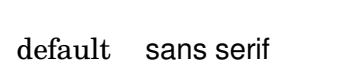
```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}
```



`\sans` *arg* (markup)

Switch to the sans serif font family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```



`\simple` *str* (string)
: A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

  Note: for creating standard text markup or defining new markup commands, the use of `\simple` is unnecessary.

```
\markup {
  \simple #"simple"
  \simple #"text"
  \simple #"strings"
}
```

simple text strings

`\small` *arg* (markup)
: Set font size to -1.

```
\markup {
  default
  \hspace #2
  \small
  small
}
```

default small

`\smallCaps` *arg* (markup)
: Emit *arg* as small caps.

  Note: `\smallCaps` does not support accented characters.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\smaller` *arg* (markup)
: Decrease the font size relative to the current setting.

```
\markup {
  \fontsize #3.5 {
    large text
    \hspace #2
    \smaller { smaller text }
    \hspace #2
    large text
  }
}
```

large text smaller text large text

`\sub` *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

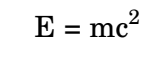


Used properties:

- `font-size` (0)

`\super` *arg* (markup)
Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

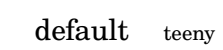


Used properties:

- `font-size` (0)

`\teeny` *arg* (markup)
Set font size to -3.

```
\markup {
  default
  \hspace #2
  \teeny
  teeny
}
```



`\text` *arg* (markup)
Use a text font instead of music symbol or music alphabet font.

```
\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
```

```
    5
  }
}
```

1, 2, three, four, **5**

`\tie` *arg* (markup)
: Adds a horizontal bow created with `make-tie-stencil` at bottom or top of *arg*. Looks at `thickness` to determine line thickness, and `offset` to determine y-offset. The added bow fits the extent of *arg*, `shorten-pair` may be used to modify this. *direction* may be set using an `override` or direction-modifiers or `voiceOne`, etc.

```
\markup {
  \override #'(direction . 1)
  \tie "above"
  \override #'(direction . -1)
  \tie "below"
}
```

above below

Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\tiny` *arg* (markup)
: Set font size to -2.

```
\markup {
  default
  \hspace #2
  \tiny
  tiny
}
```

default tiny

`\typewriter` *arg* (markup)
: Use `font-family` typewriter for *arg*.

```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

default `typewriter`

`\underline` *arg* (markup)
: Underline *arg*. Looks at `thickness` to determine line thickness, `offset` to determine line y-offset from *arg* and `underline-skip` to determine the distance of additional

lines from the others. `underline-shift` is used to get subsequent calls correct. Overriding it makes little sense, it would end up adding the provided value to the one of `offset`.

```
\markup \justify-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(underline-skip . 4)
  \underline \underline \underline "multiple underlined"
}
```

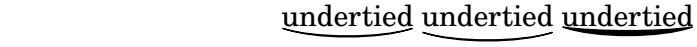


Used properties:

- `underline-skip` (2)
- `underline-shift` (0)
- `offset` (2)
- `thickness` (1)

`\undertie` *arg* (markup)

```
\markup \line {
  \undertie "undertied"
  \override #'((offset . 5) (thickness . 1))
  \undertie "undertied"
  \override #'((offset . 1) (thickness . 5))
  \undertie "undertied"
}
```



Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\upright` *arg* (markup)

Set `font-shape` to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
```

```
    }
    \hspace #2
    italic again
  }
}
```

italic text upright text *italic again*

A.11.2 Align

`\center-align` *arg* (markup)
Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```

one
two
three

`\center-column` *args* (markup list)
Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\column` *args* (markup list)
Stack the markups in *args* vertically. The property `baseline-skip` determines the space between markups in *args*.

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)
: Print two markups on top of each other.

  Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; for this purpose use `\overlay` instead.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```



`\concat` *args* (markup list)
: Concatenate *args* in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

```
\markup {
  \concat {
    one
    two
    three
  }
}
```

onetwothree

`\dir-column` *args* (markup list)
: Make a column of *args*, going up or down, depending on the setting of the `direction` layout property.

```
\markup {
  \override #`(direction . ,UP)
  \dir-column {
    going up
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #'(direction . 1)
  \dir-column {
    going up
  }
}
```

up up
going going going
down

Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *args* (markup list)

Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```
\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
    \null
    \fill-line {
      \line { Text markups }
      \line {
        \italic { evenly spaced }
      }
      \line { across the page }
    }
    \null
    \override #'(line-width . 50)
    \fill-line {
      Width explicitly specified
    }
  }
}
```

Words evenly spaced across the page

Text markups *evenly spaced* across the page

Width explicitly specified

Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* in a horizontal line of width `line-width` with a line of markups *pattern* in between. Patterns are spaced apart by *space*. Patterns are aligned to the *dir* markup.

```
\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
```

```
  "center-aligned :"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned :"
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left first
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left second
}
```

right-aligned :
first . . . . . right
second . . . . . right

center-aligned :
left - right

left-aligned :
left : : : : : : : : : : : : : : : : first
left : : : : : : : : : : : : : : : : second

Used properties:

- `line-width`
- `word-space`

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)
Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
    \line {
      one
      \general-align #Y #UP
      two
      three
    }
    \null
    \line {
      one
      \general-align #Y #3.2
      two
      three
```

```
    }
  }
}
```

one
two
three

one
two
three

one two three

one three
two

\halign *dir* (number) *arg* (markup)
: Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values in between interpolate alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
    two
    three
  }
}
```

one
two
three

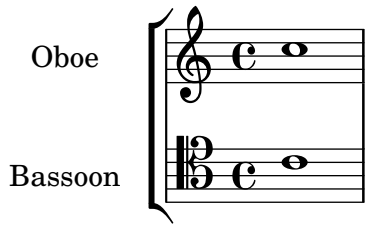
one
two
three

one
two
three

one
two
three

`\hcenter-in` *length* (number) *arg* (markup)
Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
  }
>>
```



`\hspace` *amount* (number)
Create an invisible object taking up horizontal space *amount*.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

`\justify-field` *symbol* (symbol)
Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify-line` *args* (markup list)
Put *markups* in a horizontal line of width *line-width*. The markups are spread to fill the entire line and separated by equal space. If there are no arguments, return an empty stencil.

```
\markup {
  \justify-line {
    Constant space between neighboring words
  }
}
```

Constant space between neighboring words

Used properties:

- `line-width` (#f)
- `word-space` (0.6)

- `text-direction` (1)

`\justify` *args* (markup list)
: Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\justify-string` *arg* (string)
: Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut
      labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident, sunt
      in culpa qui officia deserunt mollit anim id est
      laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align` *arg* (markup)
Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

one
two
three

`\left-column` *args* (markup list)
Put `args` in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (markup list)
Put *args* in a horizontal line. The property `word-space` determines the space between markups in *args*.

```
\markup {
```

```
  \line {
    one two three
  }
}
```

one two three

Used properties:

- `text-direction` (1)
- `word-space`

`\lower` *amount* (number) *arg* (markup)
: Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```

one three
two

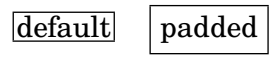
`\overlay` *args* (markup list)
: Takes a list of markups combining them.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \overlay {
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
    \translate #'(0 . 4)\arrow-head #Y #UP ##f
  }
}
```



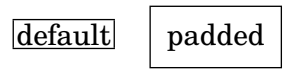
`\pad-around` *amount* (number) *arg* (markup)
: Add padding *amount* all around *arg*.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```



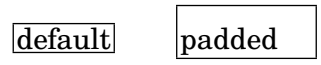
`\pad-markup` *amount* (number) *arg* (markup)
Add space around a markup object. Identical to `pad-around`.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-markup #1 {
      padded
    }
  }
}
```



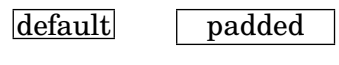
`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



`\pad-x` *amount* (number) *arg* (markup)
Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```



`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)

Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

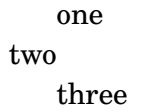
If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```


`\right-align` *arg* (markup)

Align *arg* on its right edge.

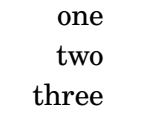
```
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
```



`\right-column` *args* (markup list)

Put `args` in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

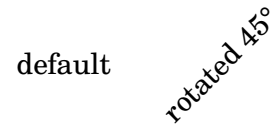


Used properties:

- `baseline-skip`

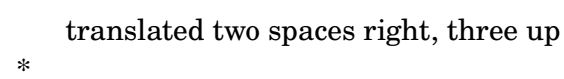
`\rotate` *ang* (number) *arg* (markup)
: Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45˚
  }
}
```



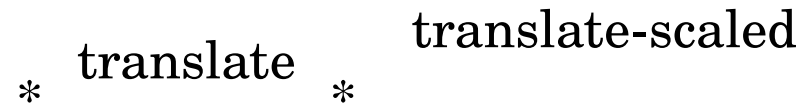
`\translate` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* relative to its surroundings. *offset* is a pair of numbers representing the displacement in the X and Y axis.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```



`\translate-scaled` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```



Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)
: Align `arg` to its Y center.

```
\markup {
  one
  \vcenter
  two
  three
}
```

one $_{\text{two}}$ three

`\vspace` *amount* (number)
: Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
    \center-column {
    one
    \vspace #2
    two
    \vspace #5
    three
  }
}
```

one

two

three

`\wordwrap-field` *symbol* (symbol)
: Wordwrap the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut
    labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris nisi
    ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\wordwrap` *args* (markup list)

Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where X is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet,
      consectetur adipisicing elit, sed do eiusmod tempor
      incididunt ut labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident,
      sunt in culpa qui officia deserunt mollit anim id
      est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et
dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

A.11.3 Graphic

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
```


`\beam` *width* (number) *slope* (number) *thickness* (number)

Create a beam with the specified parameters.

```
\markup {
  \beam #5 #1 #2
}
```



`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note {2.} #UP
  }
```

```
}
```



`\circle` *arg* (markup)
: Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```



Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle` *radius* (number) *thickness* (number) *filled* (boolean)
: A circle of radius *radius* and thickness *thickness*, optionally filled.

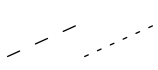
```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```



`\draw-dashed-line` *dest* (pair of numbers)
: A dashed line.

  If `full-length` is set to #t (default) the dashed-line extends to the whole length given by *dest*, without white space at beginning or end. `off` will then be altered to fit. To insist on the given (or default) values of `on`, `off` use `\override #'(full-length . #f)` Manual settings for `on`,`off` and `phase` are possible.

```
\markup {
  \draw-dashed-line #'(5.1 . 2.3)
  \override #'((on . 0.3) (off . 0.5))
  \draw-dashed-line #'(5.1 . 2.3)
}
```



Used properties:

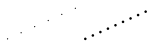
- `full-length` (#t)
- `phase` (0)
- `off` (1)
- `on` (1)

- `thickness` (1)

`\draw-dotted-line` *dest* (pair of numbers)
A dotted line.

The dotted-line always extends to the whole length given by *dest*, without white space at beginning or end. Manual settings for `off` are possible to get larger or smaller space between the dots. The given (or default) value of `off` will be altered to fit the line-length.

```
\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'((thickness . 2) (off . 0.2))
  \draw-dotted-line #'(5.1 . 2.3)
}
```

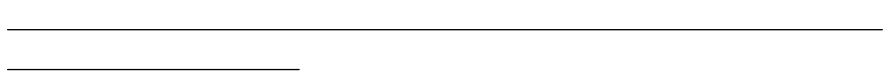


Used properties:

- `phase` (0)
- `off` (1)
- `thickness` (1)

`\draw-hline`
Draws a line across a page, where the property `span-factor` controls what fraction of the page is taken up.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```

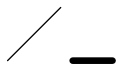


Used properties:

- `span-factor` (1)
- `line-width`
- `draw-line-markup`

`\draw-line` *dest* (pair of numbers)
A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



Used properties:

- `thickness` (1)

`\draw-squiggle-line` *sq-length* (number) *dest* (pair of numbers) *eq-end?* (boolean)
A squiggled line.

If `eq-end?` is set to `#t`, it is ensured the squiggled line ends with a bow in same direction as the starting one. `sq-length` is the length of the first bow. `dest` is the end point of the squiggled line. To match `dest` the squiggled line is scaled accordingly. Its appearance may be customized by overrides for `thickness`, `angularity`, `height` and `orientation`.

```
\markup
  \column {
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(orientation . -1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \draw-squiggle-line #0.5 #'(6 . 0) ##f
    \override #'(height . 1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(thickness . 5)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(angularity . 2)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
  }
```



Used properties:

- `orientation` (1)
- `height` (0.5)
- `angularity` (0)
- `thickness` (0.5)

`\ellipse` *arg* (markup)
Draw an ellipse around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \ellipse {
    Hi
  }
}
```



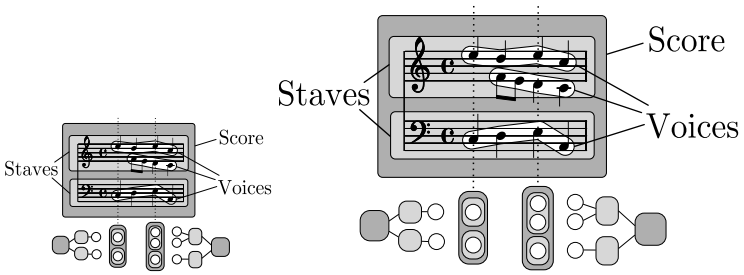
Used properties:

- `y-padding` (0.2)
- `x-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an EPS image. The image is scaled along *axis* to *size*.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```



`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)

Draw a box with rounded corners of dimensions *xext* and *yext*. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

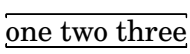
```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```



`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```



`\oval` *arg* (markup)

Draw an oval around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
```

```
  \oval {
    Hi
  }
}
```

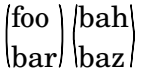

Used properties:

- `y-padding` (0.75)
- `x-padding` (0.75)
- `font-size` (0)
- `thickness` (1)

`\parenthesize` *arg* (markup)

Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \parenthesize
  \column {
    foo
    bar
  }
  \override #'(angularity . 2)
  \parenthesize
  \column {
    bah
    baz
  }
}
```



Used properties:

- `width` (0.25)
- `line-thickness` (0.1)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line *thickness* according to the directions given in *commands*. *commands* is a list of lists where the `car` of each sublist is a drawing command and the `cdr` comprises the associated arguments for each command.

There are seven commands available to use in the list `commands`: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin with *r* are the relative variants of the other three commands.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments; they are the X and Y coordinates for the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands *must* begin with a `moveto` or `rmoveto` to work with the SVG output.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are `'butt`, `'round`, and `'square`. Available line-join styles are `'miter`, `'round`, and `'bevel`.

The property `filled` specifies whether or not the path is filled with color.

```
samplePath =
  #'((moveto 0 0)
     (lineto -1 1)
     (lineto 1 1)
     (lineto 1 -1)
     (curveto -5 -5 -5 5 -1 0)
     (closepath))

\markup {
  \path #0.25 #samplePath

  \override #'(line-join-style . miter)
  \path #0.25 #samplePath

  \override #'(filled . #t)
  \path #0.25 #samplePath
}
```



Used properties:

- `filled` (#f)
- `line-join-style` (round)
- `line-cap-style` (round)

`\postscript` *str* (string)
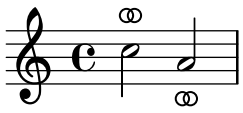
This inserts *str* directly into the output as a PostScript command string.

```
ringsps = #"
  0.15 setlinewidth
  0.9 0.6 moveto
  0.4 0.6 0.5 0 361 arc
  stroke
  1.0 0.6 0.5 0 361 arc
  stroke
  "

rings = \markup {
```

```
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}
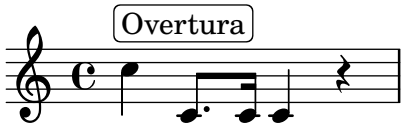
\relative c'' {
  c2^\rings
  a2_\rings
}
```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup; the `corner-radius` property makes it possible to define another shape for the corners (default is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```



Used properties:

- `box-padding` (0.5)
- `font-size` (0)
- `corner-radius` (1)
- `thickness` (1)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*. *factor-pair* is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

```
\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
    mirrored
  }
}
```

stretched mirrored

`\triangle` *filled* (boolean)

A triangle, either filled or empty.

```
\markup {
```

```
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```

▲ △

Used properties:

- `font-size` (0)
- `thickness` (0.1)

`\with-url` *url* (string) *arg* (markup)
Add a link to URL *url* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-url #"https://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

A.11.4 Music

`\compound-meter` *time-sig* (number or pair)
Draw a numeric time signature.

```
\markup {
  \column {
    \line { Single number:
            \compound-meter #3 }
    \line { Conventional:
            \compound-meter #'(4 . 4) or
            \compound-meter #'(4 4) }
    \line { Compound:
            \compound-meter #'(2 3 8) }
    \line { Single-number compound:
            \compound-meter #'((2) (3)) }
    \line { Complex compound:
            \compound-meter #'((2 3 8) (3 4)) }
  }
}
```

Single number: 3
Conventional: 4/4 or 4/4
Compound: 2+3/8
Single-number compound: 2+3
Complex compound: 2+3/8+3/4

`\customTabClef` *num-strings* (integer) *staff-space* (number)
Draw a tab clef sans-serif style.

`\doubleflat`
: Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```



`\doublesharp`
: Draw a double sharp symbol.

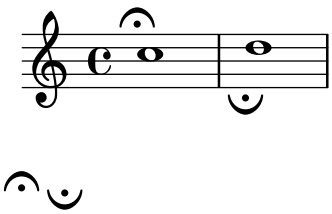
```
\markup {
  \doublesharp
}
```



`\fermata` Create a fermata glyph. When *direction* is `DOWN`, use an inverted glyph. Note that within music, one would usually use the `\fermata` articulation instead of a markup.

```
{ c''1^\markup \fermata d''1_\markup \fermata }

\markup { \fermata \override #`(direction . ,DOWN) \fermata }
```



Used properties:

- `direction` (1)

`\flat` Draw a flat symbol.

```
\markup {
  \flat
}
```



`\multi-measure-rest-by-number` *duration-scale* (non-negative integer)
: Returns a multi-measure rest symbol.

  If the number of measures is greater than the number given by `expand-limit` a horizontal line is printed. For every multi-measure rest lasting more than one measure a number is printed on top.

```
\markup {
  Multi-measure rests may look like
  \multi-measure-rest-by-number #12
  or
  \multi-measure-rest-by-number #7
  (church rests)
}
```

Multi-measure rests may look like 12 or 7 (church rests)

Used properties:

- `multi-measure-rest-number` (#t)
- `width` (8)
- `expand-limit` (10)
- `hair-thickness` (2.0)
- `thick-thickness` (6.6)
- `word-space`
- `style` (())
- `font-size` (0)

`\musicglyph` *glyph-name* (string)

glyph-name is converted to a musical symbol; for example, `\musicglyph #"accidentals.natural"` selects the natural sign from the music font. See Section "The Emmentaler font" dans *Manuel de notation* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```



`\natural` Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem and flag. By using fractional values for *dir*, longer or shorter stems can be obtained. Supports all note-head-styles. Ancient note-head-styles will get mensural-style-flags. `flag-style` may be overridden independently. Supported flag-styles are `default`, `old-straight-flag`, `modern-straight-flag`, `flat-flag`, `mensural` and `neomensural`. The latter two flag-styles will both result in mensural-flags. Both are supplied for convenience.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- `style` (())

- `flag-style` (())
- `font-size` (0)

`\note` *duration* (duration) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note {4.} #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross)
  \note {4..} #UP
  \hspace #2
  \note {\breve} #0
}
```



Used properties:

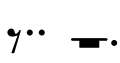
- `style` (())
- `flag-style` (())
- `font-size` (0)

`\rest-by-number` *log* (integer) *dot-count* (integer)

A rest symbol.

For duration logs specified with property `ledgers`, rest symbols with ledger lines are selected.

```
\markup {
  \rest-by-number #3 #2
  \hspace #2
  \rest-by-number #0 #1
}
```



Used properties:

- `style` (())
- `ledgers` ((-1 0 1))
- `font-size` (0)

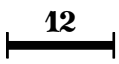
`\rest` *duration* (duration)

Returns a rest symbol.

If `multi-measure-rest` is set to true, a multi-measure rest symbol my be returned. In this case the duration needs to be entered as `{ 1*2 }`to get a multi-measure rest for two bars. Actually, it's only the scaling factor that determines the length, the basic duration is disregarded.

```
\markup {
  Rests:
  \hspace #2
  \rest { 4.. }
  \hspace #2
  \rest { \breve }
  \hspace #2
```

```
  Multi-measure rests:
  \override #'(multi-measure-rest . #t)
  {
  \hspace #2
  \override #'(multi-measure-rest-number . #f)
  \rest { 1*7 }
  \hspace #2
  \rest { 1*12 }
  }
}
```

Rests:  Multi-measure rests: 

Used properties:

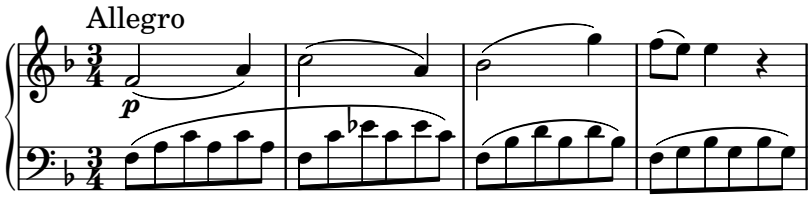
- `multi-measure-rest-number` (#t)
- `width` (8)
- `expand-limit` (10)
- `hair-thickness` (2.0)
- `thick-thickness` (6.6)
- `word-space`
- `style` (())
- `font-size` (0)
- `style` (())
- `ledgers` ((-1 0 1))
- `font-size` (0)

`\score` *score* (score)

Inline an image of music. The reference point (usually the middle staff line) of the lowest staff in the top system is placed on the baseline.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
        f,( bes d bes d bes)
        f( g bes g bes g)
```

```
      }
    >>
    \layout {
      indent = 0.0\cm
      \context {
        \Score
        \override RehearsalMark.break-align-symbols =
           #'(time-signature key-signature)
        \override RehearsalMark.self-alignment-X = #LEFT
      }
      \context {
        \Staff
        \override TimeSignature
                   .break-align-anchor-alignment = #LEFT
      }
    }
  }
}
```



Used properties:

- `baseline-skip`

`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```



`\semisharp`

Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```



`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```



`\sesquisharp`
Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```



`\sharp` Draw a sharp symbol.

```
\markup {
  \sharp
}
```



`\tied-lyric` *str* (string)
Like simple-markup, but use tie characters for '~' tilde symbols.

```
\markup \column {
  \tied-lyric
    #"Siam navi~all'onde~algenti Lasciate~in abbandono"
  \tied-lyric
    #"Impetuosi venti I nostri~affetti sono"
  \tied-lyric
    #"Ogni diletto~e scoglio Tutta la vita~e~un mar."
}
```

Siam navi all'onde algenti Lasciate in abbandono
Impetuosi venti I nostri affetti sono
Ogni diletto e scoglio Tutta la vita e un mar.

Used properties:

- `word-space`

A.11.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)
Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t:`*number* – Set the line thickness (relative to normal line thickness). Default: 0.5.
  - `h:`*number* – Set the height of the diagram in frets. Default: 4.
  - `w:`*number* – Set the width of the diagram in strings. Default: 6.
  - `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.

- `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
- `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
- `c:`*string1*`-`*string2*`-`*fret* – Include a barre mark from *string1* to *string2* on *fret*.
- *string*`-`*fret* – Place a dot on *string* at *fret*. If *fret* is '`o`', *string* is identified as open. If *fret* is '`x`', *string* is identified as muted.
- *string*`-`*fret*`-`*fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.

– Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a `-`, followed by the finger indicator, e.g. '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
  #'((mute 6) (mute 5) (open 4)
     (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute` *`string-number`*`)`
: Place a small ‘x’ at the top of string *string-number*.

`(open` *`string-number`*`)`
: Place a small ‘o’ at the top of string *string-number*.

`(barre` *`start-string end-string fret-number`*`)`
: Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo` *`fret-number`*`)`
: Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret` *`string-number fret-number`* `[`*`finger-value`*`] [`*`color-modifier`*`] [`*`color`*`] ['parenthesized ['default-paren-color]])`
: Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, an optional color modifier *color-modifier*, an optional color *color*, an optional parenthesis `'parenthesized` and an optional paranthesis color `'default-paren-color`. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable *dot-color* or for a single dot by setting the value of *color*. The dot can be parenthesized by adding `'parenthesized`. By default the color for the parenthesis is taken from the dot. Adding `'default-paren-color` will take the parenthesis-color from the global *dot-color*, as a fall-back black will be used. Setting *color-modifier* to `inverted` inverts the dot color for a specific fingering. The values for *string-number*, *fret-number*, and the optional *finger* should be entered first in that order. The order of the other optional arguments does not matter. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\harp-pedal` *definition-string* (string)
: Make a harp pedal diagram.

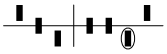
  Possible elements in *definition-string*:

`^`	pedal is up
`-`	pedal is neutral
`v`	pedal is down
`|`	vertical divider line
`o`	the following pedal should be circled (indicating a change)

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (`\override Voice.TextScript.size = #0.3`) for the overall, the thickness property (`\override Voice.TextScript.thickness = #3`) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (`\override Voice.TextScript.harp-pedal-details.box-width = #1`). It contains the following settings: `box-offset` (vertical shift of the box center for up/down pedals), `box-width`, `box-height`, `space-before-divider` (the spacing between two boxes before the divider) and `space-after-divider` (box spacing after the divider).

```
\markup \harp-pedal #"^-v|--ov^"
```



Used properties:

- `thickness` (0.5)
- `harp-pedal-details` (())
- `size` (1.2)

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)
Make a woodwind-instrument diagram. For example, say

```
\markup \woodwind-diagram
  #'oboe #'((lh . (d ees)) (cc . (five3qT1q)) (rh . (gis)))
```

for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:

– piccolo
– flute
– oboe
– clarinet
– bass-clarinet
– saxophone
– bassoon
– contrabassoon

To see all of the callable keys for a given instrument, include the function `(print-keys 'instrument)` in your .ly file, where instrument is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

– 1q (1/4 covered)
– 1h (1/2 covered)

– 3q (3/4 covered)
– R (ring depressed)
– F (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between R and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke `(print-keys-verbose 'instrument)`.

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

```
\markup \woodwind-diagram #'oboe #'()
```

Used properties:

- `graphical` (#t)
- `thickness` (0.1)
- `size` (1)

A.11.6 Accordion Registers

`\discant` *name* (string)

`\discant` ***name*** generates a discant accordion register symbol.

To make it available,

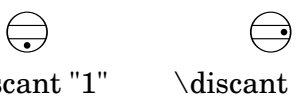
```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The register names in the default `\discant` register set have modeled after numeric Swiss notation like depicted in `http://de.wikipedia.org/wiki/Register_%28Akkordeon%29`, omitting the slashes and dropping leading zeros.

The string *name* is basically a three-digit number with the lowest digit specifying the number of 16' reeds, the tens the number of 8' reeds, and the hundreds specifying the number of 4' reeds. Without modification, the specified number of reeds in 8' is centered in the symbol. Newer instruments may have registrations where 8' can be used either within or without a tone chamber, 'cassotto'. Notationally, the central dot then indicates use of cassotto. One can suffix the tens' digits '`1`' and '`2`' with '`+`' or '`-`' to indicate clustering the dots at the right or left respectively rather than centered.

Some examples are



\discant "1"

\discant "1+0"



\discant "120"



\discant "131"

Used properties:

- `font-size` (0)

`\freeBass` *name* (string)

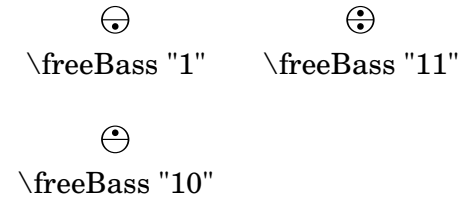
`\freeBass` ***name*** generates a free bass/converter accordion register symbol for the usual two-reed layout.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

Available registrations are



Used properties:

- `font-size` (0)

`\stdBass` *name* (string)

`\stdBass` *`name`* generates a standard bass accordion register symbol.

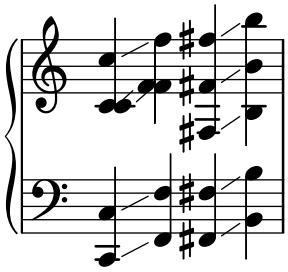
To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The default bass register definitions have been modeled after the article `http://www.accordions.com/index/art/stradella.shtml` originally appearing in Accord Magazine.

The underlying register model is



This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five reed definitions makes more sense than using a four reed layout: in that manner, the '`Master`' register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are



\stdBass "Soprano"



\stdBass "Soft Bass"



\stdBass "Alto"



\stdBass "Soft Tenor"



\stdBass "Tenor"



\stdBass "Bass/Alto"



\stdBass "Master"

Used properties:

- `font-size` (0)

`\stdBassIV` *name* (string)
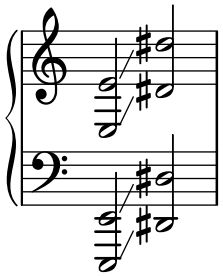
`\stdBassIV` ***name*** generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

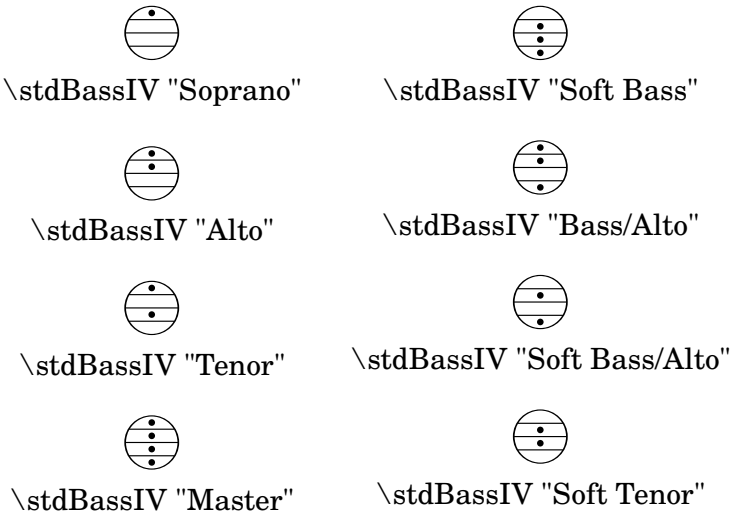
The main use is for four-reed standard bass instruments with reedbank layout



Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches '`Tenor`', '`Master`', '`Soft Bass`'. Note that the sound is quite darker than the same registrations of '`c,`'-based instruments.

Available registrations are



Used properties:

- `font-size` (0)

`\stdBassV` *name* (string)
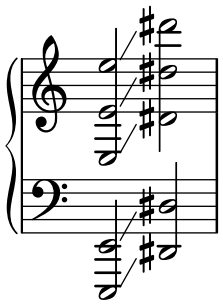
`\stdBassV` *`name`* generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout



This tends to be the bass layout for Hohner's Morino series without convertor or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. The Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional '`Solo Bass`' setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are



\stdBassV "Bass/Alto"



\stdBassV "Soft Bass"



\stdBassV "Soft Bass/Alto"



\stdBassV "Soft Tenor"



\stdBassV "Alto"



\stdBassV "Soprano"



\stdBassV "Tenor"



\stdBassV "Sopranos"



\stdBassV "Master"



\stdBassV "Solo Bass"

Used properties:

- `font-size` (0)

`\stdBassVI` *name* (string)
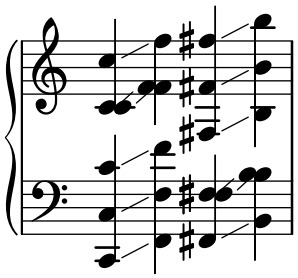
`\stdBassVI` ***name*** generates a standard bass accordion register symbol for six reed basses.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

This is primarily the register layout for the Hohner « Gola » model. The layout is



The registers are effectively quite similar to that of `\stdBass`. An additional bass reed at alto pitch is omitted for esthetical reasons from the '`Master`' setting, so the symbols are almost the same except for the '`Alto/Soprano`' register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are



\stdBassVI "Soprano"



\stdBassVI "Alto/Soprano"



\stdBassVI "Alto"



\stdBassVI "Bass/Alto"



\stdBassVI "Soft Tenor"



\stdBassVI "Soft Bass"



\stdBassVI "Master"

Used properties:

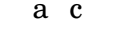
- `font-size` (0)

A.11.7 Other

`\auto-footnote` *mkup* (markup) *note* (markup)

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```



The footnote will be annotated automatically.

Used properties:

- `padding` (0.0)
- `raise` (0.5)

`\backslashed-digit` *num* (integer)

A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\char` *num* (integer)

Produce a single character. Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
```

```
  \char #65 \char ##x00a9
}
```

A ©

`\eyeglasses`
: Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\first-visible` *args* (markup list)
: Use the first markup in *args* that yields a non-empty stencil and ignore the rest.

```
\markup {
  \first-visible {
    \fromproperty #'header:composer
    \italic Unknown
  }
}
```

Unknown

`\footnote` *mkup* (markup) *note* (markup)
: Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

a [1] c [2]

The footnote will not be annotated automatically.

`\fraction` *arg1* (markup) *arg2* (markup)
: Make a fraction of two markups.

```
\markup {
  π ≈
  \fraction 355 113
}
```

$\pi \approx \frac{355}{113}$

Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
: Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
```

```
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

from *myTitle*

`\left-brace` *size* (number)
: A feta brace in point size *size*.

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```



`\lookup` *glyph-name* (string)
: Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```



`\markalphabet` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

H Z

`\markletter` *num* (integer)
Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

H AA

`\null`
An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (procedure) *arg* (markup)
Apply the *procedure* markup command to *arg*. *procedure* takes the same arguments as `interpret-markup` and returns a stencil.

`\override` *new-prop* (pair) *arg* (markup)
Add the argument *new-prop* to the property list. Properties may be any property supported by Section "font-interface" dans *Référence des propriétés internes*, Section "text-interface" dans *Référence des propriétés internes* and Section "instrument-specific-markup-interface" dans *Référence des propriétés internes*.

new-prop may be either a single alist pair, or non-empty alist of its own.

```
\markup {
  \undertie "undertied"
  \override #'(offset . 15)
  \undertie "offset undertied"
  \override #'((offset . 15)(thickness . 3))
  \undertie "offset thick undertied"
}
```

undertied offset undertied offset thick undertied

`\page-link` *page-number* (number) *arg* (markup)
Add a link to the page *page-number* around *arg*. This only works in the PDF backend.

```
\markup {
  \page-link #2  { \italic { This links to page 2... } }
}
```

This links to page 2...

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)

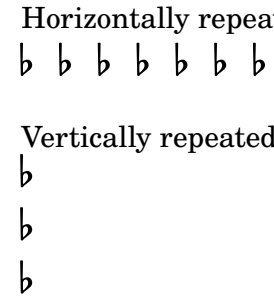
Reference to a page number. *label* is the label set on the referenced page (using `\label` or `\tocItem`), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

(If the current book or bookpart is set to use roman numerals for page numbers, the reference will be formatted accordingly – in which case the *gauge*'s width may require additional tweaking.)

`\pattern` *count* (non-negative integer) *axis* (non-negative integer) *space* (number) *pattern* (markup)

Prints *count* times a *pattern* markup. Patterns are spaced apart by *space* (defined as for `\hspace` or `\vspace`, respectively). Patterns are distributed on *axis*.

```
\markup \column {
  "Horizontally repeated :"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated :"
  \pattern #3 #Y #0.5 \flat
}
```



`\property-recursive` *symbol* (symbol)

Print out a warning when a header field markup contains some recursive markup definition.

`\right-brace` *size* (number)

A feta brace in point size *size*, rotated 180 degrees.

```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```



`\slashed-digit` *num* (integer)

A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
```

```
  \slashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (stencil)
: Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`
: Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)
: Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)
: Read the contents of file *name*, and include it verbatim.

```
\markup {
  \verbatim-file #"en/included/simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative {
  c' d e f g a b c
}
%% Optional helper for automatic updating
%% by convert-ly.  May be omitted.
\version "2.19.21"
```

`\whiteout` *arg* (markup)
: Provide a white background for *arg*. The shape of the white background is determined by `style`. The default is `box` which produces a rectangle. `rounded-box` produces a rounded rectangle. `outline` approximates the outline of the markup.

```
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
```

```
    \override #'(thickness . 1.5)
    \whiteout whiteout-box
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'((style . rounded-box) (thickness . 3))
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'((style . outline) (thickness . 3))
    \whiteout whiteout-outline
}
```







Used properties:

- `thickness` (())
- `style` (box)

`\with-color` *color* (color) *arg* (markup)
Draw *arg* in color specified by *color*.

```
\markup {
  \with-color #red
  red
  \hspace #2
  \with-color #green
  green
  \hspace #2
  \with-color "#0000ff"
  blue
}
```

red green blue

`\with-dimensions-from` *arg1* (markup) *arg2* (markup)
Print *arg2* with the horizontal and vertical dimensions of *arg1*.

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
Set the horizontal and vertical dimensions of *arg* to *x* and *y*.

`\with-link` *label* (symbol) *arg* (markup)
: Add a link to the page holding label *label* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-link #'label {
    \italic { This links to the page
              containing the label... }
  }
}
```

This links to the page containing the label...

`\with-outline` *outline* (markup) *arg* (markup)
: Print *arg* with the outline and dimensions of *outline*. The outline is used by skylines to resolve collisions (not for whiteout).

A.12 Commandes pour liste de *markups*

Les commandes suivantes peuvent être utilisées dans un bloc `\markuplist { }`.

`\column-lines` *args* (markup list)
: Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

  Used properties:

  - `baseline-skip`

`\justified-lines` *args* (markup list)
: Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\map-markup-commands` *compose* (procedure) *args* (markup list)
: This applies the function *compose* to every markup in *args* (including elements of markup list command calls) in order to produce a new markup list. Since the return value from a markup list command call is not a markup list but rather a list of stencils, this requires passing those stencils off as the results of individual markup calls. That way, the results should work out as long as no markups rely on side effects.

`\override-lines` *new-prop* (pair) *args* (markup list)
: Like `\override`, for markup lists.

`\score-lines` *score* (score)
: This is the same as the `\score` markup but delivers its systems as a list of lines. Its *score* argument is entered in braces like it would be for `\score`.

`\string-lines` *strg* (string)
: Takes the string *strg* and splits it at the character provided by the property `split-char`, defaulting to `#\newline`. Surrounding whitespace is removed from

every resulting string. The returned list of markups is ready to be formatted by other markup or markup list commands like `\column`, `\line`, etc.

```
\markup {
  \column
    \string-lines
     "foo, foo,
     bar, bar,
     buzz, buzz!"
}
```

foo, foo,
bar, bar,
buzz, buzz!

Used properties:

- `split-char` (#\newline)

`\table` *column-align* (number list) *lst* (markup list)

Returns a table.

column-align specifies how each column is aligned, possible values are -1, 0, 1. The number of elements in *column-align* determines how many columns will be printed. The entries to print are given by *lst*, a markup-list. If needed, the last row is filled up with `point-stencil`s. Overriding `padding` may be used to increase columns horizontal distance. Overriding `baseline-skip` to increase rows vertical distance.

```
\markuplist {
  \override #'(padding . 2)
  \table
    #'(0 1 0 -1)
    {
      \underline { center-aligned right-aligned
                   center-aligned left-aligned }
      one      \number    1 thousandth \number 0.001
      eleven   \number   11 hundredth  \number 0.01
      twenty   \number   20 tenth      \number 0.1
      thousand \number 1000 one        \number 1.0
    }
}
```

center-aligned	right-aligned	center-aligned	left-aligned
one	1	thousandth	0.001
eleven	11	hundredth	0.01
twenty	20	tenth	0.1
thousand	1000	one	1.0

Used properties:

- `baseline-skip`

- `padding` (0)

`\table-of-contents`
: Used properties:
  - `baseline-skip`

`\wordwrap-internal` *justify* (boolean) *args* (markup list)
: Internal markup list command used to define `\justify` and `\wordwrap`.

  Used properties:
  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)

`\wordwrap-lines` *args* (markup list)
: Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where X is the number of staff spaces.

  Used properties:
  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\wordwrap-string-internal` *justify* (boolean) *arg* (string)
: Internal markup list command that is used to define `\justify-string` and `\wordwrap-string`.

  Used properties:
  - `text-direction` (1)
  - `word-space`
  - `line-width`

A.13 Liste des caractères spéciaux

Voici une table des caractères spéciaux disponibles. Pour plus de précisions, voir [Équivalents ASCII], page 550.

Cette liste utilise la syntaxe HTML, à l'instar de la plupart des caractères qui la composent ; les autres sont inspirés du langage LaTeX.

Les caractères sont ici inclus dans une boîte, de façon à mettre leur taille en évidence, et un léger décalage a été appliqué pour les décoller de l'encadrement.

¡	¡	¿	¿	&solidus;	⁄	&flq;	‹		
&frq;	›	&flqq;	«	&frqq;	»	&glq;	‚		
&grq;	‘	&glqq;	„	&grqq;	“	&elq;	‘		
&erq;	’	&elqq;	“	&erqq;	”				
						&nnbsp;			

‍	‍	‌	‌	·	·	•	•
&copyright;	©	&registered;	®	&trademark;	™	†	†
‡	‡	№	№	ª	ª	º	º
¶	¶	§	§	°	°	№	№
‰	‰	¦	¦	´	´	&acutedbl;	˝
`	̀	˘	̆	ˇ	̌	&cedilla;	̧
&circumflex;	̂	&diaeresis;	̈	&macron;	̄	&aa;	å
&AA;	Å	&ae;	æ	&AE;	Æ	ä	ä
Ä	Ä	&dh;	ð	&DH;	Ð	&dj;	đ
&DJ;	Đ	&l;	ł	&L;	Ł	&ng;	ŋ
&NG;	Ŋ	&o;	ø	&O;	Ø	&oe;	œ
&OE;	Œ	ö	ö	Ö	Ö	&s;	ſ
&ss;	ß	&th;	þ	&TH;	Þ	ü	ü
Ü	Ü	+	+	−	−	×	×
÷	÷	¹	¹	²	²	³	³
&sqrt;	√	&increment;	Δ	&infty;	∞	∑	∑
±	±	&bulletop;	∙	&partial;	∂	&neg;	¬
&currency;	¤	$	$	€	€	&pounds;	£
¥	¥	¢	¢				
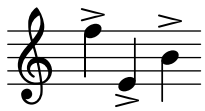
A.14 Liste des signes d'articulation

Dans la logique interne de LilyPond, une « articulation » est un objet, hormis les nuances, qui peut s'attacher directement après un événement rythmique, qu'il s'agisse d'une note ou d'un accord, y compris un silence, un saut ou bien une construction en accord vide `<>` (voir Section "Structuration de la saisie des notes" dans *Manuel d'initiation*). Même les liaisons, doigtés et scripts textuels sont techniquement des articulations, bien qu'ils ne soient pas indiqués ici.

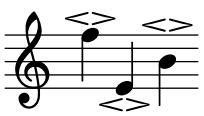
Les listes qui suivent recensent les articulations et autres symboles prédéfinis dans la fonte Emmentaler, que vous pouvez attacher à une note (par ex. `f\accent` ou `f->`). Chaque exemple illustre les trois positionnements : en surplomb (*up*), en dessous (*down*) et à l'appréciation de LilyPond (*neutral*). Consultez aussi [Glyphes de script], page 727, pour un recensement plus exhaustif des glyphes accessibles par la commande de *markup* `\musicglyph` en suivant les préceptes énoncés dans [Notation musicale dans du texte formaté], page 275.

Scripts d'articulation
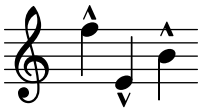
`\accent` ou `->`
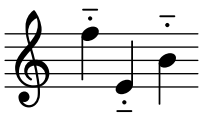


`\espressivo`
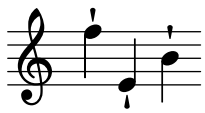


`\marcato` ou `-^`
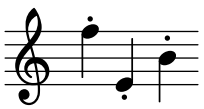


`\portato` ou `-_`
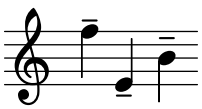


`\staccatissimo` ou `-!`



`\staccato` ou `-.`
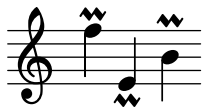


`\tenuto` ou `--`
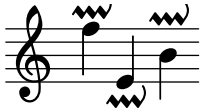


Scripts d'ornement

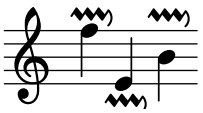
`\prall`
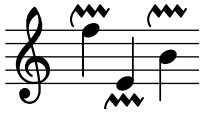


`\prallup`
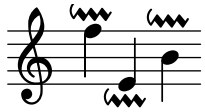


`\pralldown`
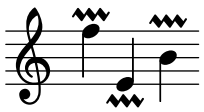


`\upprall`
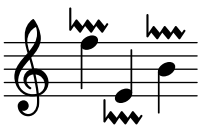


`\downprall`
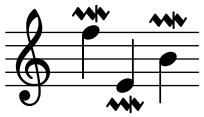


`\prallprall`
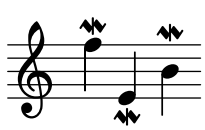


`\lineprall`
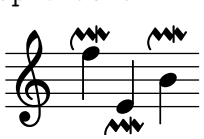


`\prallmordent`
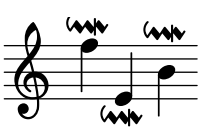


`\mordent`
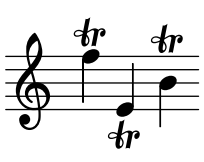


`\upmordent`
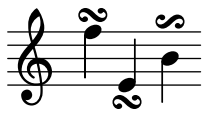


`\downmordent`
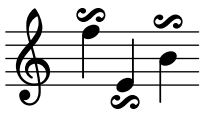


`\trill`
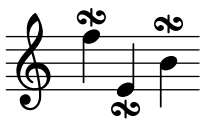


\turn



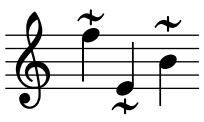
\reverseturn



\slashturn
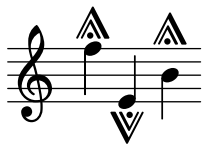


\haydnturn
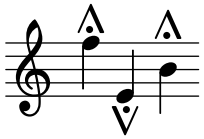


Scripts de point d'orgue et point d'arrêt
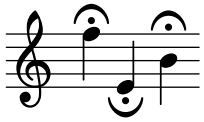
\veryshortfermata



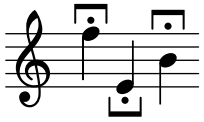
\shortfermata



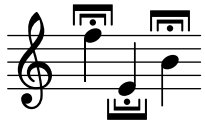
\fermata
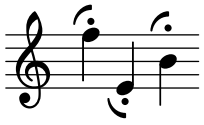


\longfermata
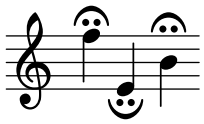


\verylongfermata



\henzeshortfermata
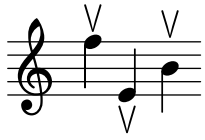


\henzelongfermata
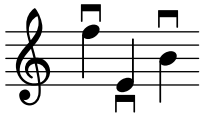


Scripts spécifiques à certains instruments

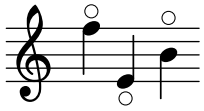
\upbow



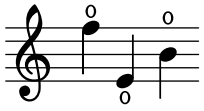
\downbow
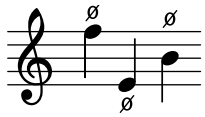


\flageolet



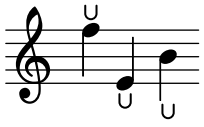
\open



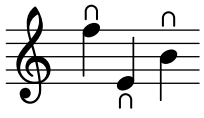
\halfopen



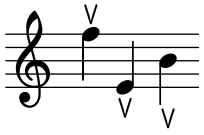
\lheel
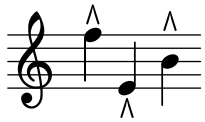


\rheel



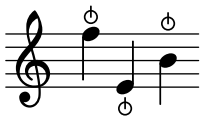
\ltoe



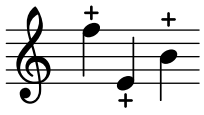
\rtoe



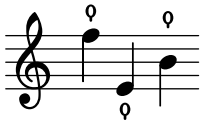
\snappizzicato



\stopped ou -+
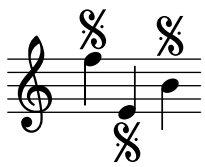


\thumb
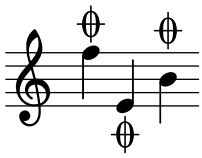


Scripts de reprise et de répétition

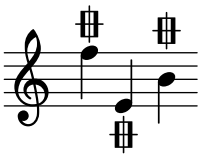
\segno



\coda

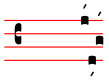


\varcoda

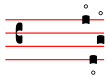


Scripts pour musique ancienne

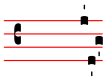
\accentus



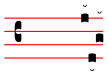
\circulus



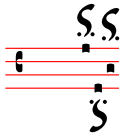
\ictus



\semicirculus
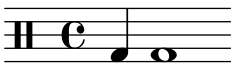


\signumcongruentiae
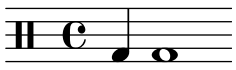


A.15 Notes utilisées en percussion
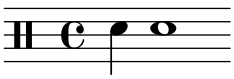
bassdrum
bd
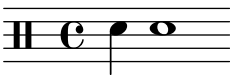


acousticbassdrum
bda
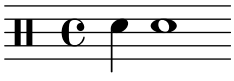


snare
sn
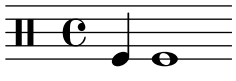


acousticsnare
sna
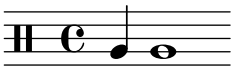


electricsnare
sne
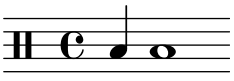


lowfloortom
tomfl
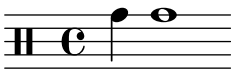


highfloortom
tomfh
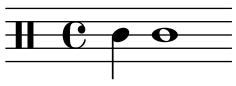


lowtom
toml
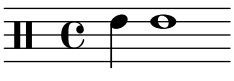


hightom
tomh
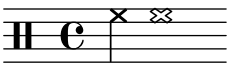


lowmidtom
tomml
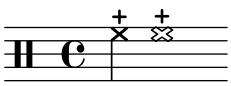


himidtom
tommh
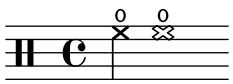


highhat
hh
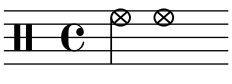


closedhihat
hhc
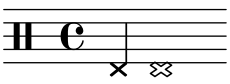


openhighhat
hho
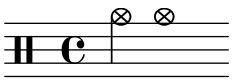


halfopenhihat
hhho
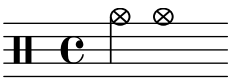


pedalhihat
hhp
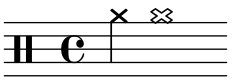


crashcymbal
cymc
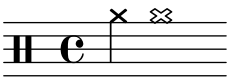


crashcymbala
cymca
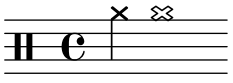


crashcymbalb
cymcb
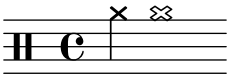


ridecymbal
cymr
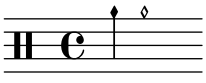


ridecymbala
cymra
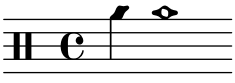


ridecymbalb
cymrb
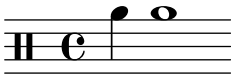


chinesecymbal
cymch
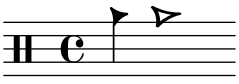


splashcymbal
cyms
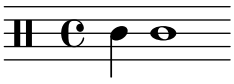


ridebell
rb
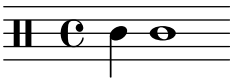


cowbell
cb



hibongo
boh



openhibongo
boho
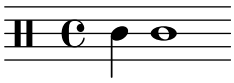


mutehibongo
bohm

lobongo
bol
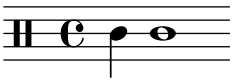
openlobongo
bolo

mutelobongo
bolm
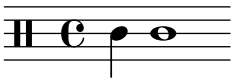
hiconga
cgh
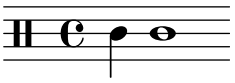


openhiconga
cgho
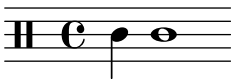


mutehiconga
cghm
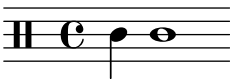


loconga
cgl
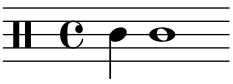


openloconga
cglo
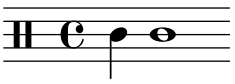


muteloconga
cglm
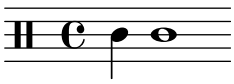


hitimbale
timh
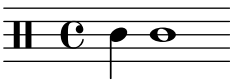


lotimbale
timl
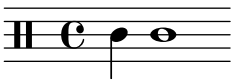


hiagogo
agh
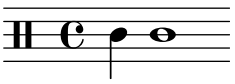


loagogo
agl
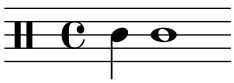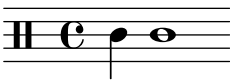


sidestick
ss
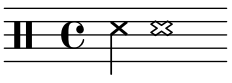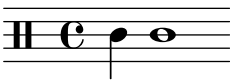


hisidestick
ssh
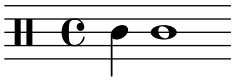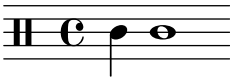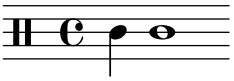










losidestick
ssl
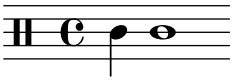


guiro
gui
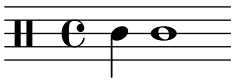


shortguiro
guis
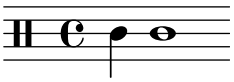


longguiro
guil
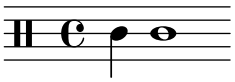


cabasa
cab
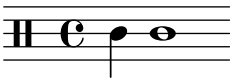


maracas
mar
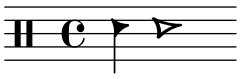


shortwhistle
whs
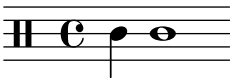


longwhistle
whl
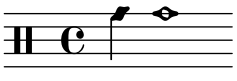


handclap
hc
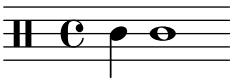


tambourine
tamb
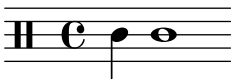


vibraslap
vibs
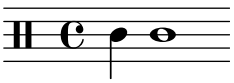


tamtam
tt
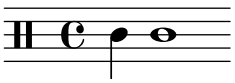


claves
cl
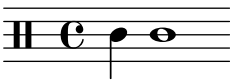


hiwoodblock
wbh
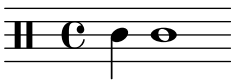


lowoodblock
wbl
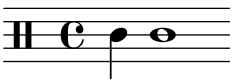


opencuica
cuio
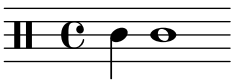


mutecuica
cuim
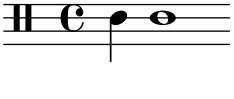


triangle
tri
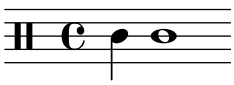


opentriangle
trio
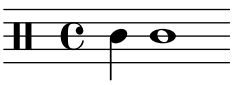


mutetriangle
trim
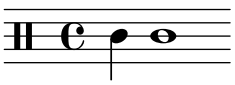


oneup
ua
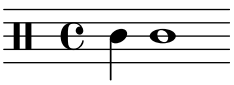


twoup
ub
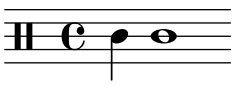


threeup
uc
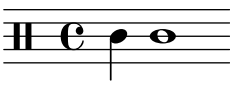


fourup
ud
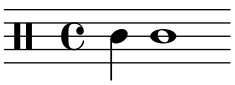


fiveup
ue
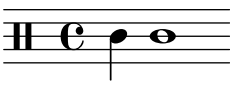


onedown
da
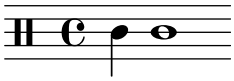


twodown
db
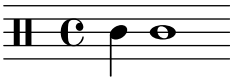


threedown
dc



```
fourdown
dd
```



```
fivedown
de
```



A.16 Glossaire technique

Ce glossaire regroupe les termes techniques et concepts utilisés en interne par LilyPond. Ils apparaissent aussi bien dans les manuels, que sur les listes de diffusion et dans le code source.

alist (liste associative)

Une liste associative – **alist** pour *association list* – est une paire Scheme qui associe une valeur à une clé : `(clé . valeur)`. Le fichier `scm/lily.scm` contient par exemple une *alist* « type-p-name-alist » qui associe certains types de prédicat (par ex. `ly:music?`) à des noms (par ex. « music ») de telle sorte qu'une erreur lors d'un contrôle de typage puisse être rapportée en console avec mention du type de prédicat attendu.

callback (rappel)

Un **rappel**, *callback* en anglais, est une routine, fonction ou méthode qui est passée en argument à une autre fonction. Cette dernière peut alors faire usage de cette fonction de rappel comme de n'importe quelle autre fonction, alors qu'elle ne la connaît pas par avance. Cette façon de procéder permet à des couches logicielles de bas niveau d'appeler des fonctions définies à des niveaux plus élevés. LilyPond utilise abondamment les *callbacks* afin que le code Scheme saisi par l'utilisateur puisse définir les actions de bas niveau à opérer.

closure (clôture)

En Scheme, une **clôture** (en anglais, *closure*) est créée lorsqu'une fonction, généralement une expression lambda, est passée en tant que variable. La clôture comporte, en plus du code de la fonction, des références à des variables libres dans l'environnement lexical – variables utilisées dans l'expression mais définies ailleurs. Lorsque la fonction est par la suite appliquée aux différents arguments, les références aux variables libres, capturées par la clôture, permettent d'obtenir la valeur de ces variables libres qui sera utilisée lors des calculs. L'une des propriétés intéressantes des clôtures est la rétention de la valeur de variables internes tout au long des différentes invocations, leur état étant alors persistant.

glyphe

Un **glyphe** est une représentation graphique particulière d'un caractère typographique ou d'une combinaison de deux caractères formant une ligature. Un jeu de glyphes aux mêmes style et allure constitue une fonte ; un jeu de fontes comprenant plusieurs styles et tailles constitue un type de caractère.

Voir aussi

Manuel de notation : Section 1.8.3 [Fontes], page 279, Section 3.3.3 [Caractères spéciaux], page 549.

grob (objet graphique)

Dans LilyPond, les objets représentants les différents éléments de notation, comme les têtes de note, hampes, liaisons, doigtés, clefs etc. sont appelés « objets de rendu » ou objet graphique –

en anglais *GRaphical OBjects* couramment abrégé en **grob**. Ils sont représentés par des instances de la classe `grob`.

Voir aussi

Manuel d'initiation : Section "Objets et interfaces" dans *Manuel d'initiation*, Section "Conventions de nommage des objets et propriétés" dans *Manuel d'initiation*, Section "Propriétés des objets de rendu" dans *Manuel d'initiation*.

Référence des propriétés internes : Section "grob-interface" dans *Référence des propriétés internes*, Section "All layout objects" dans *Référence des propriétés internes*.

inaltérable

Un objet est dit **inaltérable** – *immutable* en anglais – dès lors que son état ne peut être modifié après sa création ; les objets altérables, à l'inverse, peuvent être modifiés après création.

Pour LilyPond, une propriété est inaltérable ou commune lorsqu'elle définit les style ou le comportement par défaut d'objets graphiques ; une telle propriété est partagée par un certain nombre de *grobs*. En apparente contradiction avec ce que leur nom laisse accroire, de telles propriétés peuvent être adaptées par `\override` et `\revert`.

Voir aussi

Manuel de notation : [altérable], page 811.

interface

Les actions et propriétés communes à plusieurs objets graphiques sont regroupées dans un objet appelé `grob-interface`, ou « interface » pour faire court.

Voir aussi

Manuel d'initiation : Section "Objets et interfaces" dans *Manuel d'initiation*, Section "Conventions de nommage des objets et propriétés" dans *Manuel d'initiation*, Section "Propriétés listées par interface" dans *Manuel d'initiation*.

Manuel de notation : Section 5.2.2 [Interfaces de rendu], page 649.

Référence des propriétés internes : Section "Graphical Object Interfaces" dans *Référence des propriétés internes*.

lexer (analyseur lexical)

Un **lexer** est un programme chargé de convertir une séquence de caractères en une séquence de jetons. Cette opération s'appelle l'analyse lexicale. L'analyseur lexical de LilyPond convertit le flot d'information contenu dans un fichier `.ly` en flot de jetons qui pourront être traités lors de l'étape suivant, l'analyse grammaticale abordée à la rubrique [parser (analyseur syntaxique)], page 812. L'analyseur lexical de LilyPond repose sur `Flex` ; les règles lexicales sont regroupées dans le fichier `lily/lexer.ll`. Ce fichier, partie intégrante des sources, n'est pas distribué avec les programmes binaires de LilyPond.

altérable

Un objet est dit **altérable** – *mutable* en anglais – lorsque son état est sujet à modification après sa création, à l'inverse des objets inaltérables dont l'état est figé dès leur création.

Les propriétés altérables contiennent, pour LilyPond, des valeurs spécifiques à un objet graphique. En particulier, les listes d'autres objets ou résultats de calculs sont enregistrés sous dorme de propriétés altérables.

Voir aussi

Manuel de notation : [inaltérable], page 811.

output-def (définition de sortie)

Une instance de la classe `Output-def` contient les méthodes et structures des données associées à un bloc de sortie. Ces instances sont créées par les blocs `\midi`, `\layout` et `\paper`.

parser (analyseur syntaxique)

Un analyseur syntaxique – **parser** en anglais – est un programme qui analyse la séquence de jetons produite par l'analyseur lexical pour en déterminer la structure grammaticale. Les jetons sont, pour ce faire, regroupés progressivement en tronçons plus importants, selon des règles grammaticales. Lorsque la séquence de jetons est valide, les produit final est une arborescence de jetons ayant à sa base le symbole grammatical de début. Dès lors que cette étape n'est pas concluante, le fichier est déclaré invalide ; un message approprié est alors émis. Les différents regroupements syntaxiques ainsi que les règles de construction des regroupements relatifs à la grammaire de LilyPond sont définis dans le fichier `lily/parser.yy` et présentés selon la forme de Backus-Naur (BNF) à la rubrique Section "Grammaire de LilyPond" dans *Guide du contributeur*. Ce fichier est utilisé par le générateur de *parser* Bison lors de la construction du programme. Partie intégrante des sources, il n'est pas distribué avec les programmes binaires de LilyPond.

variable de l'analyseur grammatical

Il s'agit de variables définies directement en Scheme. Dans la mesure où leur champ sémantique peut porter à confusion, il est fortement déconseillé de les utiliser tels quels.

La modification des valeurs de l'une de ces variables dans un fichier `.ly` sera effective de manière globale. Les valeurs modifiées, sauf à être explicitement remises à leur état d'origine, affecteront tous les blocs `\score` rencontrés, y compris s'ils proviennent d'autres fichiers ajoutés par une commande `\include`. Ceci peut avoir des conséquences inattendues et les erreurs qui pourraient en découler difficiles à localiser dans le cadre d'un projet d'envergure.

LilyPond utilise les variables suivantes :

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

prob (objet de propriété)

Les objets de propriété – **probs** pour *PRoperty OBjects* – sont des instances de la classe `Prob`, une classe de base simple pour les objets qui disposent de de listes associatives de propriétés altérables et inaltérables ainsi que les méthodes pour les manipuler. Les classes `Music` et `Stream_event` dérivent d'un `prob`. Les instances de la classe `prob` se créent aussi pour garder trace du contenu des systèmes une fois formatés et des blocs de titrage lors de la phase de mise en forme des pages.

smob (objet Scheme)

Les objets Scheme – **Smobs** pour *ScheMe OBjects* – font partie du mécanisme utilisé par l'interpréteur Guile pour exporter en code Scheme les objets C ou C++. Dans LilyPond, les *smobs* sont créés, grâce à des macros, à partir d'objets C++. On peut distinguer deux types d'objets *smob* : des *smobs* simples destinés aux objets inaltérables comme les nombres par exemples, et des *smobs* complexes utilisés pour des objets possédant une identité. De plus amples informations sont disponibles dans les sources de LilyPond, au sein du fichier `lily/includes/smob.hh`.

stencil

Une instance de la classe **stencil** comporte l'information nécessaire à l'impression d'un objet typographique. Il s'agit d'un *smob* simple qui contient un espace de confinement qui définit l'envergure verticale et horizontale de l'objet ainsi qu'une expression Scheme qui imprimera l'objet après évaluation. Les stencils peuvent se combiner et adopter une forme plus complexe définie par une arborescence d'expressions Scheme des stencils qui la composent.

La propriété `stencil`, qui permet de connecter un *grob* à son stencil, est définie par l'interface `grob-interface`.

Voir aussi

Référence des propriétés internes : Section “grob-interface” dans *Référence des propriétés internes*.

A.17 Liste des propriétés de contexte

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`additionalBassStrings` (list)
: The additional tablature bass-strings, which will not get a seprate line in TabStaff. It is a list of the pitches of each string (starting with the lowest numbered one).

`additionalPitchPrefix` (string)
: Text with which to prefix additional pitches within a chord name.

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`alternativeNumber` (integer)
: When set, the index of the current `\alternative` element, starting from one. Not set outside of alternatives. Note the distinction from volta number: an alternative may pertain to multiple volte.

`alternativeNumberingStyle` (symbol)
: The scheme and style for numbering bars in repeat alternatives. If not set (the default), bar numbers continue through alternatives. Can be set to `numbers` to reset the bar number at each alternative, or set to `numbers-with-letters` to reset and also include letter suffixes.

`alternativeRestores` (symbol list)
: Timing variables that are restored to their value at the start of the first alternative in subsequent alternatives.

`associatedVoice` (string)
: Name of the context (see `associatedVoiceType` for its type, usually `Voice`) that has the melody for this `Lyrics` line.

`associatedVoiceType` (symbol)
: Type of the context that has the melody for this `Lyrics` line.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  symbol The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Section "Score" dans *Référence des propriétés internes* then all staves share accidentals, and if *context* is Section "Staff" dans *Référence des propriétés internes* then all voices in the same staff share accidentals, but staves do not.

  procedure The procedure represents an accidental rule to be applied to the previously specified context.

  The procedure takes the following arguments:

  `context` The current context to which the rule should be applied.

  `pitch` The pitch of the note to be evaluated.

  `barnum` The current bar number.

  `measurepos`
  : The current measure position.

  The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoBeamCheck` (procedure)
: A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-`#f` return value starts or stops the auto beam.

`autoBeaming` (boolean)
: If set to true then beams are generated automatically.

`autoCautionaries` (list)
: List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`automaticBars` (boolean)
: If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
: If set to true a bar line is drawn after each note.

`barCheckSynchronize` (boolean)
: If true then reset `measurePosition` when finding a bar check.

`barNumberFormatter` (procedure)
: A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
: A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

  The following procedures are predefined:

  `all-bar-numbers-visible`
  : Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

  `first-bar-number-invisible`
  : Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

  `first-bar-number-invisible-save-broken-bars`
  : Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

  `first-bar-number-invisible-and-no-parenthesized-bar-numbers`
  : Enable bar numbers for all bars except the first bar and broken bars. This is the default.

  `(every-nth-bar-number-visible` *n*`)`
  : Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

  `(modulo-bar-number-visible` *n m*`)`
  : If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*chord* `.` *markup*`)` entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNameLowercaseMinor` (boolean)
: Downcase roots of minor chords?

`chordNameSeparator` (markup)
: The markup object used to separate parts of a chord name.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)
: The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`completionFactor` (an exact rational or procedure)
: When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.

  If `#f`, the completion engraver uses the scale-factor of each duration being split.

  If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

`completionUnit` (moment)
: Sub-bar unit of completion.

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Section "Timing_translator" dans *Référence des propriétés internes* at Section "Score" dans *Référence des propriétés internes* level.

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)
: A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`agostini-drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

  The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list `(notehead-style script vertical-position)` as values.

`endAtSkip` (boolean)
: End `DurationLine` grob on `skip-event`

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`figuredBassPlusDirection` (direction)
: Where to put plus signs relative to the main figure.

`fineBarType` (string)
: The bar line for `\fine`. See `whichBar` for information on available bar types.

`fineSegnoType` (string)
: Set the default bar line for a requested segno with fine. Default is '`|.S`'.

`fineStartRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno and fine. Default is '`|.S.|:`'.

`fineText` (markup)
: The text to print at `\fine`.

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)
: If true, create a new clef when starting a staff.

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

`fontSize` (number)
: The relative size of all grobs in a context.

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`handleNegativeFrets` (symbol)
: How the automatic fret calculator should handle calculated negative frets. Values include `'ignore`, to leave them out of the diagram completely, `'include`, to include them as calculated, and `'recalculate`, to ignore the specified string and find a string where they will fit with a positive fret number.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`ignoreBarChecks` (boolean)
: Ignore bar checks.

`ignoreBarNumberChecks` (boolean)
: Ignore bar number checks.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`ignoreMelismata` (boolean)
: Ignore melismata for this Section "Lyrics" dans *Référence des propriétés internes* line.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Section "Lyrics" dans *Référence des propriétés internes*.

`initialTimeSignatureVisibility` (vector)
: break visibility for the initial time signature.

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(`*`min . max`*`)` pair of numbers for the loudness range of the instrument.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`instrumentTransposition` (pitch)
: Define the transposition of the instrument. Its value is the pitch that sounds when the instrument plays written middle C. This is used to transpose the MIDI output, and `\quote`s.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step . alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (number)
: Alignment to use for a melisma syllable.

`magnifyStaffValue` (positive number)
: The most recent value set with `\magnifyStaff`.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`measureStartNow` (boolean)
: True at the beginning of a measure.

`melismaBusyProperties` (list)
: A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example,

if set to `'(melismaBusy beamMelismaBusy)`, only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiBalance` (number)
: Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChannelMapping` (symbol)
: How to map MIDI channels: per `staff` (default), `instrument` or `voice`.

`midiChorusLevel` (number)
: Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiExpression` (number)
: Expression control for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMergeUnisons` (boolean)
: If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`midiPanPosition` (number)
: Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)
: Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`minorChordModifier` (markup)
: Markup displayed following the root for a minor chord

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`noteNameFunction` (procedure)
: Function used to convert pitches into strings and markups.

`noteNameSeparator` (string)
: String used to separate simultaneous NoteName objects.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`nullAccidentals` (boolean)
: The `Accidental_engraver` generates no accidentals for notes in contexts were this is set. In addition to supressing the printed accidental, this option removes any effect the note would have had on accidentals in other voices.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`ottavationMarkups` (list)
: An alist defining the markups used for ottava brackets. It contains entries of the form `(`*`number of octaves . markup`*`)`.

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`partCombineForced` (symbol)
: Override for the partCombine decision. Can be `apart`, `chords`, `unisono`, `solo1`, or `solo2`.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(`*`up updown down`*`)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`printAccidentalNames` (boolean or symbol)
: Print accidentals in the `NoteNames` context.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printNotesLanguage` (string)
: Use a specific language in the `NoteNames` context.

`printOctaveNames` (boolean or symbol)
: Print octave marks in the `NoteNames` context.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where x is a string or `#f`. `'end-repeat` is also accepted as a command.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`restrainOpenStrings` (boolean)
: Exclude open strings from the automatic fret calculator.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

`sectionBarType` (string)
: The bar line for `\section`. See `whichBar` for information on available bar types.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`shapeNoteStyles` (vector)
: Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`skipBars` (boolean)
: If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`slashChordSeparator` (markup)
: The markup object used to separate a chord name from its root note in case of inversions or slash chords.

`soloIIText` (markup)
: The text for the start of a solo for voice ‘two’ when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

`squashedPosition` (integer)
: Vertical position of squashing for Section “Pitch_squash_engraver” dans *Référence des propriétés internes*.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

`stanza` (markup)
: Stanza ‘number’ to print before the start of a verse. Use in `Lyrics` context.

`startAtNoteColumn` (boolean)
: Start `DurationLine` grob at entire `NoteColumn`.

`startAtSkip` (boolean)
: Start `DurationLine` grob at `skip-event`.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is ‘`S.|:`’.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`strictBeatBeaming` (boolean)
: Should partial beams reflect the beat structure even if it causes flags to hang out?

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

`suggestAccidentals` (boolean or symbol)
: If set to `#t`, accidentals are typeset as suggestions above the note. Setting it to `'cautionary` only applies that to cautionary accidentals.

`supportNonIntegerFret` (boolean)
: If set in `Score` the `TabStaff` will print micro-tones as '$2\frac{1}{2}$'

`suspendMelodyDecisions` (boolean)
: When using the `Melody_engraver`, stop changing orientation of stems based on the melody when this is set to true.

`suspendRestMerging` (boolean)
: When using the Merge_rest_engraver do not merge rests when this is set to true.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`timeSignatureSettings` (list)
: A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for `baseMoment`, `beatStructure`, and `beamExceptions`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`tonic` (pitch)
: The tonic of the current scale.

`topLevelAlignment` (boolean)
: If true, the *Vertical_align_engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

`tupletSpannerDuration` (moment)
: Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

```
{
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```

`underlyingRepeatType` (string)
: Set the bar line to use at points of repetition or departure where no bar line would normally appear, for example at the end of a system broken in mid measure where the next system begins with a segno.

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

`vocalName` (markup)
: Name of a vocal line.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

```
\set Staff.whichBar = ".|:"
```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

A.18 Propriétés de mise en forme

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)
: Alteration numbers for accidental.

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)
: Arrow length.

`arrow-width` (number)
: Arrow width.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits and it is larger than this number, make a kneed beam.

`automatically-numbered` (boolean)
: If set, footnotes are automatically numbered.

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`avoid-scripts` (boolean)
: If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`axes` (list) List of axis numbers. In the case of alignment grobs, this should contain only one number.

`bar-extent` (pair of numbers)
The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`base-shortest-duration` (moment)
Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`baseline-skip` (dimension, in staff space)
Distance between base lines of multiple lines of text.

`beam-thickness` (dimension, in staff space)
Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
Width of the tremolo sign.

`beamed-stem-shorten` (list)
How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`before-line-breaking` (boolean)
Dummy property, used to trigger a callback function.

`bend-me` (boolean)
Decide whether this grob is bent.

`between-cols` (pair)
Where to attach a loose column to.

`bound-details` (list)
An alist of properties for determining attachments of spanners to edges.

`bound-padding` (number)
The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)
: Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)
: This is a vector of 3 lists: `#(`*`end-of-line unbroken start-of-line`*`)`. Each list contains *break-align symbols* that specify an order of breakable items (see Section "break-alignment-interface" dans *Référence des propriétés internes*).

  For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
                    ambitus
                    breathing-sign
                    time-signature
                    clef
                    cue-clef
                    staff-bar
                    key-cancellation
                    key-signature
                    custos))
```

`break-align-symbol` (symbol)
: This key is used for aligning, ordering, and spacing breakable items. See Section "break-alignment-interface" dans *Référence des propriétés internes*.

`break-align-symbols` (list)
: A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Section "break-alignment-interface" dans *Référence des propriétés internes*.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`breakable` (boolean)
: Allow breaks here.

`broken-bound-padding` (number)
: The amount of padding to insert when a spanner is broken at a line break.

`chord-dots-limit` (integer)
: Limits the column of dots on each chord to the height of the chord plus `chord-dots-limit` staff-positions.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`clef-alignments` (list)
: An alist of parent-alignments that should be used for clef modifiers with various clefs

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`color` (color)
: The color of this grob.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`count-from` (integer)
: The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`damping` (number)
: Amount of beam slope damping.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`dash-fraction` (number)
: Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`dashed-edge` (boolean)
: If set, the bracket edges are dashed like the rest of the bracket.

`default-direction` (direction)
: Direction determined by note head positions.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`digit-names` (vector)
: Names for string finger digits.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`dot-count` (integer)
: The number of dots.

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)
: List consisting of (*`description string-number fret-number finger-number`*) entries used to define fret diagrams.

`double-stem-separation` (number)
: The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: (*`left-height . right-height`*).

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: (*`left-text . right-text`*).

`endpoint-alignments` (pair of numbers)
: A pair of numbers representing the alignments of an object's endpoints. E.g., the ends of a hairpin relative to `NoteColumn` grobs.

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`extra-dy` (number)
: Slope glissandi this much extra.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flag-count` (number)
: The number of tremolo beams.

`flag-style` (symbol)
: The style of the flag to be used with `MetronomeMark`. Available are `'modern-straight-flag`, `'old-straight-flag`, `flat-flag`, `mensural` and `'default`

`flat-positions` (list)
: Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-features` (list)
: Opentype features.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`footnote` (boolean)
: Should this be a footnote or in-note?

`footnote-music` (music)
: Music creating a footnote.

`footnote-text` (markup)
: A footnote for the grob.

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Section “note-collision-interface” dans *Référence des propriétés internes*.

`forced-spacing` (number)
: Spacing forced between grobs, used in various ligature engravers.

`fraction` (fraction, as pair)
: Numerator and denominator of a time signature object.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a `(`*`property`* `.` *`value`*`)` pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.
  - `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
  - `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
  - `handedness` – Print the fret-diagram left- or right-handed. `-1`, `LEFT` for left ; `1`, `RIGHT` for right. Default `RIGHT`.
  - `paren-padding` – The padding for the parenthesis. Default 0.05.

- `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
- `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
- `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string k is given by `thickness` * (1+`string-thickness-factor`) ^ (k-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`full-size-change` (boolean)
: Don't make a change clef smaller.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`grow-direction` (direction)
: Crescendo or decrescendo?

`hair-thickness` (number)
: Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a (*`property . value`*) pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.
  - `box-height` – Height of each pedal box. Default value 1.0.
  - `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
  - `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
  - `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
  - `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
  - `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizon-padding` (number)
: The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Section “note-collision-interface” dans *Référence des propriétés internes*.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`id` (string)
: An id string for the grob.

`ignore-ambitus` (boolean)
: If set, don’t consider this notehead for ambitus calculation.

`ignore-collision` (boolean)
: If set, don’t do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur position to a (quantized) position that is as close as possible to this value, and print the demerits for the inspected position in the output.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`knee` (boolean)
: Is this beam kneed?

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`knee-to-beam` (boolean)
: Determines whether a tuplet number will be positioned next to a kneed beam.

`labels` (list)
: List of labels (symbols) placed on a column.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-extra` (dimension, in staff space)
: Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)
: Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`ledger-positions-function` (any type)
: A quoted Scheme procedure that takes a `StaffSymbol` grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems (each unit represents half a `staff-space`).

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual « pen » that draws the two arcs of the curve's outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`long-text` (markup)
: Text markup. See Section “Formatting text” dans *Manuel de notation*.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`max-symbol-separation` (number)
: The maximum distance between symbols making up a church rest.

`maximum-gap` (number)
: Maximum value allowed for `gap` property.

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Section "note-collision-interface" dans *Référence des propriétés internes*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-after-break` (dimension, in staff space)
: If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance to the notehead.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

`non-default` (boolean)
: Set for manually specified clefs and keys.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`note-collision-threshold` (dimension, in staff space)
: Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`number-type` (symbol)
: Numbering style. Choices include `roman-lower`, `roman-upper` and `arabic`.

`output-attributes` (list)
: An alist of attributes for the grob, to be included in output files. When the SVG typesetting backend is used, the attributes are assigned to a group (`<g>`) containing all of the stencils that comprise a given grob. For example,

```
'((id . 123) (class . foo) (data-whatever . "bar"))
```

produces

```
<g id="123" class="foo" data-whatever="bar"> ... </g>
```

In the Postscript backend, where there is no way to group items, the setting of the `output-attributes` property has no effect.

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-placement-directive` (symbol)
: One of four directives telling how outside staff objects should be placed.
  - `left-to-right-greedy` – Place each successive grob from left to right.
  - `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
  - `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
  - `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping `(`*`name`*` . `*`name`*`)` to distances.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-number` (number)
: Page number on which this system ends up.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`parent-alignment-X` (number)
: Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`parent-alignment-Y` (number)
: Like `parent-alignment-X` but for the Y axis.

`parenthesis-friends` (list)
: A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates (*start . end*), where *start* and *end* are vertical positions in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`protrusion` (number)
: In an arpeggio bracket, the length of the horizontal edges.

`rank-on-page` (number)
: 0-based index of the system on a page.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`remove-layer` (index or symbol)
: When set as a positive integer, the `Keep_alive_together_engraver` removes all `VerticalAxisGroup` grobs with a `remove-layer` larger than the smallest retained `remove-layer`. Set to `#f` to make a layer independent of the `Keep_alive_together_engraver`. Set to `'()`, the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: `#'any` to keep the layer alive with any other layer in the group; `#'above` or `#'below` to keep the layer alive with the context immediately before or after it, respectively.

`replacement-alist` (list)
: Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

`restore-first` (boolean)
: Print a natural before the accidental.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `'(45 0 0)` rotates by 45 degrees around the center of this object.

`round-up-exceptions` (list)
: A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)
: Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a 3/2 measure.

`rounded` (boolean)
: Decide whether lines should be drawn rounded or not.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`segno-kern` (number)
: The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`shape` (symbol)
: This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

`sharp-positions` (list)
: Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`shortest-duration-space` (number)
: Start with this multiple of `spacing-increment` space for the shortest duration. See also Section “spacing-spanner-interface” dans *Référence des propriétés internes*.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`side-axis` (number)
: If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`simple-Y` (boolean)
: Should the Y placement of a spanner disregard changes in system heights?

`size` (number)
: The ratio of the size of the object to its default size.

`skip-quanting` (boolean)
: Should beam quanting be skipped?

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`snap-radius` (number)
: The maximum distance between two objects that will cause them to snap to alignment along an axis.

`space-alist` (list)
: An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *`break-align-symbol`* are listed in Section “break-alignment-interface” dans *Référence des propriétés internes*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`spacing-increment` (dimension, in staff space)
: The unit of length for note-spacing. Typically, the width of a note head. See also Section “spacing-spanner-interface” dans *Référence des propriétés internes*.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object’s spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest.spacing-pair =
                #'(staff-bar . staff-bar)
```

`spanner-id` (index or symbol)
: An identifier to distinguish concurrent spanners.

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stacking-dir` (direction)
: Stack objects in which direction?

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
  - `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
  - `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`stem-begin-position` (number)
: User override for the begin position of a stem.

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

`stemlet-length` (number)
: How long should be a stem over a rest?

`stencil` (stencil)
: The symbol to print.

`stencils` (list)
: Multiple stencils, used as intermediate value.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`text` (markup)
: Text markup. See Section “Formatting text” dans *Manuel de notation*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`thick-thickness` (number)
: Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual « pen » that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`tie-configuration` (list)
: List of `(position . dir)` pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`toward-stem-shift-in-column` (number)
: Amount by which a script is shifted toward the stem if its direction coincides with the stem direction and it is associated with a `ScriptColumn` object. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`tuplet-slur` (boolean)
: Draw a slur instead of a bracket for tuplets.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

`usable-duration-logs` (list)
: List of `duration-log`s that can be used in typesetting the grob.

`use-skylines` (boolean)
: Should skylines be used for side positioning?

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`voiced-position` (number)
: The staff-position of a voiced `Rest`, negative if the rest has `direction DOWN`.

`when` (moment)
: Global time step associated with this column.

`whiteout` (boolean-or-number)
: If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`. Usually `#f` by default.

`whiteout-style` (symbol)
: Determines the shape of the `whiteout` background. Available are `'outline`, `'rounded-box`, and the default `'box`. There is one exception: Use `'special` for `LyricHyphen`.

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

`X-align-on-main-noteheads` (boolean)
: If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

`zigzag-length` (dimension, in staff space)
: The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the spanner line can be constructed from a whole number of squiggles.

A.19 Fonctions musicales prédéfinies

`\absolute` [music] - *music* (music)
: Make *music* absolute. This does not actually change the music itself but rather hides it from surrounding `\relative` and `\fixed` commands.

`\acciaccatura` [music] - *music* (music)
: Create an acciaccatura from the following music expression

`\accidentalStyle` [music] - *style* (symbol list)
: Set accidental style to symbol list *style* in the form '`piano-cautionary`'. If *style* has a form like '`Staff.piano-cautionary`', the settings are applied to that context. Otherwise, the context defaults to '`Staff`', except for piano styles, which use '`GrandStaff`' as a context.

`\addChordShape` [void] - *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair)
: Add chord shape *shape-definition* to the *chord-shape-table* hash with the key `(cons key-symbol tuning)`.

`\addInstrumentDefinition` [void] - *name* (string) *lst* (list)
: Create instrument *name* with properties *list*.

`\addQuote` [void] - *name* (string) *music* (music)
: Define *music* as a quotable music expression named *name*

`\afterGrace` [music] - *fraction* [non-negative rational, fraction, or moment] *main* (music) *grace* (music)
: Create *grace* note(s) after a *main* music expression.

  The musical position of the grace expression is after a given fraction of the main note's duration has passed. If *fraction* is not specified as first argument, it is taken from `afterGraceFraction` which has a default value of `3/4`.

`\allowPageTurn` [music]
: Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

`\allowVoltaHook` [void] - *bar* (string)
: Allow the volta bracket hook being drawn over bar line *bar*.

`\alterBroken` [music] - *property* (key list or symbol) *arg* (list) *item* (key list or music)
: Override *property* for pieces of broken spanner *item* with values *arg*. *item* may either be music in the form of a starting spanner event, or a symbol list in the form '`Context.Grob`' or just '`Grob`'. Iff *item* is in the form of a spanner event, *property* may also have the form '`Grob.property`' for specifying a directed tweak.

`\ambitusAfter` [music] - *target* (symbol)
: Move the ambitus after the break-align symbol *target*.

`\appendToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
: Append *more* to the `elements` of all music expressions in *music* that are tagged with *tag*.

`\applyContext` [music] - *proc* (procedure)
: Modify context properties with Scheme procedure *proc*.

`\applyMusic` [music] - *func* (procedure) *music* (music)
: Apply procedure *func* to *music*.

`\applyOutput` [music] - *target* (symbol list or symbol) *proc* (procedure)
: Apply function `proc` to every layout object matched by *target* which takes the form `Context` or `Context.Grob`.

`\appoggiatura` [music] - *music* (music)
: Create an appoggiatura from *music*

`\assertBeamQuant` [music] - *l* (pair) *r* (pair)
: Testing function: check whether the beam quants *l* and *r* are correct

`\assertBeamSlope` [music] - *comp* (procedure)
: Testing function: check whether the slope of the beam is the same as `comp`

`\autoChange` [music] - *pitch* [pitch] *clef-1* [context modification] *clef-2* [context modification] *music* (music)
: Make voices that switch between staves automatically. As an option the pitch where to switch staves may be specified. The clefs for the staves are optional as well. Setting clefs works only for implicitly instantiated staves.

`\balloonGrobText` [music] - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
: Attach *text* to *grob-name* at offset *offset* (use like `\once`)

`\balloonText` [post event] - *offset* (pair of numbers) *text* (markup)
: Attach *text* at *offset* (use like `\tweak`)

`\bar` [music] - *type* (string)
: Insert a bar line of type *type*

`\barNumberCheck` [music] - *n* (integer)
: Print a warning if the current bar number is not *n*.

`\beamExceptions` (any type) - *music* (music)
: Extract a value suitable for setting `Timing.beamExceptions` from the given pattern with explicit beams in *music*. A bar check `|` has to be used between bars of patterns in order to reset the timing.

`\bendAfter` [post event] - *delta* (real number)
: Create a fall or doit of pitch interval *delta*.

`\bendHold` [post event] - *mus* (music)
: Sets the `'style` of a `BendSpanner` to `'hold`.

`\bendStartLevel` [post event] - *idx* (non-negative integer) *mus* (music)
Sets the `details.successive-level` of a `BendSpanner` to *idx*.

`\bookOutputName` [void] - *newfilename* (string)
Direct output for the current book block to *newfilename*.

`\bookOutputSuffix` [void] - *newsuffix* (string)
Set the output filename suffix for the current book block to *newsuffix*.

`\breathe` [music]
Insert a breath mark.

`\chordRepeats` [music] - *event-types* [list] *music* (music)
Walk through *music* putting the notes of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(string-number-event)`.

`\clef` [music] - *type* (string)
Set the current clef to *type*.

`\compoundMeter` [music] - *args* (pair)
Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

`\compressMMRests` [music] - *music* (music)
Remove the empty bars created by multi-measure rests, leaving just the first bar containing the MM rest itself.

`\crossStaff` [music] - *notes* (music)
Create cross-staff stems

`\cueClef` [music] - *type* (string)
Set the current cue clef to *type*.

`\cueClefUnset` [music]
Unset the current cue clef.

`\cueDuring` [music] - *what* (string) *dir* (direction) *main-music* (music)
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`\cueDuringWithClef` [music] - *what* (string) *dir* (direction) *clef* (string) *main-music* (music)
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`\deadNote` [music] - *note* (music)
Print *note* with a cross-shaped note head.

`\defineBarLine` [void] - *bar* (string) *glyph-list* (list)
Define bar line settings for bar line *bar*. The list *glyph-list* must have three entries which define the appearance at the end of line, at the beginning of the next line, and the span bar, respectively.

`\displayLilyMusic` [music] - *port* [output port] *music* (music)
Display the LilyPond input representation of *music* to *port*, defaulting to the console.

`\displayMusic` [music] - *port* [output port] *music* (music)
Display the internal representation of *music* to *port*, default to the console.

`\displayScheme` (any type) - *port* [output port] *expr* (any type)
: Display the internal representation of *expr* to *port*, default to the console.

`\dropNote` [music] - *num* (integer) *music* (music)
: Drop a note of any chords in *music*, in *num* position from above.

`\endSpanners` [music] - *music* (music)
: Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

`\eventChords` [music] - *music* (music)
: Compatibility function wrapping `EventChord` around isolated rhythmic events occuring since version 2.15.28, after expanding repeat chords '`q`'.

`\featherDurations` [music] - *factor* (moment) *argument* (music)
: Adjust durations of music in *argument* by rational *factor*.

`\finger` [post event] - *finger* (integer or markup)
: Apply *finger* as a fingering indication.

`\fixed` [music] - *pitch* (pitch) *music* (music)
: Use the octave of *pitch* as the default octave for *music*.

`\footnote` [music] - *mark* [markup] *offset* (pair of numbers) *footnote* (markup) *item* (symbol list or music)
: Make the markup *footnote* a footnote on *item*. The footnote is marked with a markup *mark* moved by *offset* with respect to the marked music.

  If *mark* is not given or specified as `\default`, it is replaced by an automatically generated sequence number. If *item* is a symbol list of form '`Grob`' or '`Context.Grob`', then grobs of that type will be marked at the current time step in the given context (default `Bottom`).

  If *item* is music, the music will get a footnote attached to a grob immediately attached to the event, like `\tweak` does. For attaching a footnote to an *indirectly* caused grob, write `\single\footnote`, use *item* to specify the grob, and follow it with the music to annotate.

  Like with `\tweak`, if you use a footnote on a following post-event, the `\footnote` command itself needs to be attached to the preceding note or rest as a post-event with `-`.

`\grace` [music] - *music* (music)
: Insert *music* as grace notes.

`\grobdescriptions` (any type) - *descriptions* (list)
: Create a context modification from *descriptions*, a list in the format of `all-grob-descriptions`.

`\harmonicByFret` [music] - *fret* (number) *music* (music)
: Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at *fret*.

`\harmonicByRatio` [music] - *ratio* (number) *music* (music)
: Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through *ratio*.

`\harmonicNote` [music] - *note* (music)
: Print *note* with a diamond-shaped note head.

`\harmonicsOn` [music]
: Set the default note head style to a diamond-shaped style.

`\hide` [music] - *item* (symbol list or music)
: Set *item*'s '`transparent`' property to `#t`, making it invisible while still retaining its dimensions.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`\incipit` [music] - *incipit-music* (music)
: Output *incipit-music* before the main staff as an indication of its appearance in the original music.

`\inherit-acceptability` [void] - *to* (symbol) *from* (symbol)
: When used in an output definition, will modify all context definitions such that context *to* is accepted as a child by all contexts that also accept *from*.

`\inStaffSegno` [music]
: Put the segno variant 'varsegno' at this position into the staff, compatible with the repeat command.

`\instrumentSwitch` [music] - *name* (string)
: Switch instrument to *name*, which must have been predefined with function `\addInstrumentDefinition`.

`\inversion` [music] - *around* (pitch) *to* (pitch) *music* (music)
: Invert *music* about *around* and transpose from *around* to *to*.

`\invertChords` [music] - *num* (integer) *music* (music)
: Invert any chords in *music* into their *num*-th position. (Chord inversions may be directed downwards using negative integers.)

`\keepWithTag` [music] - *tags* (symbol list or symbol) *music* (music)
: Include only elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

  Each tag may be declared as a member of at most one tag group (defined with `\tagGroup`). If none of a *music* element's tags share a tag group with one of the specified *tags*, the element is retained.

`\key` [music] - *tonic* [pitch] *pitch-alist* [list of number pairs]
: Set key to *tonic* and scale *pitch-alist*. If both are null, just generate `KeyChangeEvent`.

`\killCues` [music] - *music* (music)
: Remove cue notes from *music*.

`\label` [music] - *label* (symbol)
: Create *label* as a referrable label.

`\language` [void] - *language* (string)
: Set note names for language *language*.

`\languageRestore` [void]
: Restore a previously-saved pitchnames alist.

`\languageSaveAndChange` [void] - *language* (string)
: Store the previous pitchnames alist, and set a new one.

`\magnifyMusic` [music] - *mag* (positive number) *music* (music)
: Magnify the notation of *music* without changing the staff-size, using *mag* as a size factor. Stems, beams, slurs, ties, and horizontal spacing are adjusted automatically.

`\magnifyStaff` [music] - *mag* (positive number)
: Change the size of the staff, adjusting notation size and horizontal spacing automatically, using *mag* as a size factor.

`\makeClusters` [music] - *arg* (music)
: Display chords in *arg* as clusters.

`\makeDefaultStringTuning` [void] - *symbol* (symbol) *pitches* (list)
: This defines a string tuning *symbol* via a list of *pitches*. The *symbol* also gets registered in `defaultStringTunings` for documentation purposes.

`\mark` [music] - *label* [integer or markup]
: Make the music for the \mark command.

`\markupMap` [music] - *path* (symbol list or symbol) *markupfun* (markup-function) *music* (music)
: This applies the given markup function *markupfun* to all markup music properties matching *path* in *music*.

  For example,

```
\new Voice { g'2 c'' }
\addlyrics {
  \markupMap LyricEvent.text
             \markup \with-color #red \etc
             { Oh yes! }
}
```

`\modalInversion` [music] - *around* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Invert *music* about *around* using *scale* and transpose from *around* to *to*.

`\modalTranspose` [music] - *from* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Transpose *music* from pitch *from* to pitch *to* using *scale*.

`\musicMap` [music] - *proc* (procedure) *mus* (music)
: Apply *proc* to *mus* and all of the music it contains.

`\noPageBreak` [music]
: Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\noPageTurn` [music]
: Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\octaveCheck` [music] - *pitch* (pitch)
: Octave check.

`\offset` [music] - *property* (symbol list or symbol) *offsets* (any type) *item* (key list or music)
: Offset the default value of *property* of *item* by *offsets*. If *item* is a string, the result is `\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`\omit` [music] - *item* (symbol list or music)
: Set *item*'s '`stencil`' property to `#f`, effectively omitting it without taking up space.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`\once` [music] - *music* (music)
: Set `once` to `#t` on all layout instruction events in *music*. This will complain about music with an actual duration. As a special exception, if *music* might be the result of a `\tweak` command, no warning will be given in order to allow for `\once \propertyTweak` to work as both one-time override and proper tweak.

`\ottava` [music] - *octave* (integer)
: Set the octavation.

`\overrideProperty` [music] - *grob-property-path* (list of indexes or symbols) *value* (any type)
: Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well.

  As opposed to `\override` which overrides the context-dependent defaults with which a grob is created, this command uses `Output_property_engraver` at the grob acknowledge stage. This may be necessary for overriding values set after the initial grob creation.

`\overrideTimeSignatureSettings` [music] - *time-signature* (fraction, as pair) *base-moment* (fraction, as pair) *beat-structure* (list) *beam-exceptions* (list)
: Override `timeSignatureSettings` for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.

`\pageBreak` [music]
: Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\pageTurn` [music]
: Force a page turn between two scores or top-level markups.

`\palmMute` [music] - *note* (music)
: Print *note* with a triangle-shaped note head.

`\palmMuteOn` [music]
: Set the default note head style to a triangle-shaped style.

`\parallelMusic` [void] - *voice-ids* (list) *music* (music)
: Define parallel music sequences, separated by '`|`' (bar check signs), and assign them to the identifiers provided in *voice-ids*.

  voice-ids: a list of music identifiers (symbols containing only letters)

  music: a music sequence, containing BarChecks as limiting expressions.

  Example:

```
\parallelMusic A,B,C {
  c c | d d | e e |
  d d | e e | f f |
}
<==>
A = { c c | d d }
B = { d d | e e }
C = { e e | f f }
```

  The last bar checks in a sequence are not copied to the result in order to facilitate ending the last entry at non-bar boundaries.

`\parenthesize` [music] - *arg* (music)
: Tag *arg* to be parenthesized.

`\partCombine` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and return a music expression containing simultaneous voices, where *part1* and *part2* are combined into one voice where appropriate. Optional *chord-range* sets the distance in steps between notes that may be combined into a chord or unison.

`\partCombineDown` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed downward.

`\partCombineForce` [music] - *type* [symbol]
: Override the part-combiner.

`\partCombineUp` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed upward.

`\partial` [music] - *dur* (duration)
: Make a partial measure.

`\phrasingSlurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for phrasing slurs.

`\pitchedTrill` [music] - *main-note* (music) *secondary-note* (music)
: Print a trill with *main-note* as the main note of the trill and print *secondary-note* as a stemless note head in parentheses.

`\pointAndClickOff` [void]
: Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

`\pointAndClickOn` [void]
: Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

`\pointAndClickTypes` [void] - *types* (symbol list or symbol)
: Set a type or list of types (such as `#'note-event`) for which point-and-click info is generated.

`\preBend` [post event] - *mus* (music)
: Sets the `'style` of a `BendSpanner` to `'pre-bend`.

`\preBendHold` [post event] - *mus* (music)
: Sets the `'style` of a `BendSpanner` to `'pre-bend-hold`.

`\propertyOverride` [music] - *grob-property-path* (list of indexes or symbols) *value* (any type)
: Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\override` command.

`\propertyRevert` [music] - *grob-property-path* (list of indexes or symbols)
: Revert the grob property specified by *grob-property-path* to its previous value. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\revert` command.

`\propertySet` [music] - *property-path* (symbol list or symbol) *value* (any type)
Set the context property specified by *property-path* to *value*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\set` command.

`\propertyTweak` [music] - *prop* (key list or symbol) *value* (any type) *item* (key list or music)
Add a tweak to the following *item*, usually music. This generally behaves like `\tweak` but will turn into an `\override` when *item* is a symbol list.

In that case, *item* specifies the grob path to override. This is mainly useful when using `\propertyTweak` as as a component for building other functions like `\omit`. It is not the default behavior for `\tweak` since many input strings in `\lyricmode` can serve equally as music or as symbols which causes surprising behavior when tweaking lyrics using the less specific semantics of `\propertyTweak`.

prop can contain additional elements in which case a nested property (inside of an alist) is tweaked.

`\propertyUnset` [music] - *property-path* (symbol list or symbol)
Unset the context property specified by *property-path*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\unset` command.

`\pushToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
Add *more* to the front of `elements` of all music expressions in *music* that are tagged with *tag*.

`\quoteDuring` [music] - *what* (string) *main-music* (music)
Indicate a section of music to be quoted. *what* indicates the name of the quoted voice, as specified in an `\addQuote` command. *main-music* is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

`\raiseNote` [music] - *num* (integer) *music* (music)
Raise a note of any chords in *music*, in *num* position from below.

`\reduceChords` [music] - *music* (music)
Reduce chords contained in *music* to single notes, intended mainly for reusing music in RhythmicStaff. Does not reduce parallel music.

`\relative` [music] - *pitch* [pitch] *music* (music)
Make *music* relative to *pitch*. If *pitch* is omitted, the first note in *music* is given in absolute pitch.

`\removeWithTag` [music] - *tags* (symbol list or symbol) *music* (music)
Remove elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

`\resetRelativeOctave` [music] - *pitch* (pitch)
Set the octave inside a \relative section.

`\retrograde` [music] - *music* (music)
Return *music* in reverse order.

`\revertTimeSignatureSettings` [music] - *time-signature* (pair)
Revert `timeSignatureSettings` for time signatures of *time-signature*.

`\rightHandFinger` [post event] - *finger* (integer or markup)
Apply *finger* as a fingering indication.

`\scaleDurations` [music] - *fraction* (non-negative rational, fraction, or moment) *music* (music)
Multiply the duration of events in *music* by *fraction*.

`\settingsFrom` (any type) - *ctx* [symbol] *music* (music)
: Take the layout instruction events from *music*, optionally restricted to those applying to context type *ctx*, and return a context modification duplicating their effect.

`\shape` [music] - *offsets* (list) *item* (key list or music)
: Offset control-points of *item* by *offsets*. The argument is a list of number pairs or list of such lists. Each element of a pair represents an offset to one of the coordinates of a control-point. The y-coordinate of each number pair is scaled by staff space. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`\shiftDurations` [music] - *dur* (integer) *dots* (integer) *arg* (music)
: Change the duration of *arg* by adding *dur* to the `durlog` of *arg* and *dots* to the `dots` of *arg*.

`\single` [music] - *overrides* (music) *music* (music)
: Convert *overrides* to tweaks and apply them to *music*. This does not convert `\revert`, `\set` or `\unset`.

`\skip` [music] - *dur* (duration)
: Skip forward by *dur*.

`\slashedGrace` [music] - *music* (music)
: Create slashed graces (slashes through stems, but no slur) from the following music expression

`\slurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for slurs.

`\spacingTweaks` [music] - *parameters* (list)
: Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

`\storePredefinedDiagram` [void] - *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair)
: Add predefined fret diagram defined by *diagram-definition* for the chord pitches *chord* and the stringTuning *tuning*.

`\stringTuning` (any type) - *chord* (music)
: Convert *chord* to a string tuning. *chord* must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

`\styledNoteHeads` [music] - *style* (symbol) *heads* (symbol list or symbol) *music* (music)
: Set *heads* in *music* to *style*.

`\tabChordRepeats` [music] - *event-types* [list] *music* (music)
: Walk through *music* putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(articulation-event)`.

`\tabChordRepetition` [void]
: Include the string and fingering information in a chord repetition. This function is deprecated; try using `\tabChordRepeats` instead.

`\tag` [music] - *tags* (symbol list or symbol) *music* (music)
: Tag the following *music* with *tags* and return the result, by adding the single symbol or symbol list *tags* to the `tags` property of *music*.

`\tagGroup` [void] - *tags* (symbol list)
: Define a tag group comprising the symbols in the symbol list *tags*. Tag groups must not overlap.

`\temporary` [music] - *music* (music)
: Make any `\override` in *music* replace an existing grob property value only temporarily, restoring the old value when a corresponding `\revert` is executed. This is achieved by clearing the '`pop-first`' property normally set on `\override`s.

  An `\override`/`\revert` sequence created by using `\temporary` and `\undo` on the same music containing overrides will cancel out perfectly or cause a warning.

  Non-property-related music is ignored, warnings are generated for any property-changing music that isn't an `\override`.

`\tieDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for ties.

`\time` [music] - *beat-structure* [number list] *fraction* (fraction, as pair)
: Set *fraction* as time signature, with optional number list *beat-structure* before it.

`\times` [music] - *fraction* (fraction, as pair) *music* (music)
: Scale *music* in time by *fraction*.

`\tocItem` [music] - *label* [symbol list or symbol] *text* (markup)
: Add a line to the table of contents, using the `tocItemMarkup` paper variable markup and assigning it to *label* if one is provided. If a hierarchy of labels is given, make the current item a child of the corresponding objects.

`\transpose` [music] - *from* (pitch) *to* (pitch) *music* (music)
: Transpose *music* from pitch *from* to pitch *to*.

`\transposedCueDuring` [music] - *what* (string) *dir* (direction) *pitch* (pitch) *main-music* (music)
: Insert notes from the part *what* into a voice called `cue`, using the transposition defined by *pitch*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

`\transposition` [music] - *pitch* (pitch)
: Set instrument transposition

`\tuplet` [music] - *ratio* (fraction, as pair) *tuplet-span* [duration] *music* (music)
: Scale the given *music* to tuplets. *ratio* is a fraction that specifies how many notes are played in place of the nominal value: it will be '`3/2`' for triplets, namely three notes being played in place of two. If the optional duration *tuplet-span* is specified, it is used instead of `tupletSpannerDuration` for grouping the tuplets. For example,

  ```
  \tuplet 3/2 4 { c8 c c c c c }
  ```

  will result in two groups of three tuplets, each group lasting for a quarter note.

`\tupletSpan` [music] - *tuplet-span* [duration]
: Set `tupletSpannerDuration`, the length into which `\tuplet` without an explicit '`tuplet-span`' argument of its own will group its tuplets, to the duration *tuplet-span*. To revert to the default of not subdividing the contents of a `\tuplet` command without explicit '`tuplet-span`', use

  ```
  \tupletSpan \default
  ```

`\tweak` [music] - *prop* (key list or symbol) *value* (any type) *music* (music)
: Add a tweak to the following *music*. Layout objects created by *music* get their property *prop* set to *value*. If *prop* has the form '`Grob.property`', like with

```
\tweak Accidental.color #red cis'
```

: an indirectly created grob ('`Accidental`' is caused by '`NoteHead`') can be tweaked; otherwise only directly created grobs are affected.

: *prop* can contain additional elements in which case a nested property (inside of an alist) is tweaked.

: If *music* is an '`event-chord`', every contained '`rhythmic-event`' is tweaked instead.

`\undo` [music] - *music* (music)
: Convert `\override` and `\set` in *music* to `\revert` and `\unset`, respectively. Any reverts and unsets already in *music* cause a warning. Non-property-related music is ignored.

`\unfolded` [music] - *music* (music)
: Mask *music* until the innermost enclosing repeat is unfolded.

`\unfoldRepeats` [music] - *types* [symbol list or symbol] *music* (music)
: Force `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`, if specified in the optional symbol-list *types*. The default for *types* is an empty list, which will force any of those commands in *music* to be interpreted as `\repeat unfold`. Possible entries are `volta`, `tremolo` or `percent`. Multiple entries are possible.

`\voices` [music] - *ids* (list of indexes or symbols) *music* (music)
: Take the given key list of numbers (indicating the use of '`\voiceOne`'...) or symbols (indicating voice names, typically converted from strings by argument list processing) and assign the following `\\`-separated music to contexts according to that list. Named rather than numbered contexts can be used for continuing one voice (for the sake of spanners and lyrics), usually requiring a `\voiceOne`-style override at the beginning of the passage and a `\oneVoice` override at its end.

: The default

```
<< ... \\ ... \\ ... >>
```

: construct would correspond to

```
\voices 1,2,3 << ... \\ ... \\ ... >>
```

`\void` [void] - *arg* (any type)
: Accept a scheme argument, return a void expression. Use this if you want to have a scheme expression evaluated because of its side-effects, but its value ignored.
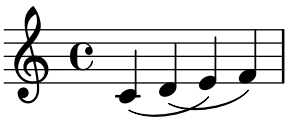
`\volta` [music] - *volta-numbers* (number list) *music* (music)
: Mark *music* as being limited to the volte given in *volta-numbers* when the innermost enclosing repeat is unfolded. Volta number begins at 1 and increases by 1 with each repetition.

`\withMusicProperty` [music] - *sym* (symbol) *val* (any type) *music* (music)
: Set *sym* to *val* in *music*.

`\xNote` [music] - *note* (music)
: Print *note* with a cross-shaped note head.

`\=` [post event] - *id* (index or symbol) *event* (post event)
: This sets the `spanner-id` property of the following *event* to the given *id* (non-negative integer or symbol). This can be used to tell LilyPond how to connect overlapping or parallel slurs or phrasing slurs within a single `Voice`.

```
\fixed c' { c\=1( d\=2( e\=1) f\=2) }
```



A.20 Identificateurs de modification de contexte

Les commandes suivantes permettent de modifier des contextes au sein d'un bloc `\layout` ou `\with`.

`\RemoveAllEmptyStaves`

Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`, including those in the first system.

- Sets grob property `remove-empty` in `Section ‘‘VerticalAxisGroup’’ dans` *Référence des propriétés internes* to `#t`.
- Sets grob property `remove-first` in `Section ‘‘VerticalAxisGroup’’ dans` *Référence des propriétés internes* to `#t`.

`\RemoveEmptyStaves`

Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`.

- Sets grob property `remove-empty` in `Section ‘‘VerticalAxisGroup’’ dans` *Référence des propriétés internes* to `#t`.

A.21 Types de prédicats prédéfinis

Predicates return `#t` when their argument is of the named type and `#f` if it isn't.

R5RS primary predicates

Primary predicates can be applied to any expression. They can be used on their own as predicates for LilyPond functions. The predicates here are part of the Scheme standard R5RS.

Type predicate	**Description**
`boolean?`	boolean
`char?`	character
`complex?`	complex number
`eof-object?`	end-of-file object
`input-port?`	input port
`integer?`	integer
`list?`	list *(use* `cheap-list?` *for faster processing)*
`null?`	null
`number?`	number
`output-port?`	output port
`pair?`	pair
`port?`	port
`procedure?`	procedure
`rational?`	rational number
`real?`	real number
`string?`	string
`symbol?`	symbol
`vector?`	vector

R5RS secondary predicates

Secondary predicates are only applicable to specific expressions (for example, to numbers). They will throw a type error when applied to expressions they are not intended for. The predicates here are part of the Scheme standard R5RS.

Type predicate	**Description**
`char-alphabetic?`	alphabetic character
`char-lower-case?`	lower-case character
`char-numeric?`	numeric character
`char-upper-case?`	upper-case character
`char-whitespace?`	whitespace character
`even?`	even number
`exact?`	exact number
`inexact?`	inexact number
`negative?`	negative number
`odd?`	odd number
`positive?`	positive number
`zero?`	zero

Guile predicates

These predicates are defined by Guile but are not part of a Scheme standard.

Type predicate	**Description**
`hash-table?`	hash table

LilyPond scheme predicates

These predicates are only available within LilyPond and defined in Scheme.

Type predicate	**Description**
`boolean-or-symbol?`	boolean or symbol
`cheap-list?`	list *(use this instead of* `list?` *for faster processing)*
`color?`	color
`fraction?`	fraction, as pair
`grob-list?`	list of grobs
`index?`	non-negative integer
`integer-or-markup?`	integer or markup
`key?`	index or symbol
`key-list?`	list of indexes or symbols
`key-list-or-music?`	key list or music
`key-list-or-symbol?`	key list or symbol
`markup?`	markup
`markup-command-list?`	markup command list
`markup-list?`	markup list
`moment-pair?`	pair of moment objects
`number-list?`	number list
`number-or-grob?`	number or grob
`number-or-pair?`	number or pair
`number-or-string?`	number or string
`number-pair?`	pair of numbers
`number-pair-list?`	list of number pairs
`rational-or-procedure?`	an exact rational or procedure

`rhythmic-location?`	rhythmic location
`scale?`	non-negative rational, fraction, or moment
`scheme?`	any type
`string-or-music?`	string or music
`string-or-pair?`	string or pair
`string-or-symbol?`	string or symbol
`symbol-list?`	symbol list
`symbol-list-or-music?`	symbol list or music
`symbol-list-or-symbol?`	symbol list or symbol
`void?`	void

LilyPond exported predicates

These predicates are only available within LilyPond and usually defined in C++.

Type predicate	**Description**
`ly:book?`	book
`ly:context?`	context
`ly:context-def?`	context definition
`ly:context-mod?`	context modification
`ly:dimension?`	dimension, in staff space
`ly:dir?`	direction
`ly:dispatcher?`	dispatcher
`ly:duration?`	duration
`ly:event?`	post event
`ly:font-metric?`	font metric
`ly:grob?`	graphical (layout) object
`ly:grob-array?`	array of grobs
`ly:grob-properties?`	grob properties
`ly:input-location?`	input location
`ly:item?`	item
`ly:iterator?`	iterator
`ly:lily-lexer?`	lily-lexer
`ly:lily-parser?`	lily-parser
`ly:listener?`	listener
`ly:moment?`	moment
`ly:music?`	music
`ly:music-function?`	music function
`ly:music-list?`	list of music objects
`ly:music-output?`	music output
`ly:otf-font?`	OpenType font
`ly:output-def?`	output definition
`ly:page-marker?`	page marker
`ly:pango-font?`	pango font
`ly:paper-book?`	paper book
`ly:paper-system?`	paper-system Prob
`ly:pitch?`	pitch
`ly:prob?`	property object
`ly:score?`	score
`ly:skyline?`	skyline
`ly:skyline-pair?`	pair of skylines
`ly:source-file?`	source file

`ly:spanner?`	spanner
`ly:spring?`	spring
`ly:stencil?`	stencil
`ly:stream-event?`	stream event
`ly:transform?`	coordinate transform
`ly:translator?`	translator
`ly:translator-group?`	translator group
`ly:unpure-pure-container?`	unpure/pure container

A.22 Fonctions Scheme

`ly:add-context-mod` *contextmods modification* [Fonction]
Adds the given context *modification* to the list *contextmods* of context modifications.

`ly:add-file-name-alist` *alist* [Fonction]
Add mappings for error messages from *alist*.

`ly:add-interface` *iface desc props* [Fonction]
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *callback disp cl* [Fonction]
Add the single-argument procedure *callback* as listener to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it calls *callback* with it.

`ly:add-option` *sym val description* [Fonction]
Add a program option *sym*. *val* is the default value and *description* is a string description.

`ly:all-grob-interfaces` [Fonction]
Return the hash table with all grob interface descriptions.

`ly:all-options` [Fonction]
Get all option settings in an alist.

`ly:all-stencil-expressions` [Fonction]
Return all symbols recognized as stencil expressions.

`ly:angle` *x y* [Fonction]
Calculates angle in degrees of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:assoc-get` *key alist default-value strict-checking* [Fonction]
Return value if *key* in *alist*, else *default-value* (or `#f` if not specified). If *strict-checking* is set to `#t` and *key* is not in *alist*, a programming_error is output.

`ly:axis-group-interface::add-element` *grob grob-element* [Fonction]
Set *grob* the parent of *grob-element* on all axes of *grob*.

`ly:basic-progress` *str rest* [Fonction]
A Scheme callable function to issue a basic progress message *str*. The message is formatted with `format` and *rest*.

`ly:book?` *x* [Fonction]
Is *x* a `Book` object?

`ly:book-add-bookpart!` *book-smob book-part* [Fonction]
Add *book-part* to *book-smob* book part list.

`ly:book-add-score!` *book-smob score* [Fonction]
Add *score* to *book-smob* score list.

`ly:book-book-parts` *book* [Fonction]
Return book parts in *book*.

`ly:book-header` *book* [Fonction]
Return header in *book*.

`ly:book-paper` *book* [Fonction]
Return paper in *book*.

`ly:book-process` *book-smob default-paper default-layout output* [Fonction]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-process-to-systems` *book-smob default-paper default-layout output* [Fonction]
Print book. *output* is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

`ly:book-scores` *book* [Fonction]
Return scores in *book*.

`ly:book-set-header!` *book module* [Fonction]
Set the book header.

`ly:bp` *num* [Fonction]
num bigpoints (1/72th inch).

`ly:bracket` *a iv t p* [Fonction]
Make a bracket in direction *a*. The extent of the bracket is given by *iv*. The wings protrude by an amount of *p*, which may be negative. The thickness is given by *t*.

`ly:broadcast` *disp ev* [Fonction]
Send the stream event *ev* to the dispatcher *disp*.

`ly:camel-case->lisp-identifier` *name-sym* [Fonction]
Convert `FooBar_Bla` to `foo-bar-bla` style symbol.

`ly:chain-assoc-get` *key achain default-value strict-checking* [Fonction]
Return value for *key* from a list of alists *achain*. If no entry is found, return *default-value* or `#f` if *default-value* is not specified. With *strict-checking* set to `#t`, a programming_error is output in such cases.

`ly:check-expected-warnings` [Fonction]
Check whether all expected warnings have really been triggered.

`ly:cm` *num* [Fonction]
num cm.

`ly:command-line-code` [Fonction]
The Scheme code specified on command-line with `-e`.

`ly:command-line-options` [Fonction]
The Scheme options specified on command-line with `-d`.

`ly:connect-dispatchers` *to from* [Fonction]
Make the dispatcher *to* listen to events from *from*.

`ly:context?` *x* [Fonction]
Is *x* a `Context` object?

`ly:context-current-moment` *context* [Fonction]
Return the current moment of *context*.

`ly:context-def?` *x* [Fonction]
Is *x* a `Context_def` object?

`ly:context-def-lookup` *def sym val* [Fonction]
Return the value of *sym* in context definition *def* (e.g., `\Voice`). If no value is found, return *val* or `'()` if *val* is undefined. *sym* can be any of '`default-child`', '`consists`', '`description`', '`aliases`', '`accepts`', '`property-ops`', '`context-name`', '`group-type`'.

`ly:context-def-modify` *def mod* [Fonction]
Return the result of applying the context-mod *mod* to the context definition *def*. Does not change *def*.

`ly:context-event-source` *context* [Fonction]
Return `event-source` of context *context*.

`ly:context-events-below` *context* [Fonction]
Return a `stream-distributor` that distributes all events from *context* and all its subcontexts.

`ly:context-find` *context name* [Fonction]
Find a parent of *context* that has name or alias *name*. Return `#f` if not found.

`ly:context-grob-definition` *context name* [Fonction]
Return the definition of *name* (a symbol) within *context* as an alist.

`ly:context-id` *context* [Fonction]
Return the ID string of *context*, i.e., for `\context Voice = "one" ...` return the string `one`.

`ly:context-matched-pop-property` *context grob cell* [Fonction]
This undoes a particular `\override`, `\once \override` or `\once \revert` when given the specific alist pair to undo.

`ly:context-mod?` *x* [Fonction]
Is *x* a `Context_mod` object?

`ly:context-mod-apply!` *context mod* [Fonction]
Apply the context modification *mod* to *context*.

`ly:context-name` *context* [Fonction]
Return the name of *context*, i.e., for `\context Voice = "one" ...` return the symbol `Voice`.

`ly:context-now` *context* [Fonction]
Return `now-moment` of context *context*.

`ly:context-parent` *context* [Fonction]
Return the parent of *context*, `#f` if none.

`ly:context-property` *context sym def* [Fonction]
Return the value for property *sym* in *context*. If *def* is given, and property value is `'()`, return *def*.

`ly:context-property-where-defined` *context name* [Fonction]
Return the context above *context* where *name* is defined.

`ly:context-pushpop-property` *context grob eltprop val* [Fonction]
Do `\temporary \override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).

`ly:context-set-property!` *context name val* [Fonction]
Set value of property *name* in context *context* to *val*.

`ly:context-unset-property` *context name* [Fonction]
Unset value of property *name* in context *context*.

`ly:debug` *str rest* [Fonction]
A Scheme callable function to issue a debug message *str*. The message is formatted with `format` and *rest*.

`ly:default-scale` [Fonction]
Get the global default scale.

`ly:dimension?` *d* [Fonction]
Is *d* a dimension? Used to distinguish length variables from normal numbers.

`ly:dir?` *s* [Fonction]
Is *s* a direction? Valid directions are `-1`, `0`, or `1`, where `-1` represents left or down, `1` represents right or up, and `0` represents a neutral direction.

`ly:directed` *direction magnitude* [Fonction]
Calculates an `(x . y)` pair with optional *magnitude* (defaulting to `1.0`) and *direction* specified either as an angle in degrees or a coordinate pair giving the direction. If *magnitude* is a pair, the respective coordinates are scaled independently, useful for ellipse drawings.

`ly:disconnect-dispatchers` *to from* [Fonction]
Stop the dispatcher *to* listening to events from *from*.

`ly:dispatcher?` *x* [Fonction]
Is *x* a `Dispatcher` object?

`ly:duration?` *x* [Fonction]
Is *x* a `Duration` object?

`ly:duration<?` *p1 p2* [Fonction]
Is *p1* shorter than *p2*?

`ly:duration->string` *dur* [Fonction]
Convert *dur* to a string.

`ly:duration-dot-count` *dur* [Fonction]
Extract the dot count from *dur*.

`ly:duration-factor` *dur* [Fonction]
Extract the compression factor from *dur*. Return it as a pair.

`ly:duration-length` *dur* [Fonction]
The length of the duration as a `moment`.

`ly:duration-log` *dur* [Fonction]
Extract the duration log from *dur*.

`ly:duration-scale` *dur* [Fonction]
Extract the compression factor from *dur*. Return it as a rational.

`ly:effective-prefix` [Fonction]
Return effective prefix.

`ly:encode-string-for-pdf` *str* [Fonction]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).

`ly:engraver-announce-end-grob` *engraver grob cause* [Fonction]
Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.

`ly:engraver-make-grob` *engraver grob-name cause* [Fonction]
Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.

`ly:error` *str rest* [Fonction]
A Scheme callable function to issue the error *str*. The error is formatted with `format` and *rest*.

`ly:event?` *obj* [Fonction]
Is *obj* a proper (non-rhythmic) event object?

`ly:event-deep-copy` *m* [Fonction]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym val* [Fonction]
Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or `'()` if *val* is not specified.

`ly:event-set-property!` *ev sym val* [Fonction]
Set property *sym* in event *ev* to *val*.

`ly:expect-warning` *str rest* [Fonction]
A Scheme callable function to register a warning to be expected and subsequently suppressed. If the warning is not encountered, a warning about the missing warning will be shown. The message should be translated with `(_ ...)` and changing parameters given after the format string.

`ly:extract-subfont-from-collection` *collection-file-name idx subfont-file-name* [Fonction]
Extract the subfont of index *idx* in TrueType collection (TTC) or OpenType/CFF collection (OTC) file *collection_file_name* and write it to file *subfont_file_name*.

`ly:find-file` *name* [Fonction]
Return the absolute file name of *name*, or `#f` if not found.

`ly:font-config-add-directory` *dir* [Fonction]
Add directory *dir* to FontConfig.

`ly:font-config-add-font` *font* [Fonction]
Add font *font* to FontConfig.

`ly:font-config-display-fonts` [Fonction]
Dump a list of all fonts visible to FontConfig.

`ly:font-config-get-font-file` *name* [Fonction]
Get the file for font *name*.

`ly:font-design-size` *font* [Fonction]
Given the font metric *font*, return the design size, relative to the current output-scale.

`ly:font-file-name` *font* [Fonction]
Given the font metric *font*, return the corresponding file name.

`ly:font-get-glyph` *font name* [Fonction]
Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-charcode` *font name* [Fonction]
Return the character code for glyph *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-glyph-name-to-index` *font name* [Fonction]
Return the index for *name* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-index-to-charcode` *font index* [Fonction]
Return the character code for *index* in *font*.

Note that this command can only be used to access glyphs from fonts loaded with `ly:system-font-load`; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings `fetaMusic` and `fetaBraces`, respectively.

`ly:font-magnification` *font* [Fonction]
Given the font metric *font*, return the magnification, relative to the current output-scale.

`ly:font-metric?` *x* [Fonction]
Is *x* a `Font_metric` object?

`ly:font-name` *font* [Fonction]
Given the font metric *font*, return the corresponding name.

`ly:font-sub-fonts` *font* [Fonction]
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

`ly:format` *str rest* [Fonction]
LilyPond specific format, supporting `~a` and `~[0-9]f`. Basic support for `~s` is also provided.

`ly:format-output` *context* [Fonction]
Given a global context in its final state, process it and return the `Music_output` object in its final state.

`ly:generic-bound-extent` *grob common* [Fonction]
Determine the extent of *grob* relative to *common* along the X axis, finding its extent as a bound when it a has `bound-alignment-interfaces` property list set and otherwise the full extent.

`ly:get-all-function-documentation` [Fonction]
Get a hash table with all LilyPond Scheme extension functions.

`ly:get-all-translators` [Fonction]
Return a list of all translator objects that may be instantiated.

`ly:get-cff-offset` *font-file-name idx* [Fonction]
Get the offset of 'CFF' table for *font_file_name*, returning it as an integer. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:get-context-mods` *contextmod* [Fonction]
Returns the list of context modifications stored in *contextmod*.

`ly:get-font-format` *font-file-name idx* [Fonction]
Get the font format for *font_file_name*, returning it as a symbol. The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:get-option` *var* [Fonction]
Get a global option setting.

`ly:get-spacing-spec` *from-scm to-scm* [Fonction]
Return the spacing spec going between the two given grobs, *from_scm* and *to_scm*.

`ly:gettext` *original* [Fonction]
A Scheme wrapper function for `gettext`.

`ly:grob?` *x* [Fonction]
Is *x* a `Grob` object?

`ly:grob-alist-chain` *grob global* [Fonction]
Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, `font-defaults` from the layout block is taken.

`ly:grob-array?` *x* [Fonction]
Is *x* a `Grob_array` object?

`ly:grob-array->list` *grob-arr* [Fonction]
Return the elements of *grob-arr* as a Scheme list.

`ly:grob-array-length` *grob-arr* [Fonction]
Return the length of *grob-arr*.

`ly:grob-array-ref` *grob-arr index* [Fonction]
Retrieve the *index*th element of *grob-arr*.

`ly:grob-basic-properties` *grob* [Fonction]
Get the immutable properties of *grob*.

`ly:grob-chain-callback` *grob proc sym* [Fonction]
Find the callback that is stored as property *sym* of grob *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.

`ly:grob-common-refpoint` *grob other axis* [Fonction]
Find the common refpoint of *grob* and *other* for *axis*.

`ly:grob-common-refpoint-of-array` *grob others axis* [Fonction]
Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

`ly:grob-default-font` *grob* [Fonction]
Return the default font for grob *grob*.

`ly:grob-extent` *grob refp axis* [Fonction]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-get-vertical-axis-group-index` *grob* [Fonction]
Get the index of the vertical axis group the grob *grob* belongs to; return `-1` if none is found.

`ly:grob-interfaces` *grob* [Fonction]
Return the interfaces list of grob *grob*.

`ly:grob-layout` *grob* [Fonction]
Get `\layout` definition from grob *grob*.

`ly:grob-object` *grob sym* [Fonction]
Return the value of a pointer in grob *grob* of property *sym*. It returns `'()` (end-of-list) if *sym* is undefined in *grob*.

`ly:grob-original` *grob* [Fonction]
Return the unbroken original grob of *grob*.

`ly:grob-parent` *grob axis* [Fonction]
Get the parent of *grob*. *axis* is 0 for the X-axis, 1 for the Y-axis.

`ly:grob-pq<?` *a b* [Fonction]
Compare two grob priority queue entries. This is an internal function.

`ly:grob-properties?` *x* [Fonction]
Is *x* a `Grob_properties` object?

`ly:grob-property` *grob sym val* [Fonction]
Return the value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-property-data` *grob sym* [Fonction]
Return the value for property *sym* of *grob*, but do not process callbacks.

`ly:grob-pure-height` *grob refp beg end val* [Fonction]
Return the pure height of *grob* given refpoint *refp*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-pure-property` *grob sym beg end val* [Fonction]
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-relative-coordinate` *grob refp axis* [Fonction]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-robust-relative-extent` *grob refp axis* [Fonction]
Get the extent in *axis* direction of *grob* relative to the grob *refp*, or `(0,0)` if empty.

`ly:grob-script-priority-less` *a b* [Fonction]
Compare two grobs by script priority. For internal use.

`ly:grob-set-nested-property!` *grob symlist val* [Fonction]
Set nested property *symlist* in grob *grob* to value *val*.

`ly:grob-set-object!` *grob sym val* [Fonction]
Set *sym* in grob *grob* to value *val*.

`ly:grob-set-parent!` *grob axis parent-grob* [Fonction]
Set *parent-grob* the parent of grob *grob* in axis *axis*.

`ly:grob-set-property!` *grob sym val* [Fonction]
Set *sym* in grob *grob* to value *val*.

`ly:grob-spanned-rank-interval` *grob* [Fonction]
Returns a pair with the `rank` of the furthest left column and the `rank` of the furthest right column spanned by `grob`.

`ly:grob-staff-position` *sg* [Fonction]
Return the Y-position of *sg* relative to the staff.

`ly:grob-suicide!` *grob* [Fonction]
Kill *grob*.

`ly:grob-system` *grob* [Fonction]
Return the system grob of *grob*.

`ly:grob-translate-axis!` *grob d a* [Fonction]
Translate *grob* on axis *a* over distance *d*.

`ly:grob-vertical<?` *a b* [Fonction]
Does *a* lie above *b* on the page?

`ly:gulp-file` *name size* [Fonction]
Read *size* characters from the file *name*, and return its contents in a string. If *size* is undefined, the entire file is read. The file is looked up using the search path.

`ly:gulp-file-utf8` *name size* [Fonction]
Read *size* characters from the file *name*, and return its contents in a string decoded from UTF-8. If *size* is undefined, the entire file is read. The file is looked up using the search path.

`ly:has-glyph-names?` *font-file-name idx* [Fonction]
Does the font for *font_file_name* have glyph names? The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:hash-table-keys` *tab* [Fonction]
Return a list of keys in *tab*.

`ly:inch` *num* [Fonction]
num inches.

`ly:input-both-locations` *sip* [Fonction]
Return input location in *sip* as `(file-name first-line first-column last-line last-column)`.

`ly:input-file-line-char-column` *sip* [Fonction]
Return input location in *sip* as `(file-name line char column)`.

`ly:input-location?` *x* [Fonction]
Is *x* a `Input` object?

`ly:input-message` *sip msg rest* [Fonction]
Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:input-warning` *sip msg rest* [Fonction]
Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:interpret-music-expression` *mus ctx* [Fonction]
Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.

`ly:intlog2` *d* [Fonction]
The 2-logarithm of $1/d$.

`ly:item?` *g* [Fonction]
Is *g* an `Item` object?

`ly:item-break-dir` *it* [Fonction]
The break status direction of item *it*. `-1` means end of line, `0` unbroken, and `1` beginning of line.

`ly:item-get-column` *it* [Fonction]
Return the `PaperColumn` or `NonMusicalPaperColumn` associated with this `Item`.

`ly:iterator?` *x* [Fonction]
Is *x* a `Music_iterator` object?

`ly:length` *x y* [Fonction]
Calculates magnitude of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:lily-lexer?` *x* [Fonction]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Fonction]
Is *x* a `Lily_parser` object?

`ly:line-interface::line` *grob startx starty endx endy* [Fonction]
Make a line using layout information from grob *grob*.

`ly:listened-event-class?` *disp cl* [Fonction]
Does *disp* listen to any event type in the list *cl*?

`ly:listened-event-types` *disp* [Fonction]
Return a list of all event types that *disp* listens to.

`ly:listener?` *x* [Fonction]
Is *x* a `Listener` object?

`ly:make-book` *paper header scores* [Fonction]
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.

`ly:make-book-part` *scores* [Fonction]
Make a `\bookpart` containing `\scores`.

`ly:make-context-mod` *mod-list* [Fonction]
Creates a context modification, optionally initialized via the list of modifications *mod-list*.

`ly:make-dispatcher` [Fonction]
Return a newly created dispatcher.

`ly:make-duration` *length dotcount num den* [Fonction]
length is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.

The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

`ly:make-global-context` *output-def* [Fonction]
Set up a global interpretation context, using the output block *output-def*. The context is returned.

`ly:make-global-translator` *global* [Fonction]
Create a translator group and connect it to the global context *global*. The translator group is returned.

`ly:make-grob-properties` *alist* [Fonction]
This packages the given property list *alist* in a grob property container stored in a context property with the name of a grob.

`ly:make-moment` *m g gn gd* [Fonction]
Create the moment with rational main timing *m*, and optional grace timing *g*.

A *moment* is a point in musical time. It consists of a pair of rationals (*m*, *g*), where *m* is the timing for the main notes, and *g* the timing for grace notes. In absence of grace notes, *g* is zero.

For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.

`ly:make-music` *props* [Fonction]
Make a C++ `Music` object and initialize it with *props*.

This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.

`ly:make-music-function` *signature func* [Fonction]
Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*'s cdr is a list containing either `ly:music?` predicates or other type predicates. Its car is the syntax function to call.

`ly:make-music-relative!` *music pitch* [Fonction]
Make *music* relative to *pitch*, return final pitch.

`ly:make-output-def` [Fonction]
Make an output definition.

`ly:make-page-label-marker` *label* [Fonction]
Return page marker with label *label*.

`ly:make-page-permission-marker` *symbol permission* [Fonction]
Return page marker with page breaking and turning permissions.

`ly:make-pango-description-string` *chain size* [Fonction]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.

`ly:make-paper-outputter` *port alist default-callback* [Fonction]
Create an outputter dumping to `port`. `alist` should map symbols to procedures. See `output-ps.scm` for an example. If `default_callback` is given, it is called for unsupported expressions

`ly:make-pitch` *octave note alter* [Fonction]
octave is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob` *type init rest* [Fonction]
Create a `Prob` object.

`ly:make-rotation` *angle center* [Fonction]
Make a transform rotating by *angle* in degrees. If *center* is given as a pair of coordinates, it is the center of the rotation, otherwise the rotation is around (0 . 0).

`ly:make-scale` *steps* [Fonction]
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

`ly:make-scaling` *scale scaley* [Fonction]
Create a scaling transform from argument *scale* and optionally *scaley*. When both arguments are given, they must be real and give the scale in *x* and *y* direction. If only *scale* is given, it may also be complex to indicate a scaled rotation in the manner of complex number rotations, or a pair of reals for specifying different scales in *x* and *y* direction like with the first calling convention.

`ly:make-score` *music* [Fonction]
Return score with *music* encapsulated in it.

`ly:make-spring` *ideal min-dist* [Fonction]
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.

`ly:make-stencil` *expr xext yext* [Fonction]
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example `scm/output-ps.scm`.
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use `empty-interval` as its value), it is taken to be empty.

`ly:make-stream-event` *cl proplist* [Fonction]
Create a stream event of class *cl* with the given mutable property list.

`ly:make-transform` *xx yx xy yy x0 y0* [Fonction]
Create a transform. Without options, it is an identity transform. Given four arguments *xx*, *yx*, *xy*, and *yy*, it is a linear transform, given six arguments (with *x0* and *y0* last), it is an affine transform. Transforms can be called as functions on other transforms (concatening them) or on points given either as complex number or real number pair. See also `ly:make-rotation`, `ly:make-scaling`, and `ly:make-translation`.

`ly:make-translation` *x y* [Fonction]
Make a transform translating by *x* and *y*. If only *x* is given, it can also be a complex number or a pair of numbers indicating the offset to use.

`ly:make-unpure-pure-container` *unpure pure* [Fonction]
Make an unpure-pure container. *unpure* should be an unpure expression, and *pure* should be a pure expression. If *pure* is omitted, the value of *unpure* will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.

`ly:message` *str rest* [Fonction]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.

`ly:minimal-breaking` *pb* [Fonction]
Break (pages and lines) the `Paper_book` object *pb* without looking for optimal spacing: stack as many lines on a page before moving to the next one.

`ly:mm` *num* [Fonction]
num mm.

`ly:module->alist` *mod* [Fonction]
Dump the contents of module *mod* as an alist.

`ly:module-copy` *dest src* [Fonction]
Copy all bindings from module *src* into *dest*.

`ly:modules-lookup` *modules sym def* [Fonction]
Look up *sym* in the list *modules*, returning the first occurence. If not found, return *def* or `#f` if *def* isn't specified.

`ly:moment?` *x* [Fonction]
Is *x* a `Moment` object?

`ly:moment<?` *a b* [Fonction]
Compare two moments.

`ly:moment-add` *a b* [Fonction]
Add two moments.

`ly:moment-div` *a b* [Fonction]
Divide two moments.

`ly:moment-grace` *mom* [Fonction]
Extract grace timing as a rational number from *mom*.

`ly:moment-grace-denominator` *mom* [Fonction]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Fonction]
Extract numerator from grace timing.

`ly:moment-main` *mom* [Fonction]
Extract main timing as a rational number from *mom*.

`ly:moment-main-denominator` *mom* [Fonction]
Extract denominator from main timing.

`ly:moment-main-numerator` *mom* [Fonction]
Extract numerator from main timing.

`ly:moment-mod` *a b* [Fonction]
Modulo of two moments.

`ly:moment-mul` *a b* [Fonction]
Multiply two moments.

`ly:moment-sub` *a b* [Fonction]
Subtract two moments.

`ly:music?` *obj* [Fonction]
Is *obj* a music object?

`ly:music-compress` *m factor* [Fonction]
Compress music object *m* by scale *factor*.

`ly:music-deep-copy` *m origin* [Fonction]
Copy *m* and all sub expressions of *m*. *m* may be an arbitrary type; cons cells and music are copied recursively. If *origin* is given, it is used as the origin for one level of music by calling `ly:set-origin!` on the copy.

`ly:music-duration-compress` *mus fact* [Fonction]
Compress *mus* by factor *fact*, which is a `Moment`.

`ly:music-duration-length` *mus* [Fonction]
Extract the duration field from *mus* and return the length.

`ly:music-function?` *x* [Fonction]
Is *x* a `Music_function` object?

`ly:music-function-extract` *x* [Fonction]
Return the Scheme function inside *x*.

`ly:music-function-signature` *x* [Fonction]
Return the function signature inside *x*.

`ly:music-length` *mus* [Fonction]
Get the length of music expression *mus* and return it as a `Moment` object.

`ly:music-list?` *lst* [Fonction]
Is *lst* a list of music objects?

`ly:music-mutable-properties` *mus* [Fonction]
Return an alist containing the mutable properties of *mus*. The immutable properties are not available, since they are constant and initialized by the `make-music` function.

`ly:music-output?` *x* [Fonction]
Is *x* a `Music_output` object?

`ly:music-property` *mus sym val* [Fonction]
Return the value for property *sym* of music expression *mus*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:music-set-property!` *mus sym val* [Fonction]
Set property *sym* in music expression *mus* to *val*.

`ly:music-start` *mus* [Fonction]
Get the start of music expression *mus* and return it as a `Moment` object.

`ly:music-transpose` *m p* [Fonction]
Transpose *m* such that central C is mapped to *p*. Return *m*.

`ly:note-column-accidentals` *note-column* [Fonction]
Return the `AccidentalPlacement` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-column-dot-column` *note-column* [Fonction]
Return the `DotColumn` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-head::stem-attachment` *font-metric glyph-name direction* [Fonction]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name* in the direction *direction* (default UP).

`ly:number->string` *s* [Fonction]
Convert *s* to a string without generating many decimals.

`ly:one-line-auto-height-breaking` *pb* [Fonction]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.

`ly:one-line-breaking` *pb* [Fonction]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.

`ly:one-page-breaking` *pb* [Fonction]
Put each score on a single page. The paper-height settings are modified so each score fits on one page, and the height of the page matches the height of the full score.

`ly:optimal-breaking` *pb* [Fonction]
Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in bother vertical and horizontal spacing.

`ly:option-usage` *port* [Fonction]
Print `ly:set-option` usage. Optional *port* argumentfor the destination defaults to current output port.

`ly:otf->cff` *otf-file-name idx* [Fonction]
Convert the contents of an OTF file to a CFF file, returning it as a string. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:otf-font?` *font* [Fonction]
Is *font* an OpenType font?

`ly:otf-font-glyph-info` *font glyph* [Fonction]
Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).

`ly:otf-font-table-data` *font tag* [Fonction]
Extract a table *tag* from *font*. Return empty string for non-existent *tag*.

`ly:otf-glyph-count` *font* [Fonction]
Return the number of glyphs in *font*.

`ly:otf-glyph-list` *font* [Fonction]
Return a list of glyph names for *font*.

`ly:output-def?` *x* [Fonction]
Is *x* a `Output_def` object?

`ly:output-def-clone` *def* [Fonction]
Clone output definition *def*.

`ly:output-def-lookup` *def sym val* [Fonction]
Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `'()` if *val* is undefined.

`ly:output-def-parent` *def* [Fonction]
Return the parent output definition of *def*.

`ly:output-def-scope` *def* [Fonction]
Return the variable scope inside *def*.

`ly:output-def-set-variable!` *def sym val* [Fonction]
Set an output definition *def* variable *sym* to *val*.

`ly:output-description` *output-def* [Fonction]
Return the description of translators in *output-def*.

`ly:output-find-context-def` *output-def context-name* [Fonction]
Return an alist of all context defs (matching *context-name* if given) in *output-def*.

`ly:output-formats` [Fonction]
Formats passed to `--format` as a list of strings, used for the output.

`ly:outputter-close` *outputter* [Fonction]
Close port of *outputter*.

`ly:outputter-dump-stencil` *outputter stencil* [Fonction]
Dump stencil *expr* onto *outputter*.

`ly:outputter-dump-string` *outputter str* [Fonction]
Dump *str* onto *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Fonction]
Output *expr* to the paper outputter.

`ly:outputter-port` *outputter* [Fonction]
Return output port for *outputter*.

`ly:page-marker?` *x* [Fonction]
Is *x* a `Page_marker` object?

`ly:page-turn-breaking` *pb* [Fonction]
Optimally break (pages and lines) the `Paper_book` object *pb* such that page turns only happen in specified places, returning its pages.

`ly:pango-font?` *f* [Fonction]
Is *f* a pango font?

`ly:pango-font-physical-fonts` *f* [Fonction]
Return alist of `(ps-name file-name font-index)` lists for Pango font *f*.

`ly:paper-book?` *x* [Fonction]
Is *x* a `Paper_book` object?

`ly:paper-book-header` *pb* [Fonction]
Return the header definition (`\header`) in `Paper_book` object *pb*.

`ly:paper-book-pages` *pb* [Fonction]
Return pages in `Paper_book` object *pb*.

`ly:paper-book-paper` *pb* [Fonction]
Return the paper output definition (`\paper`) in `Paper_book` object *pb*.

`ly:paper-book-performances` *pb* [Fonction]
Return performances in `Paper_book` object *pb*.

`ly:paper-book-scopes` *pb* [Fonction]
Return scopes in `Paper_book` object *pb*.

`ly:paper-book-systems` *pb* [Fonction]
Return systems in `Paper_book` object *pb*.

`ly:paper-column::break-align-width` *col align-syms* [Fonction]
Determine the extent along the X-axis of a grob used for break-alignment organized by column *col*. The grob is specified by *align-syms*, which contains either a single `break-align-symbol` or a list of such symbols.

`ly:paper-column::print` [Fonction]
Optional stencil for `PaperColumn` or `NonMusicalPaperColumn`. Draws the `rank number` of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.

`ly:paper-fonts` *def* [Fonction]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).

`ly:paper-get-font` *def chain* [Fonction]
Find a font metric in output definition *def* satisfying the font-qualifiers in alist chain *chain*, and return it. (An alist chain is a list of alists, containing grob properties.)

`ly:paper-get-number` *def sym* [Fonction]
Return the value of variable *sym* in output definition *def* as a double.

`ly:paper-outputscale` *def* [Fonction]
Return the output-scale for output definition *def*.

`ly:paper-score-paper-systems` *paper-score* [Fonction]
Return vector of `paper_system` objects from *paper-score*.

`ly:paper-system?` *obj* [Fonction]
Is *obj* a C++ `Prob` object of type `paper-system`?

`ly:paper-system-minimum-distance` *sys1 sys2* [Fonction]
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

`ly:parse-file` *name* [Fonction]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.

`ly:parse-init` *name* [Fonction]
Parse the init file `name`.

`ly:parse-string-expression` *parser-smob ly-code filename line* [Fonction]
Parse the string *ly-code* with *parser-smob*. Return the contained music expression. *filename* and *line* are optional source indicators.

`ly:parsed-undead-list!` [Fonction]
Return the list of objects that have been found live that should have been dead, and clear that list.

`ly:parser-clear-error` *parser* [Fonction]
Clear error flag for *parser*, defaulting to current parser.

`ly:parser-clone` *closures location* [Fonction]
Return a clone of current parser. An association list of port positions to closures can be specified in *closures* in order to have `$` and `#` interpreted in their original lexical environment. If *location* is a valid location, it becomes the source of all music expressions inside.

`ly:parser-define!` *symbol val* [Fonction]
Bind *symbol* to *val* in current parser's module.

`ly:parser-error` *msg input* [Fonction]
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.

`ly:parser-has-error?` *parser* [Fonction]
Does *parser* (defaulting to current parser) have an error flag?

`ly:parser-include-string` *ly-code* [Fonction]
Include the string *ly-code* into the input stream for current parser. Can only be used in immediate Scheme expressions (`$` instead of `#`).

`ly:parser-lookup` *symbol* [Fonction]
Look up *symbol* in current parser's module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Fonction]
Return the base name of the output file. If `parser` is left off, use currently active parser.

`ly:parser-parse-string` *parser-smob ly-code* [Fonction]
Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.

`ly:parser-set-note-names` *names* [Fonction]
Replace current note names in parser. *names* is an alist of symbols. This only has effect if the current mode is notes.

`ly:performance-headers` *performance* [Fonction]
Return the list of headers with the innermost first.

`ly:performance-write` *performance filename name* [Fonction]
Write *performance* to *filename* storing *name* as the name of the performance in the file metadata.

`ly:pitch?` *x* [Fonction]
Is *x* a `Pitch` object?

`ly:pitch<?` *p1 p2* [Fonction]
Is *p1* lexicographically smaller than *p2*?

`ly:pitch-alteration` *pp* [Fonction]
Extract the alteration from pitch *pp*.

`ly:pitch-diff` *pitch root* [Fonction]
Return pitch *delta* such that *root* transposed by *delta* equals *pitch*.

`ly:pitch-negate` *p* [Fonction]
Negate *p*.

`ly:pitch-notename` *pp* [Fonction]
Extract the note name from pitch *pp*.

`ly:pitch-octave` *pp* [Fonction]
Extract the octave from pitch *pp*.

`ly:pitch-quartertones` *pp* [Fonction]
Calculate the number of quarter tones of *pp* from middle C.

`ly:pitch-semitones` *pp* [Fonction]
Calculate the number of semitones of *pp* from middle C.

`ly:pitch-steps` *p* [Fonction]
Number of steps counted from middle C of the pitch *p*.

`ly:pitch-tones` *pp* [Fonction]
Calculate the number of tones of *pp* from middle C as a rational number.

`ly:pitch-transpose` *p delta* [Fonction]
Transpose *p* by the amount *delta*, where *delta* is relative to middle C.

`ly:pointer-group-interface::add-grob` *grob sym grob-element* [Fonction]
Add *grob-element* to *grob*'s *sym* grob array.

`ly:position-on-line?` *sg spos* [Fonction]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).

`ly:prob?` *x* [Fonction]
Is *x* a `Prob` object?

`ly:prob-immutable-properties` *prob* [Fonction]
Retrieve an alist of immutable properties.

`ly:prob-mutable-properties` *prob* [Fonction]
Retrieve an alist of mutable properties.

`ly:prob-property` *prob sym val* [Fonction]
Return the value for property *sym* of Prob object *prob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:prob-property?` *obj sym* [Fonction]
Is boolean prop *sym* of *sym* set?

`ly:prob-set-property!` *obj sym value* [Fonction]
Set property *sym* of *obj* to *value*.

`ly:prob-type?` *obj type* [Fonction]
Is *obj* the specified prob-type?

`ly:programming-error` *str rest* [Fonction]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format` and *rest*.

`ly:progress` *str rest* [Fonction]
A Scheme callable function to print progress *str*. The message is formatted with `format` and *rest*.

`ly:property-lookup-stats` *sym* [Fonction]
Return hash table with a property access corresponding to *sym*. Choices are `prob`, `grob`, and `context`.

`ly:protects` [Fonction]
Return hash of protected objects.

`ly:pt` *num* [Fonction]
num printer points.

`ly:pure-call` *data grob start end rest* [Fonction]
Convert property *data* (unpure-pure container or procedure) to value in a pure context defined by *grob*, *start*, *end*, and possibly *rest* arguments.

`ly:randomize-rand-seed` [Fonction]
Randomize C random generator.

`ly:register-stencil-expression` *symbol* [Fonction]
Add *symbol* as head of a stencil expression.

`ly:register-translator` *creator name description* [Fonction]
Register a translator *creator* (usually a descriptive alist or a function/closure returning one when given a context argument) with the given symbol *name* and the given *description* alist.

`ly:relative-group-extent` *elements common axis* [Fonction]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`ly:rename-file` *oldname newname* [Fonction]
Rename *oldname* to *newname*. In contrast to Guile's rename-file, this replaces the destination if it already exists. On Windows, fall back to copying the file contents if *newname* cannot be deleted.

`ly:reset-all-fonts` [Fonction]
Forget all about previously loaded fonts.

`ly:round-filled-box` *xext yext blot* [Fonction]
Make a `Stencil` object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.

`ly:round-filled-polygon` *points blot extroversion* [Fonction]
Make a `Stencil` object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*. Optional*extroversion* shifts the outline outward, with thedefault of `-1.0` keeping the outer boundary ofthe outline just inside of the polygon.

`ly:run-translator` *mus output-def* [Fonction]
Process *mus* according to *output-def*. An interpretation context is set up, and *mus* is interpreted with it. The context is returned in its final state.

Optionally, this routine takes an object-key to to uniquely identify the score block containing it.

`ly:score?` *x* [Fonction]
Is *x* a `Score` object?

`ly:score-add-output-def!` *score def* [Fonction]
Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Fonction]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

`ly:score-error?` *score* [Fonction]
Was there an error in the score?

`ly:score-header` *score* [Fonction]
Return score header.

`ly:score-music` *score* [Fonction]
Return score music.

`ly:score-output-defs` *score* [Fonction]
All output definitions in a score.

`ly:score-set-header!` *score module* [Fonction]
Set the score header.

`ly:separation-item::print` [Fonction]
Optional stencil for `PaperColumn` or `NonMusicalPaperColumn`. This function draws `horizontal-skylines` of each `PaperColumn`, showing the shapes used to determine the minimum distances between `PaperColumns` at the note-spacing step, before staves have been spaced (vertically) on the page.

`ly:set-default-scale` *scale* [Fonction]
Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.

`ly:set-grob-creation-callback` *cb* [Fonction]
Specify a procedure that will be called every time a new grob is created. The callback will receive as arguments the grob that was created, the name of the C++ source file that caused the grob to be created, and the corresponding line number in the C++ source file. Call with `#f` as argument to unset the callback.

`ly:set-grob-modification-callback` *cb* [Fonction]
Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property. Call with `#f` as argument to unset the callback.

`ly:set-middle-C!` *context* [Fonction]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.

`ly:set-option` *var val* [Fonction]
Set a program option.

`ly:set-origin!` *m origin* [Fonction]
This sets the origin given in *origin* to *m*. *m* will typically be a music expression or a list of music. List structures are searched recursively, but recursion stops at the changed music expressions themselves. *origin* is generally of type `ly:input-location?`, defaulting to `(*location*)`. Other valid values for `origin` are a music expression which is then used as the source of location information, or `#f` or `'()` in which case no action is performed. The return value is *m* itself.

`ly:set-property-cache-callback` *cb* [Fonction]
Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property. Call with `#f` as argument to unset the callback.

`ly:skyline?` *x* [Fonction]
Is *x* a `Skyline` object?

`ly:skyline-empty?` *sky* [Fonction]
Return whether *sky* is empty.

`ly:skyline-pair?` *x* [Fonction]
Is *x* a `Skyline_pair` object?

`ly:smob-protects` [Fonction]
Return LilyPond's internal smob protection list.

`ly:solve-spring-rod-problem` *springs rods length ragged* [Fonction]
Solve a spring and rod problem for *count* objects, that are connected by *count*-1 *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs* argument has the format `(ideal, inverse_hook)` and *rods* is of the form `(idx1, idx2, distance)`.

length is a number, *ragged* a boolean.

The function returns a list containing the force (positive for stretching, negative for compressing and `#f` for non-satisfied constraints) followed by *spring-count*+1 positions of the objects.

`ly:source-file?` *x* [Fonction]
Is *x* a `Source_file` object?

`ly:source-files` *parser-smob* [Fonction]
A list of input files that have been opened up to here, including the files that have been closed already. a PARSER may optionally be specified.

`ly:spanner?` *g* [Fonction]
Is *g* a spanner object?

`ly:spanner-bound` *spanner dir* [Fonction]
Get one of the bounds of *spanner*. *dir* is `-1` for left, and `1` for right.

`ly:spanner-broken-into` *spanner* [Fonction]
Return broken-into list for *spanner*.

`ly:spanner-set-bound!` *spanner dir item* [Fonction]
Set grob *item* as bound in direction *dir* for *spanner*.

`ly:spawn` *command rest* [Fonction]
Simple interface to g_spawn_sync *str*. The error is formatted with `format` and *rest*.

`ly:spring?` *x* [Fonction]
Is *x* a `Spring` object?

`ly:spring-set-inverse-compress-strength!` *spring strength* [Fonction]
Set the inverse compress *strength* of *spring*.

`ly:spring-set-inverse-stretch-strength!` *spring strength* [Fonction]
Set the inverse stretch *strength* of *spring*.

`ly:staff-symbol-line-thickness` *grob* [Fonction]
Returns the current staff-line thickness in the staff associated with *grob*, expressed as a multiple of the current staff-space height.

`ly:staff-symbol-staff-radius` *grob* [Fonction]
Returns the radius of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Fonction]
Returns the current staff-space height in the staff associated with *grob*, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`ly:stderr-redirect` *fd-or-file-name mode* [Fonction]
Redirect stderr to *fd* if the first parameter is an integer, or to *file-name*, opened with *mode*.

`ly:stencil?` *x* [Fonction]
Is *x* a `Stencil` object?

`ly:stencil-add` *args* [Fonction]
Combine stencils. Takes any number of arguments.

`ly:stencil-aligned-to` *stil axis dir* [Fonction]
Align *stil* using its own extents. *dir* is a number. `-1` and `1` are left and right, respectively. Other values are interpolated (so `0` means the center).

`ly:stencil-combine-at-edge` *first axis direction second padding* [Fonction]
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil axis* [Fonction]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.

`ly:stencil-expr` *stil* [Fonction]
Return the expression of *stil*.

`ly:stencil-extent` *stil axis* [Fonction]
Return a pair of numbers signifying the extent of *stil* in *axis* direction (`0` or `1` for x and y axis, respectively).

`ly:stencil-in-color` *stc r g b a* [Fonction]
Put *stc* in a different color.Accepts either three values for *r*,*g*, *b* and an optional valuefor *a*, or a single CSS-like string.

`ly:stencil-outline` *stil outline* [Fonction]
Return a stencil with the stencil expression (inking) of stencil *stil* but with outline and dimensions from stencil *outline*.

`ly:stencil-rotate` *stil angle x y* [Fonction]
Return a stencil *stil* rotated *angle* degrees around the relative offset (*x*, *y*). E.g., an offset of (-1, 1) will rotate the stencil around the left upper corner.

`ly:stencil-rotate-absolute` *stil angle x y* [Fonction]
Return a stencil *stil* rotated *angle* degrees around point (*x*, *y*), given in absolute coordinates.

`ly:stencil-scale` *stil x y* [Fonction]
Scale stencil *stil* using the horizontal and vertical scaling factors *x* and *y*. Negative values will flip or mirror *stil* without changing its origin; this may result in collisions unless it is repositioned.

`ly:stencil-stack` *first axis direction second padding mindist* [Fonction]
Construct a stencil by stacking *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`. As opposed to `ly:stencil-combine-at-edge`, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.

If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, *padding* and *mindist* do not apply.

`ly:stencil-translate` *stil offset* [Fonction]
Return a *stil*, but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Fonction]
Return a copy of *stil* but translated by *amount* in *axis* direction.

`ly:stream-event?` *obj* [Fonction]
Is *obj* a `Stream_event` object?

`ly:string-percent-encode` *str* [Fonction]
Encode all characters in string *str* with hexadecimal percent escape sequences, with the following exceptions: characters `-`, `.`, `/`, and `_`; and characters in ranges `0-9`, `A-Z`, and `a-z`.

`ly:string-substitute` *a b s* [Fonction]
Replace string *a* by string *b* in string *s*.

`ly:system-font-load` *name* [Fonction]
Load the OpenType system font *name*`.otf`. Fonts loaded with this command must contain three additional SFNT font tables called `LILC`, `LILF`, and `LILY`, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.

Note that only `ly:font-get-glyph` and derived code (like `\lookup`) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.

`ly:text-interface::interpret-markup` [Fonction]
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.

layout is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.

`ly:transform?` *x* [Fonction]
Is *x* a `Transform` object?

`ly:transform->list` *transform* [Fonction]
Convert a transform matrix to a list of six values. Values are *xx*, *yx*, *xy*, *yy*, *x0*, *y0*.

`ly:translate-cpp-warning-scheme` *str* [Fonction]
Translates a string in C++ printf format and modifies it to use it for scheme formatting.

`ly:translator?` *x* [Fonction]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Fonction]
Return the context of the translator object *trans*.

`ly:translator-description` *creator* [Fonction]
Return an alist of properties of translator definition *creator*.

`ly:translator-group?` *x* [Fonction]
Is *x* a `Translator_group` object?

`ly:translator-name` *creator* [Fonction]
Return the type name of the translator definition *creator*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Fonction]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:ttf->pfa` *ttf-file-name idx* [Fonction]
Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:ttf-ps-name` *ttf-file-name idx* [Fonction]
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:type1->pfa` *type1-file-name* [Fonction]
Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in PFA format, pass through it.

`ly:unit` [Fonction]
Return the unit used for lengths as a string.

`ly:unpure-call` *data grob rest* [Fonction]
Convert property *data* (unpure-pure container or procedure) to value in an unpure context defined by *grob* and possibly *rest* arguments.

`ly:unpure-pure-container?` *x* [Fonction]
Is *x* a `Unpure_pure_container` object?

`ly:unpure-pure-container-pure-part` *pc* [Fonction]
Return the pure part of *pc*.

`ly:unpure-pure-container-unpure-part` *pc* [Fonction]
Return the unpure part of *pc*.

`ly:usage` [Fonction]
Print usage message.

`ly:verbose-output?` [Fonction]
Was verbose output requested, i.e. loglevel at least `DEBUG`?
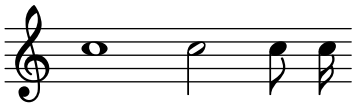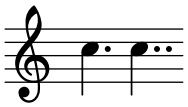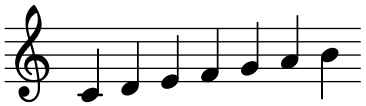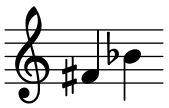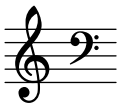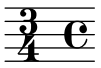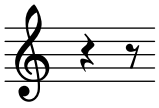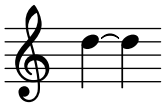
`ly:version` [Fonction]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.

`ly:warning` *str rest* [Fonction]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.

`ly:warning-located` *location str rest* [Fonction]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with `format` and *rest*.

`ly:wide-char->utf-8` *wc* [Fonction]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.
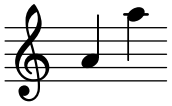
Annexe B Aide-mémoire

Syntaxe	Description	Exemple
`1 2 8 16`	valeurs rythmiques	
`c4. c4..`	notes pointées	
`c d e f g a b`	gamme	
`fis bes`	altérations	
`\clef treble \clef bass`	clés	
`\time 3/4 \time 4/4`	chiffre de mesure, métrique	
`r4 r8`	silences	
`d ~ d`	liaison de tenue	
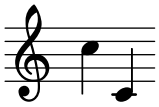
`\key es \major` armure
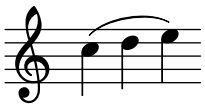


note`'` monter d'une octave
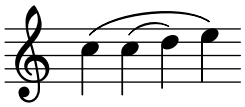


note`,` baisser d'une octave
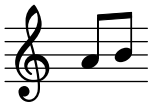


`c( d e)` liaisons



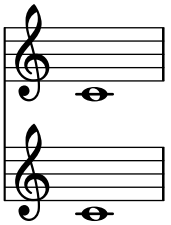
`c\( c( d) e\)` liaisons de phrasé
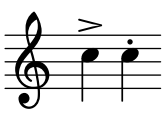


`a8[ b]` ligatures
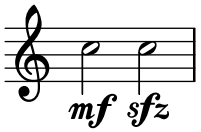


`<< \new Staff ... >>` ajouter des portées



`c-> c-.` indications d'articulation
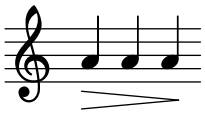


`c2\mf c\sfz` — nuances
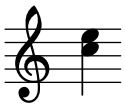


`a\< a a\!` — crescendo
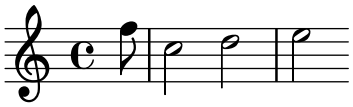


`a\> a a\!` — decrescendo
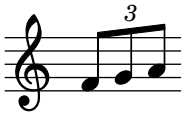


`< >` — accords
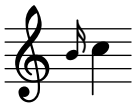


`\partial 8` — levées, anacrouses



`\tuplet 3/2 {f g a}` — triolets



`\grace` — appoggiatures

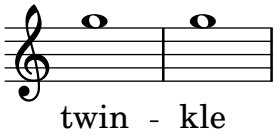


`\lyricmode { twinkle }` — ajouter des paroles : twinkle
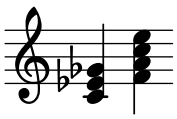
`\new Lyrics` — imprimer les paroles

twinkle

`twin -- kle` — diviser un mot en plusieurs syllabes
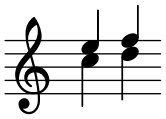


`\chordmode { c:dim f:maj7 }` accords chiffrés



`\new ChordNames` imprimer les chiffrages d'accords



`<<{e f} \\ {c d}>>` polyphonie



`s4 s8 s16` silences invisibles

Annexe C GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
`https://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Annexe D Index des commandes LilyPond

Cet index recense toutes les commandes et mots réservés de LilyPond, sous forme de lien vers les endroits du manuel où ils sont abordés.

Λ

\\ 180

!

! 6
\! 130

"

"|" 116

%

% 510, 514
%{ ... %} 510, 514

’

' 2

(

(..... 139
\(..... 141

)

) 139
\) 141

,

, ... 2

—

- 127, 466, 667
-! 805
-+ 806
-- 805
-. 805
-> 805
-^ 805
-_ 805

.

. 49

/

/ 449
/+ 449

:

: 171

<

< 173
\< 130
<...> 173
<> 174, 355

=

= 10
\= 139, 859

>

> 173
\> 130

?

? 6

[

[..... 99
\[..... 470

]

] 99
\] 470

^

^ 448, 667

_

_ 295, 667

|

| 116

~

~ 57

A

\abs-fontsize 265, 746
\absolute 848
\accent 127, 805
\accentus 482, 807
\accepts 644, 645, 646
\acciaccatura 119, 848
\accidentalStyle 29, 848
AccidentalSuggestion 128
add-grace-property 122
add-stem-support 238
add-toc-item! 540
\addChordShape 407, 848
\addInstrumentDefinition 848
additionalPitchPrefix 454
\addlyrics 289, 291, 292
\addQuote 221, 848
\aeolian 23
\afterGrace 120, 848
afterGraceFraction 812
\aikenHeads 43
\aikenHeadsMinor 44
\aikenThinHeads 43
\aikenThinHeadsMinor 44
\alias 644
alignAboveContext 211, 647
alignBelowContext 211, 311, 647
\allowPageTurn 595, 848
\allowVoltaHook 848
\alterBroken 694, 849
\alternative 157, 159
ambitusAfter 39, 849
AmbitusLine 38
annotate-spacing 624
\appendToTag 548, 849
\applyContext 633, 849
\applyMusic 849
\applyOutput 849
\applySwing 568
\applySwingWithOffset 568
\appoggiatura 119, 849
\arabicStringNumbers 370
\arpeggio 150
arpeggio-direction 150
\arpeggioArrowDown 150
\arpeggioArrowUp 150
\arpeggioBracket 151
\arpeggioNormal 150
\arpeggioParenthesis 151
\arpeggioParenthesisDashed 151
\arrow-head 274, 771
\articulate 567
articulation-event 223
\ascendens 484, 490
\assertBeamQuant 849
\assertBeamSlope 849
associatedVoice 289, 291, 323
\auctum 484, 490
aug 446
\augmentum 490
auto-first-page-number 581
\auto-footnote 794
autoBeaming 90, 629
\autoBeamOff 87, 360
\autoBeamOn 87
\autoBreaksOff 588
\autoBreaksOn 588
\autoChange 357, 849
\autoLineBreaksOff 588
\autoLineBreaksOn 588
automaticBars 682
\autoPageBreaksOff 592
\autoPageBreaksOn 592

B

\backslashed-digit 794
Balloon_engraver 248
\balloonGrobText 248, 849
\balloonLengthOff 248
\balloonLengthOn 248
\balloonText 248, 849
banjo-c-tuning 422
banjo-modal-tuning 422
banjo-open-d-tuning 422
banjo-open-dm-tuning 422
\bar 102, 109, 849
barCheckSynchronize 116
BarNumber 110
\barNumberCheck 117, 849
barNumberVisibility 110
bartype 109
base-shortest-duration 613
baseMoment 90, 94
\bassFigureExtendersOff 461
\bassFigureExtendersOn 461
\bassFigureStaffAlignmentDown 463
\bassFigureStaffAlignmentNeutral 463
\bassFigureStaffAlignmentUp 463
\beam 771
beamExceptions 90, 849
beatStructure 90, 94
\bendAfter 145, 849
\bendHold 377, 849
\bendStartLevel 377, 850
binding-offset 579
\blackTriangleMarkup 454
blank-after-score-page-penalty 581
blank-last-page-penalty 581
blank-page-penalty 581
\bold 265, 746
\book 510, 513
\bookOutputName 512, 850
\bookOutputSuffix 512, 850
\bookpart 511, 513, 592
bookTitleMarkup 523
bottom-margin 574
\box 273, 746
bracket 362
\bracket 136, 273, 771
\break 588
break-align-symbols 687
break-visibility 677
breakable 88
breakbefore 521
\breathe 143, 850
BreathingSign 144
\breve 48, 61

C

\cadenzaOff ... 78
\cadenzaOn ... 78
\caesura ... 482
\caps ... 747
\cavum ... 484, 490
\center-align ... 268, 756
\center-column ... 271, 756
\change ... 355
\char ... 794
check-consistency ... 578
choral ... 33
choral-cautionary ... 33
chordChanges ... 411, 452
\chordmode ... 14, 405, 666
chordNameExceptions ... 455
chordNameLowercaseMinor ... 453
ChordNames ... 405
chordNameSeparator ... 454, 457
chordNoteNamer ... 454
chordPrefixSpacer ... 455
\chordRepeats ... 373, 850
chordRootNamer ... 454
\chords ... 451, 666
\circle ... 273, 772
\circulus ... 482, 807
\clef ... 18, 850
clip-regions ... 552
\cm ... 669
\coda ... 127, 806
color ... 242
\column ... 271, 756
\column-lines ... 801
\combine ... 274, 757
common-shortest-duration ... 613
Completion_heads_engraver ... 83
Completion_rest_engraver ... 83
\compound-meter ... 779
\compoundMeter ... 81, 850
\compressEmptyMeasures ... 230
\compressMMRests ... 65, 66, 230, 850
\concat ... 757
\consists ... 637, 644
\context ... 630, 639
context-spec-music ... 186
controlpitch ... 10
countPercentRepeats ... 169
\cr ... 130
\cresc ... 131
crescendo-event ... 223
crescendoSpanner ... 135
crescendoText ... 135
\crescHairpin ... 131
\crescTextCresc ... 131
cross ... 40
\crossStaff ... 360, 850
\cueClef ... 224, 850
\cueClefUnset ... 224, 850
\cueDuring ... 224, 850
\cueDuringWithClef ... 224, 850
CueVoice ... 224
currentBarNumber ... 110, 125
\customTabClef ... 779

D

\dashBang ... 128
\dashDash ... 128
\dashDot ... 128
\dashHat ... 128
\dashLarger ... 128
\dashPlus ... 128
\dashUnderscore ... 128
\deadNote ... 41, 850
\deadNotesOff ... 41
\deadNotesOn ... 41
debug-beam-scoring ... 583
debug-slur-scoring ... 583
debug-tie-scoring ... 583
\decr ... 130
\decresc ... 131
decrescendoSpanner ... 135
decrescendoText ... 135
default ... 29, 31, 529
\default ... 529
default-staff-staff-spacing ... 596
\default ... 117
defaultBarType ... 109
\defaultchild ... 647
\defaultTimeSignature ... 70
\defineBarLine ... 107, 850
\deminutum ... 484, 490
\denies ... 644, 645, 646
\descendens ... 484, 490
\dim ... 131, 446
\dimHairpin ... 131
\dimTextDecr ... 131
\dimTextDecresc ... 131
\dimTextDim ... 131
\dir-column ... 757
\discant ... 789
\displayLilyMusic ... 569, 850
\displayMusic ... 850
\displayScheme ... 851
\divisioMaior ... 482
\divisioMaxima ... 482
\divisioMinima ... 482
dodecaphonic ... 35
dodecaphonic-first ... 35
dodecaphonic-no-repeat ... 35
\dorian ... 23
\dotsDown ... 49
\dotsNeutral ... 49
\dotsUp ... 49
\doubleflat ... 780
\doublesharp ... 780
\downbow ... 127, 366, 806
\downmordent ... 127, 805
\downprall ... 127, 805
\draw-circle ... 274, 772
\draw-dashed-line ... 772
\draw-dotted-line ... 773
\draw-hline ... 773
\draw-line ... 274, 773
\draw-squiggle-line ... 774
\dropNote ... 450, 851
\drummode ... 197, 423, 666
drumPitchNames ... 428
drumPitchTable ... 429

\drums 423, 666
DrumStaff 197
drumStyleTable 428
\dwn 502
\dynamic 136, 747
dynamic-event 223
\dynamicDown 132
DynamicLineSpanner 132, 135
\dynamicNeutral 132
\dynamicUp 132

E

\easyHeadsOff 41
\easyHeadsOn 41
\ellipse 774
\endcr 130
\enddecr 130
\endSpanners 675, 851
\episemFinis 482
\episemInitium 482
\epsfile 275, 775
\espressivo 127, 131, 805
\etc 700
\eventChords 851
\expandEmptyMeasures 230
explicitClefVisibility 679
explicitKeySignatureVisibility 679
extra-offset 596
\eyeglasses 795
Ez_numbers_engraver 42

F

\f 129
\featherDurations 102, 851
\fermata 127, 780, 806
\ff 129
\fff 129
\ffff 129
\fffff 129
figuredBassAlterationDirection 462
figuredBassPlusDirection 462
\figuremode 459, 666
\figures 459, 666
\fill-line 271, 758
\fill-with-pattern 541, 758
\filled-box 274, 775
\finalis 482
\finger 236, 747, 851
fingeringOrientations 237
first-page-number 581
\first-visible 795
\fixed 2, 851
\flageolet 127, 432, 806
\flat 780
\flexa 490
followVoice 359
font-encoding 286
font-interface 236, 286
font-size 232, 236
\fontCaps 747
\fontsize 265, 747
fontSize 232
\footnote 529, 795, 851
footnote-separator-markup 583
Forbid_line_break_engraver 55
forget 36
four-string-banjo 422
\fp 129
\fraction 795
\freeBass 789
\frenchChords 453
\fret-diagram 394, 785
fret-diagram-interface 401
\fret-diagram-terse 396, 786
\fret-diagram-verbose 398, 786
FretBoards 403
\fromproperty 795
\funkHeads 43
\funkHeadsMinor 44

G

\general-align 270, 759
\germanChords 453
glide 239
\glissando 146
\glissandoMap 146
\grace 119, 851
GregorianTranscriptionStaff 197
Grid_line_span_engraver 249
Grid_point_engraver 249
gridInterval 249
grob-interface 811
\grobdescriptions 851
grow-direction 102

H

\halfopen 127, 428, 806
\halign 268, 760
\harmonic 41, 367, 375
\harmonicByFret 375, 851
\harmonicByRatio 375, 851
\harmonicNote 851
\harmonicsOff 367
\harmonicsOn 367, 851
\harp-pedal 787
\haydnturn 127, 805
\hbracket 273, 775
\hcenter-in 761
\header 513
\henzelongfermata 127, 806
\henzeshortfermata 127, 806
\hide 676, 852
\hideKeySignature 434
\hideNotes 241
\hideSplitTiedTabNotes 374
\hideStaffSwitch 359
horizontal-shift 579
Horizontal_bracket_engraver 251
HorizontalBracketText 252
\hspace 269, 761
\huge 232, 268, 748

I

\ictus 482, 807
\iij 485
\IIJ 485
\ij 485
\IJ 485
\improvisationOff 47, 85
\improvisationOn 47, 85
\in 669
\incipit 494, 852
\inclinatum 484, 490
\include 514, 541
indent 219, 579, 617
\inherit-acceptability 852
inner-margin 579
\inStaffSegno 161, 852
\instrumentSwitch 852
\inversion 14, 852
\invertChords 450, 852
\ionian 23
\italianChords 453
\italic 265, 748

J

\justified-lines 278, 801
\justify 272, 763
\justify-field 762
\justify-line 762
\justify-string 763

K

keepAliveInterfaces 213
\keepWithTag 544, 852
\key 22, 44, 852
\kievanOff 491
\kievanOn 491
KievanStaff 490
KievanVoice 490
\killCues 228, 852

L

\label 538, 852
\laissezVibrer 58
\language 852
\languageRestore 852
\languageSaveAndChange 852
\large 232, 268, 748
\larger 265, 268, 748
last-bottom-spacing 577
\layout 513, 583, 628, 639
layout-set-staff-size 585
\left-align 268, 764
\left-brace 796
\left-column 764
left-margin 578
\lheel 127, 806
\line 764
line-width 272, 577, 617
\linea 484, 490
\lineprall 127, 805
\locrian 23
\longa 48, 61
\longfermata 127, 806
\lookup 796
\lower 269, 765
\ltoe 127, 806
ly:minimal-breaking 593
ly:one-line-auto-height-breaking 594
ly:one-line-breaking 594
ly:one-page-breaking 594
ly:optimal-breaking 593
ly:page-turn-breaking 594
\lydian 23
\lyricmode 288, 289, 667
\lyrics 667
\lyricsto 289, 291, 292

M

m 446
magnification->font-size 232, 585
\magnify 265, 748
\magnifyMusic 232, 852
\magnifyStaff 585, 853
magstep 232, 585, 669
maj 446
\major 23
majorSevenSymbol 454
make-dynamic-script 137
make-pango-font-tree 284
\makeClusters 178, 853
\makeDefaultStringTuning 853
\map-markup-commands 801
\marcato 127, 805
\mark 117, 259, 853
Mark_engraver 261
\markalphabet 796
\markLengthOff 75, 260
\markLengthOn 75, 260
\markletter 797
\markup 254, 259, 262, 263, 667
markup-markup-spacing 576
markup-system-spacing 576
\markuplist 262, 278
\markupMap 853
max-systems-per-page 580
\maxima 48, 61
Measure_grouping_engraver 95
measureLength 90, 125
measurePosition 77, 125
\medium 749
\melisma 296
\melismaEnd 296
MensuralStaff 197, 472, 477
MensuralVoice 472, 477
\mergeDifferentlyDottedOff 183
\mergeDifferentlyDottedOn 183
\mergeDifferentlyHeadedOff 183
\mergeDifferentlyHeadedOn 183
\mf 129
middleCPosition 26
\midi 513, 628
midiBalance 566
midiChannelMapping 564
midiChorusLevel 566

midiDrumPitches ... 429
midiExpression ... 566
midiPanPosition ... 566
midiReverbLevel ... 566
min-systems-per-page ... 580
minimum-Y-extent ... 596
minimumFret ... 372, 416
minimumPageTurnLength ... 595
minimumRepeatLengthForPageTurn ... 595
\minor ... 23
minorChordModifier ... 455
mixed ... 362
\mixolydian ... 23
\mm ... 669
\modalInversion ... 17, 853
\modalTranspose ... 16, 853
mode ... 812
modern ... 32
modern-cautionary ... 32
modern-voice ... 32
modern-voice-cautionary ... 32
\mordent ... 127, 805
\mp ... 129
\multi-measure-rest-by-number ... 780
MultiMeasureRestScript ... 66
MultiMeasureRestText ... 66
\musicglyph ... 119, 781
\musicMap ... 853
musicQuotes ... 812

N

\n ... 129
\name ... 644
\natural ... 781
neo-modern ... 34
neo-modern-cautionary ... 34
neo-modern-voice ... 34
neo-modern-voice-cautionary ... 34
\new ... 630
\newSpacingSection ... 614
no-reset ... 35
\noBeam ... 99
\noBreak ... 588
nonstaff-nonstaff-spacing ... 596
nonstaff-relatedstaff-spacing ... 596
nonstaff-unrelatedstaff-spacing ... 596
\noPageBreak ... 592, 853
\noPageTurn ... 595, 853
\normal-size-sub ... 749
\normal-size-super ... 267, 749
\normal-text ... 750
\normalsize ... 232, 268, 750
\note ... 782
\note-by-number ... 781
note-event ... 223
Note_heads_engraver ... 83
Note_name_engraver ... 246
\notemode ... 667
noteNameFunction ... 246
NoteNames ... 246
noteNameSeparator ... 246
\null ... 269, 797
NullVoice ... 318
\number ... 750
\numericTimeSignature ... 70

O

\octaveCheck ... 10, 853
\offset ... 659, 853
\omit ... 675, 853
\on-the-fly ... 527, 797
\once ... 654, 656, 661, 695, 854
\oneVoice ... 179
\open ... 127, 366, 432, 806
\oriscus ... 484, 490
\ottava ... 25, 854
ottavation ... 26
ottavation-numbers ... 25
ottavation-ordinals ... 25
ottavation-simple-ordinals ... 25
ottavationMarkups ... 25
outer-margin ... 579
output-count ... 812
output-def ... 812
output-suffix ... 812
outside-staff-horizontal-padding ... 611
outside-staff-padding ... 611
outside-staff-priority ... 611
\oval ... 775
\overlay ... 765
\override ... 654, 658, 797
\override-lines ... 801
\overrideProperty ... 658, 854
\overrideTimeSignatureSettings ... 70, 854
\overtie ... 750

P

\p ... 129
\pad-around ... 274, 765
\pad-markup ... 274, 766
\pad-to-box ... 274, 766
\pad-x ... 274, 766
page-breaking ... 580
page-breaking-system-system-spacing ... 580
page-count ... 580
\page-link ... 797
page-number-type ... 582
\page-ref ... 538, 798
page-spacing-weight ... 580
\pageBreak ... 592, 854
\pageTurn ... 595, 854
\palmMute ... 854
\palmMuteOn ... 854
\paper ... 513, 572
paper-height ... 574
paper-width ... 577
\parallelMusic ... 194, 854
\parenthesize ... 244, 776, 854
\partCombine ... 189, 318, 855
\partCombineApart ... 190
\partCombineAutomatic ... 190
\partCombineChords ... 190
\partCombineDown ... 855
\partCombineForce ... 855
partCombineListener ... 812

\partCombineSoloI ... 190
\partCombineSoloII ... 190
\partCombineUnisono ... 190
\partCombineUp ... 855
\partial ... 77, 158, 855
\path ... 776
\pattern ... 798
pedalSustainStyle ... 362
percent ... 168
\pes ... 490
\phrasingSlurDashed ... 142
\phrasingSlurDashPattern ... 142, 855
\phrasingSlurDotted ... 142
\phrasingSlurDown ... 142
\phrasingSlurHalfDashed ... 142
\phrasingSlurHalfSolid ... 142
\phrasingSlurNeutral ... 142
\phrasingSlurSolid ... 142
\phrasingSlurUp ... 142
\phrygian ... 23
piano ... 33
piano-cautionary ... 33
PianoStaff ... 354, 357
pipe, symbole ... 116
Pitch_squash_engraver ... 85
\pitchedTrill ... 155, 855
pitchnames ... 812
\pointAndClickOff ... 855
\pointAndClickOn ... 855
\pointAndClickTypes ... 855
\portato ... 127, 805
\postscript ... 275, 777
\pp ... 129
\ppp ... 129
\pppp ... 129
\ppppp ... 129
\prall ... 127, 805
\pralldown ... 127, 805
\prallmordent ... 127, 805
\prallprall ... 127, 805
\prallup ... 127, 805
\preBend ... 377, 855
\preBendHold ... 377, 855
predefinedDiagramTable ... 412
\predefinedFretboardsOff ... 414
\predefinedFretboardsOn ... 414
print-all-headers ... 582
print-first-page-number ... 581
print-page-number ... 581
printAccidentalNames ... 246
printNotesLanguage ... 246
printOctaveNames ... 246
\property-recursive ... 798
\propertyOverride ... 855
\propertyRevert ... 855
\propertySet ... 856
\propertyTweak ... 856
\propertyUnset ... 856
\pt ... 669
\pushToTag ... 548, 856
\put-adjacent ... 766

Q

\quilisma ... 484, 490
quotedCueEventTypes ... 223
quotedEventTypes ... 223
\quoteDuring ... 221, 224, 856

R

r ... 61
R ... 65
ragged-bottom ... 574
ragged-last ... 578, 617
ragged-last-bottom ... 574
ragged-right ... 578, 617
\raise ... 269, 767
\raiseNote ... 450, 856
\reduceChords ... 85, 856
\relative ... 2, 14, 358, 856
\remove ... 637
remove-empty ... 213
remove-first ... 213
remove-grace-property ... 122
remove-layer ... 215
\RemoveAllEmptyStaves ... 212, 860
\RemoveEmptyStaves ... 212, 860
\removeWithTag ... 544, 856
\repeat ... 157, 158, 159
\repeat percent ... 168
\repeat tremolo ... 170
\repeat unfold ... 157
\repeat volta ... 158, 159
repeatCommands ... 166
repeatCountVisibility ... 169
\repeatTie ... 58, 314
\replace ... 751
\resetRelativeOctave ... 5, 856
\responsum ... 485
\rest ... 61, 782
\rest-by-number ... 782
rest-event ... 223
restNumberThreshold ... 231
restrainOpenStrings ... 372
\retrograde ... 15, 856
\reverseturn ... 127, 805
\revert ... 655
\revertTimeSignatureSettings ... 71, 856
\rfz ... 129
rgb-color ... 243
\rheel ... 127, 806
RhythmicStaff ... 197
\right-align ... 268, 767
\right-brace ... 798
\right-column ... 767
right-margin ... 578
\rightHandFinger ... 417, 856
\roman ... 751
\romanStringNumbers ... 366, 370
\rotate ... 768
\rounded-box ... 273, 778
\rtoe ... 127, 806

S

s 63
\sacredHarpHeads 43
\sacredHarpHeadsMinor 44
\sans 751
\scale 778
\scaleDurations 56, 80, 856
\score 509, 513, 783
\score-lines 801
score-markup-spacing 576
score-system-spacing 576
scoreTitleMarkup 523
\segno 127, 806
self-alignment-X 596
\semicirculus 482, 807
\semiflat 784
\semiGermanChords 453
\semisharp 784
\sesquiflat 784
\sesquisharp 785
\set 90, 652, 658
set-global-fonts 284
set-global-staff-size 585
set-octavation 25
\settingsFrom 857
\sf 129
\sff 129
\sfz 129
\shape 691, 857
\sharp 785
\shiftDurations 857
\shiftOff 183
\shiftOn 183
\shiftOnn 183
\shiftOnnn 183
short-indent 219, 579
\shortfermata 127, 806
show-available-fonts 283
showFirstLength 552, 812
\showKeySignature 434
showLastLength 552, 812
\showStaffSwitch 359
\signumcongruentiae 127, 807
\simple 752
\single 532, 661, 857
\skip 63, 857
skipBars 231
skipTypesetting 552
slashChordSeparator 454
\slashed-digit 798
\slashedGrace 119, 857
\slashSeparator 582
\slashturn 127, 805
slur-event 223
\slurDashed 139
\slurDashPattern 140, 857
\slurDotted 139
\slurDown 139
\slurHalfDashed 139
\slurHalfSolid 139
\slurNeutral 139
\slurSolid 139
\slurUp 140
\small 232, 268, 752
\smallCaps 752
\smaller 265, 268, 752
\snappizzicato 127, 806
\sostenutoOff 362
\sostenutoOn 362
\sourcefileline 514
\sourcefilename 514
\southernHarmonyHeads 43
\southernHarmonyHeadsMinor 44
\sp 129
spacing 613
\spacingTweaks 857
Span_stem_engraver 360
\spp 129
\staccatissimo 127, 805
\staccato 127, 805
staff-affinity 596
staff-padding 238
\staff-space 669
staff-staff-spacing 596
Staff.midiInstrument 562, 567
Staff_collecting_engraver 261
Staff_symbol_engraver 212
staffgroup-staff-spacing 596
start-repeat 166
startAcciaccaturaMusic 122
startAppoggiaturaMusic 122
startGraceMusic 122
\startGroup 251
\startStaff 205, 209
\startTrillSpan 154
\stdBass 790
\stdBassIV 791
\stdBassV 792
\stdBassVI 793
Stem 360
stem-spacing-correction 613
\stemDown 245
stemLeftBeamCount 100
\stemNeutral 245
stemRightBeamCount 100
\stemUp 245
\stencil 799
stopAcciaccaturaMusic 122
stopAppoggiaturaMusic 122
stopGraceMusic 122
\stopGroup 251
\stopped 127, 428, 432, 806
\stopStaff 205, 209, 212
\stopTrillSpan 154
\storePredefinedDiagram 407, 412, 857
strictBeatBeaming 95
\string-lines 801
stringNumberOrientations 237
\stringTuning 390, 857
stringTunings 389, 403
strokeFingerOrientations 237, 417
\stropha 484, 490
\strut 799
\styledNoteHeads 857
\sub 267, 753
subdivideBeams 94
suggestAccidentals 128, 477
\super 267, 753
sus 449

\sustainOff 362
\sustainOn 362
system-count 580
system-separator-markup 582
system-system-spacing 576
systems-per-page 580

T

\tabChordRepeats 373, 857
\tabChordRepetition 857
\tabFullNotation 372
\table 802
\table-of-contents 541, 803
TabStaff 197, 371
TabVoice 371
\tag 544, 857
\tagGroup 547, 858
\taor 434
teaching 35
\teeny 232, 268, 753
\tempo 73
\temporary 661, 695, 858
\tenuto 127, 805
text 362, 753
\textLengthOff 66, 68, 257
\textLengthOn 66, 68, 135, 257
\textSpannerDown 257
\textSpannerNeutral 257
\textSpannerUp 257
\thumb 127, 237
\tie 754
TieColumn 60
\tied-lyric 785
\tieDashed 59
\tieDashPattern 59, 858
\tieDotted 59
\tieDown 59
\tieHalfDashed 59
\tieHalfSolid 59
\tieNeutral 59
\tieSolid 59
\tieUp 59
tieWaitForNote 60
\time 69, 90, 858
\times 858
timeSignatureFraction 80
\tiny 232, 268, 754
tocFormatMarkup 541
tocIndentMarkup 541
\tocItem 541, 858
tocItemMarkup 541
\tocItemWithDotsMarkup 540
tocTitleMarkup 541
top-margin 574
top-markup-spacing 577
top-system-spacing 577
toplevel-bookparts 812
toplevel-scores 812
\translate 270, 768
\translate-scaled 270, 768
\transparent 799
\transpose 11, 14, 858
\transposedCueDuring 228, 858
\transposition 28, 221, 858
\treCorde 362
tremolo 170
tremoloFlags 171
\triangle 274, 778
\trill 127, 154, 805
\tripletFeel 568
\tuplet 51, 80, 858
tuplet-slur 52
\tupletDown 51
\tupletNeutral 51
TupletNumber 53
\tupletSpan 52, 858
tupletSpannerDuration 52
\tupletUp 51
\turn 127, 805
\tweak 656, 658, 859
two-sided 578
\type 644
\typewriter 754

U

\unaCorda 362
\underline 265, 754
\undertie 755
\undo 661, 859
unfold 157
\unfolded 859
\unfoldRepeats 563, 859
\unHideNotes 241
\unset 653
\upbow 127, 366, 806
\upmordent 127, 805
\upprall 127, 805
\upright 755

V

\varcoda 127, 806
VaticanaStaff 197
\vcenter 768
\verbatim-file 799
\version 514
\versus 485
VerticalAxisGroup 596
\verylongfermata 127, 806
\veryshortfermata 127, 806
\virga 484, 490
\virgula 482
voice 29, 31
Voice 179
\voiceFour 179
\voiceFourStyle 182
\voiceNeutralStyle 182
\voiceOne 179
\voiceOneStyle 182
\voices 181, 859
\voiceThree 179
\voiceThreeStyle 182
\voiceTwo 179
\voiceTwoStyle 182
\void 569, 859
volta 158, 159, 859

Volta_engraver 163
\vspace 270, 769

W

\walkerHeads 43
\walkerHeadsMinor 44
whichBar 109
\whiteout 799
\whiteTriangleMarkup 454
\with 637, 641
\with-color 242, 800
\with-dimensions 800
\with-dimensions-from 800
\with-link 801
\with-outline 801
\with-url 779
\withMusicProperty 859
\woodwind-diagram 788
\wordwrap 272, 770
\wordwrap-field 769
\wordwrap-internal 803
\wordwrap-lines 278, 803
\wordwrap-string 770
\wordwrap-string-internal 803

X

x11-color 243, 244
X-offset 596
\xNote 40, 859
\xNotesOff 40
\xNotesOn 40

Annexe E Index de LilyPond

En plus des commandes et mots réservés de LilyPond, cet index recense les termes musicaux qui s'y rapportent.

Les entrées d'index en italique renvoient à des endroits (principalement des sections « Voir aussi ») qui contiennent des liens externes vers d'autres ouvrages de documentation de LilyPond comme la Référence des propriétés internes ou le Glossaire musicologique.

Λ

\\ 180

!

! 6
\! 130

"

"|" 116

%

% 510, 514
%{ ... %} 510, 514

,

' 2

(

(..... 139
\(..... 141

)

) 139
\) 141

,

, ... 2

—

- 127, 466, 667
-! 805
-+ 806
-- 805
-. 805
-> 805
-^ 805
-_ 805

.

. 49

/

/ 449
/+ 449

:

: 171

<

< 173
\< 130
<...> 173
<> 174, 355

=

= 10
\= 139, 859

>

> 173
\> 130

?

? 6

[

[..... 99
\[..... 470

]

] 99
\] 470

^

^ 448, 667

_

_ 295, 667

|

| 116

~

~ 57

« accent », articulation 127
« coda », signe 127
« espressivo », articulation 127
« flageolet », harmoniques 127
« grupetto barré », ornement 127
« grupetto Haydn », ornement 127
« grupetto renversé », ornement 127
« grupetto », ornement 127
« marcato », articulation 127
« mordant ascendant », ornement 127
« mordant descendant », ornement 127
« mordant », ornement 127
« ouvert », articulation 127
« pointe », pédalier 127
« portato », articulation 127
« pouce », doigté 127
« poussé », archet 127
« segno », signe 127
« semi-ouvert », articulation 127
« signum congruentiae » ornement 127
« snappizzicato », signe 127
« staccatissimo », articulation 127
« staccato », articulation 127
« stop », articulation 127
« talon », pédalier 127
« tenuto », articulation 127
« tiré », archet 127
« trille », ornement 127
« variante coda », signe 127

Écriture de chants simples 287, 288
à l'italienne, papier 573
échelonnement de musique 56
échelonnement des durées 56
écrire la musique en parallèle 194
égalisation en MIDI, adaptation 560
éolien 23
étiquette 263
étiquette de texte 256
étiquette et silence multimesure 66
étiquette textuelle 263
étiquette, crochet d'analyse 252
événementielle, note de bas de page 530

1

15ma 25
15mb 25

8

8va 25
8vb 25

A

a due 193
a due 189
`\abs-fontsize` 265, 746
absolue, hauteur 1
absolue, octave 1
`\absolute` 848
accent 129
accent 127
`\accent` 127, 805
`\accentus` 482, 807
accentus, grégorien, articulation 482
`\accepts` 644, 645, 646
acciaccatura 123
`\acciaccatura` 119, 848
acciaccature 119
acciaccature, multinotes 124
Accidental 8, 36
accidental 477, 481, 492
accidental-interface 8
accidental-suggestion-interface 36
Accidental_engraver 8, 36, 478
AccidentalCautionary 8
AccidentalPlacement 36
`\accidentalStyle` 29, 848
AccidentalSuggestion 36, 478
`AccidentalSuggestion` 128
accidentel, quart de ton 8
accidentelle automatique 29
accidentelle, altération 29
accolade verticale 199
accord 173
accord arpégé 150
accord chiffré, exception 455
accord et altération 36
accord et couleur 244
accord et liaisons de tenue 58
accord et mode relatif 174
accord et octave relative 5
accord et reprise 457
accord jazz 453
accord nommé et diagramme de fret 405
accord répété, suppression 411
accord vide 122, 174, 355
accord, additions 448
accord, altération d'un degré 448
accord, carrure pour cordes frettées 407
accord, chiffrage 445, 450
accord, diagramme 393, 403
accord, diagramme automatique 414
accord, doigté 237
accord, inversion 449
accord, mode 445
accord, modificateur 446
accord, No Chord 451
accord, nom 445
accord, nom alternatif 453
accord, personnalisation du chiffrage 453
accord, répétition 175, 373
accord, répartition sur plusieurs portées
avec `\autoChange` 359
accord, renversement 450
accord, séparateur 457
accord, spécifier la basse 449

accord, suppressiion des répétitions 411
accord, suppression d'un degré 448
accord, suppression des répétitions 452
accord, vocification 450
accordéon, symbole de registre 363
accordéon, tirettes 363
accordage de banjo 422
accordage non occidental 500
accordage personnalisé 390
accordages prédéfinis et cordes frettées 389
Accordion Registers 363
Accords 450
accords 450
acoustic bass 807
acoustic snare 807
`add-grace-property` 122
`add-stem-support` 238
`add-toc-item!` 540
`\addChordShape` 407, 848
adding white background, to text 799
`\addInstrumentDefinition` 848
`additionalPitchPrefix` 454
additions à un accord 448
`\addlyrics` 289, 291, 292
`\addQuote` 221, 848
`\aeolian` 23
`Affichage d'expressions musicales` 569, 658
affinage (*tweak*) 656
`\afterGrace` 120, 848
`afterGraceFraction` 812
agogo 807
Aiken, note profilée, variante fine 45
Aiken, tête de note 43
`\aikenHeads` 43
`\aikenHeadsMinor` 44
`\aikenThinHeads` 43
`\aikenThinHeadsMinor` 44
ajout d'incipit 494
ajout de texte 256
Ajout et suppression de graveurs 84
ajustement (*tweak*) 656
ajustement des symboles de portée 670
al niente 136
al niente, soufflet 134
`\alias` 644
Align 273, 275
aligné, *ragged* 571
`alignAboveContext` 211, 647
`alignBelowContext` 211, 311, 647
alignement des *markups* 268
Alignement des paroles sur une mélodie 291, 299
alignement du numéro de mesure 115
alignement du texte 268
alignement du texte, commandes 273
alignement et cadence 125
alignement horizontal du texte 268
alignement horizontal, paroles 307
alignement sur un objet 687
alignement vertical du texte 269
alignement vertical, nuance 135
alignment vertical, script textuel 135
alist 810
All layout objects 234, 652, 659, 684, 811
`\allowPageTurn` 595, 848
`\allowVoltaHook` 848
alpha, transparence 243
altérable, objet et propriété 811
altération 6, 477, 492
altération de précaution 6
altération de précaution style *modern voice* 32
altération de précaution, style *modern* 32
altération entre parenthèses 6
altération et accord 36
altération et cadence 78
altération et liaison de tenue 7
altération et musica ficta 477
altération et notes simultanées 36
altération masquée sur note tenue au début du système suivant 7
altération, basse chiffrée, position 462
altération, grégorien 481
altération, style 29
altération, style *dodecaphonic* 35
altération, style *modern* 32
altération, style *modern cautionary* 32
altération, style *no reset* 35
altération, style par défaut 29
altération, trilles avec hauteur explicite 155
altérations multivoix 32, 34
altérations style *choral* 33
altérations style *choral cautionary* 33
altérations style *piano* 33
altérations style *piano cautionary* 33
altérations, style *default* 31
altérations, style *forget* 36
altérations, style *modern* 31
altérations, style *modern-cautionary* 31
altérations, style *neo-modern* 34
altérations, style *teaching* 35
altérations, style *voice* 31
`\alterBroken` 694, 849
alternatative, table de diagrammes d'accord 412
`\alternative` 157, 159
alternative et liaison 58
alternative et paroles 312
alternative et texte 167
alternative, bascule sur une mélodie 323
alternative, reprise 157, 159
alto varC, clef 740
alto, clef 740
Amazing Grace : exemple pour cornemuse 435
ambitus 40, 288
Ambitus 40
ambitus 37
ambitus avec ligne 38
ambitus, placement 39
ambitus, un par voix 37
ambitus, voix multiples 38
ambitus-interface 40
Ambitus_engraver 40
AmbitusAccidental 40
`ambitusAfter` 39, 849
AmbitusLine 40
`AmbitusLine` 38
AmbitusNoteHead 40
amplitude 37
anacrouse 77
anacrouse et reprise 158
anacrouse, cantique 350
anacrusis 78

analyse lexicale ... 811
analyse musicologique ... 251
analyse, crochet, étiquette ... 252
analyseur syntaxique ... 812
ancien silence ... 476
ancien, crochet ... 475
ancien, script ... 807
ancienne altération ... 477
ancienne ligature ... 478
ancienne tête de note ... 475
ancienne, clef ... 18, 473
ancienne, métrique ... 474
anciennes, clefs ... 740
Ancient notation ... 472, 477, 483, 494
`annotate-spacing` ... 624
annotation ... 263
annulation d'un *override* ... 655
Apparition d'une portée supplémentaire ... 157, 159, 648
`\appendToTag` ... 548, 849
`\applyContext` ... 633, 849
`\applyMusic` ... 849
`\applyOutput` ... 849
`\applySwing` ... 568
`\applySwingWithOffset` ... 568
appoggiatura ... 123
`\appoggiatura` ... 119, 849
appoggiature ... 119
arabe, improvisation ... 504
arabe, maqam ... 500
arabe, nom de note ... 501
`\arabicStringNumbers` ... 370
archet, indication ... 127
armure ... 6, 22, 477
armure arabe ... 502
armure non traditionnelle ... 24
armure, grégorien ... 481
armure, visibilité après changement explicite ... 679
arpège ... 150
arpège, portée, enjambement ... 152
arpège, style parenthèse ... 154
arpège, symbole spécial ... 151
arpège, voix, enjambement ... 153
Arpeggio ... 153, 659
arpeggio ... 153
`\arpeggio` ... 150
arpeggio et tenue ... 60
arpeggio, crochet interportée ... 361
`arpeggio-direction` ... 150
`\arpeggioArrowDown` ... 150
`\arpeggioArrowUp` ... 150
`\arpeggioBracket` ... 151
`\arpeggioNormal` ... 150
`\arpeggioParenthesis` ... 151
`\arpeggioParenthesisDashed` ... 151
`\arrow-head` ... 274, 771
`\articulate` ... 567
articulate, script ... 567
articulation grégorienne, *accentus* ... 482
articulation grégorienne, *circulus* ... 482
articulation grégorienne, *episemFinis* ... 482
articulation grégorienne, *episemInitium* ... 482
articulation grégorienne, *ictus* ... 482
articulation grégorienne, *semicirculus* ... 482
articulation, « accent » ... 127
articulation, « espressivo » ... 127
articulation, « marcato » ... 127
articulation, « ouvert » ... 127
articulation, « portato » ... 127
articulation, « semi-ouvert » ... 127
articulation, « staccatissimo » ... 127
articulation, « staccato » ... 127
articulation, « stop » ... 127
articulation, « tenuto » ... 127
articulation, liaison ... 139
articulation, script ... 805
articulation, valeur par défaut, modification ... 128
`articulation-event` ... 223
articulations ... 805
Articulations et nuances ... 136
artificiels, harmoniques ... 367
`\ascendens` ... 484, 490
aspect d'un symbole de demi-bémol ... 502
`\assertBeamQuant` ... 849
`\assertBeamSlope` ... 849
`associatedVoice` ... 289, 291, 323
`\auctum` ... 484, 490
`aug` ... 446
`\augmentum` ... 490
`auto-first-page-number` ... 581
`\auto-footnote` ... 794
Auto_beam_engraver ... 89, 97
`autoBeaming` ... 90, 629
`\autoBeamOff` ... 87, 360
`\autoBeamOn` ... 87
`\autoBreaksOff` ... 588
`\autoBreaksOn` ... 588
`\autoChange` ... 357, 849
`\autoChange` et mode relatif ... 358
AutoChangeMusic ... 359
`\autoLineBreaksOff` ... 588
`\autoLineBreaksOn` ... 588
`automaticBars` ... 682
automatique, altération accidentelle ... 29
automatique, changement de portée ... 357
automatique, diagramme d'accord ... 414
automatique, diagramme de fret ... 414
`\autoPageBreaksOff` ... 592
`\autoPageBreaksOn` ... 592
Autres sources de documentation ... 183, 500, 501, 542, 543, 555, 562, 563, 567, 568, 649, 684
Autres utilisations des retouches ... 355
Axis_group_engraver ... 217, 601

B

bécarre ... 6
bécarre, supression du signe supplémentaire ... 7, 24
bémol ... 6
Bézier, points de contrôle d'une courbe ... 690
Backend ... 649, 652, 656
backslashed digit ... 794
`\backslashed-digit` ... 794
bagpipe ... 434
balance MIDI ... 566
balise ... 544
balises, regroupement ... 547
balloon-interface ... 249
Balloon_engraver ... 249

`Balloon_engraver` ... 248
`\balloonGrobText` ... 248, 849
`\balloonLengthOff` ... 248
`\balloonLengthOn` ... 248
`\balloonText` ... 248, 849
BalloonTextItem ... 249
bandeau avec rupture, modification ... 694
bandeau, modification ... 694
banjo, accordage ... 422
banjo, tablature ... 389, 421
banjo, tablature pour ... 369
`banjo-c-tuning` ... 422
`banjo-modal-tuning` ... 422
`banjo-open-d-tuning` ... 422
`banjo-open-dm-tuning` ... 422
`\bar` ... 102, 109, 849
Bar_engraver ... 453
Bar_number_engraver ... 116
`barCheckSynchronize` ... 116
baritone varC, clef ... 740
baritone varF, clef ... 740
BarLine ... 110
BarNumber ... 116
`BarNumber` ... 110
`\barNumberCheck` ... 117, 849
`barNumberVisibility` ... 110
barré, indication de ... 394
barré, indication de la position ... 418
barre de mesure ... 102
barre de mesure et cadence ... 78
barre de mesure et `ChordNames` ... 456
barre de mesure et paroles ... 308
barre de mesure et soufflet ... 132
barre de mesure invisible ... 103
barre de mesure manuelle ... 103
barre de mesure par défaut, modification du style ... 109
barre de mesure, entre portées ... 201
barre de mesure, impression à intervalle régulier ... 111
barre de mesure, impression du premier numéro ... 111
barre de mesure, personnalisation ... 107
barre de mesure, reprises successives ... 163
barre de mesure, suppression ... 78
barre de mesure, symbole au dessus ... 259
barre de reprise ... 102
barre finale ... 102
barres de mesure, suppression ... 682
Bartók pizzicato ... 368
`bartype` ... 109
bas de page, note de ... 529
bas, direction (_) ... 667
`base-shortest-duration` ... 613
`baseMoment` ... 90, 94
bass ... 807
bass, clef ... 740
basse chiffrée ... 459
basse chiffrée, alignement ... 464
basse chiffrée, altération, position ... 462
basse chiffrée, lignes d'extension ... 462
basse continue ... 459
basse d'un accord ... 449
BassFigure ... 462, 464
BassFigureAlignment ... 462, 464
BassFigureBracket ... 462, 464
BassFigureContinuation ... 462, 464
`\bassFigureExtendersOff` ... 461
`\bassFigureExtendersOn` ... 461
BassFigureLine ... 462, 464
`\bassFigureStaffAlignmentDown` ... 463
`\bassFigureStaffAlignmentNeutral` ... 463
`\bassFigureStaffAlignmentUp` ... 463
battements par minute ... 73
batterie ... 423, 425
batterie, portée ... 197
bayati ... 504
Beam ... 89, 97, 101, 357, 389
`\beam` ... 771
beam-interface ... 89, 97, 101
Beam_engraver ... 89, 101
BeamEvent ... 89, 101
`beamExceptions` ... 90, 849
BeamForbidEvent ... 89, 97
`beatStructure` ... 90, 94
`\bendAfter` ... 145, 849
`\bendHold` ... 377, 849
`\bendStartLevel` ... 377, 850
`binding-offset` ... 579
bisbigliando ... 364
Bison ... 812
blackmensural, clef ... 740
`\blackTriangleMarkup` ... 454
blanc ... 515
blanchiment (*whiteout*) ... 59
`blank-after-score-page-penalty` ... 581
`blank-last-page-penalty` ... 581
`blank-page-penalty` ... 581
bloc de commentaire ... 510, 514
bloc de texte ... 256
BNF ... 812
boîte englobante *(bounding box)* ... 669
bois, clés, liste ... 443
bois, diagramme, modification ... 442
`\bold` ... 265, 746
bongo ... 807
`\book` ... 510, 513
`\bookOutputName` ... 512, 850
`\bookOutputSuffix` ... 512, 850
`\bookpart` ... 511, 513, 592
`bookTitleMarkup` ... 523
`bottom-margin` ... 574
bouché ... 806
bounding box (boîte englobante) ... 669
`\box` ... 273, 746
brève, style alternatif ... 50
brace ... 202
brace, various sizes ... 286
bracket ... 202
`bracket` ... 362
`\bracket` ... 136, 273, 771
`\break` ... 588
`break-align-symbols` ... 687
break-alignment-interface ... 829, 843
`break-visibility` ... 677
`breakable` ... 88
`breakbefore` ... 521
`\breathe` ... 143, 850
Breathing_sign_engraver ... 145
BreathingEvent ... 145

BreathingSign 145, 483
`BreathingSign` 144
breve 50, 63
`\breve` 48, 61
broderie 119
bulle 248

C

césure 144, 481
C, clef 740
cabasa 807
cadence 78, 125
cadence et alignement 125
cadence et altération 78
cadence et barres de mesure 78
cadence et ligatures 78
cadence et numéro de mesure 78
cadence et saut de ligne 79
cadence et saut de page 79
cadenza 79, 125
cadenza 125
`\cadenzaOff` 78
`\cadenzaOn` 78
caesura 145, 482
`\caesura` 482
callback 810
calque 180
calque *(layer)* 676
cantique 340, 346
cantique, mesure incomplète 350
capo 398
`\caps` 747
caractère réservé, impression 264
caractère, taille 265
caractères spéciaux 549
caractères spéciaux en mode markup 264
case 372
`\cavum` 484, 490
`\center-align` 268, 756
`\center-column` 271, 756
centering column of text 756
centrage des nuances entre deux portées de piano 354
centrage du texte sur la page 271
chœur, citation d'instrument 335
chœur, partition de 329
chœur, système 199
`\change` 355
changement de fonte 265
changement de nom d'instrument 219
changement de portée 359
changement de portée automatique 357
changement de portée et collision 356
changement de portée forcé 355
changement de portée manuel 355
changing direction of text column 757
Chansons 289, 328
`\char` 794
`check-consistency` 578
chevrons 173
chiffrage d'accord 450
chiffrage d'accord, exception 455
chiffrage d'accord, personnalisation 453
chiffrage de mesure 69
chiffre de mesure multiple 637
chiffre de n-olet inhabituel 53
chiffre de n-olet, modification 53
chiffre indicateur de mesure 69
chinese cymbal 807
ChoirStaff 202, 204, 330
`choral` 33
choral cautionary, style d'altérations 33
choral, style d'altérations 33
`choral-cautionary` 33
chorale et altérations 33
chorale, citation instrumentale 337
chorale, clef de ténor 18
chorale, partition 332
chord 174, 445, 453
chord, power 420
Chord_name_engraver 453
`chordChanges` 411, 452
chordmode 445
`\chordmode` 14, 405, 666
ChordName 453
`chordNameExceptions` 455
`chordNameLowercaseMinor` 453
ChordNames 217, 453
`ChordNames` 405
`ChordNames` et barre de mesure 456
`chordNameSeparator` 454, 457
`chordNoteNamer` 454
`chordPrefixSpacer` 455
`\chordRepeats` 373, 850
`chordRootNamer` 454
Chords 445, 447, 449, 453, 457, 460, 462, 464, 702
`\chords` 451, 666
chorus MIDI 566
Christian Harmony, tête de note 43
church mode 25
chute 145
`\circle` 273, 772
circling text 772
`\circulus` 482, 807
circulus, grégorien, articulation 482
citation 221
citation et clef 18
citation, fin 228
citation, partition chorale 337
clé 473
clés, bois, liste 443
clés, portées pour instrument à 354
clôture 810
claves 807
clavier, portées pour instrument à 354
claviers, centrage des nuances 354
clef 473, 481, 491
Clef 22, 473
clef 6, 473, 480, 491
`\clef` 18, 850
clef alto 740
clef alto varC 740
clef ancienne 18
clef baritone varC 740
clef baritone varF 740
clef bass 740
clef C 740
clef d'ut 18
clef de citation 18

clef de fa ... 18
clef de sol ... 18
clef de tablature ... 391
clef et transposition ... 18
clef F ... 740
clef french ... 740
clef G ... 740
clef G2 ... 740
clef GG ... 740
clef mezzosoprano ... 740
clef moderntab ... 391
clef percussion ... 740
clef soprano ... 740
clef subbass ... 740
clef tab ... 740
clef tenor ... 740
clef tenor G ... 740
clef tenor varC ... 740
clef transposée, visibilité ... 682
clef treble ... 740
clef varbaritone ... 740
clef varC ... 740
clef, blackmensural ... 740
clef, kievan ... 740
clef, mensural ... 740
clef, musique ancienne ... 740
clef, percussion ... 424
clef, personnalisation des propriétés ... 20
clef, petrucci ... 740
clef, style ... 740
clef, violin ... 740
clef, visibilité après changement explicite ... 679
clef, visibilité de la transposition ... 682
clef-interface ... 22, 473
Clef_engraver ... 22, 473
ClefModifier ... 22, 473
`clip-regions` ... 552
closure ... 810
cluster ... 178
cluster ... 178
Cluster_spanner_engraver ... 178
ClusterSpanner ... 178
ClusterSpannerBeacon ... 178
`\cm` ... 669
coche ... 144
coda ... 119
`\coda` ... 127, 806
coda sur une barre de mesure ... 259
collision ... 183
collision de notes ... 183
collision et changement de portée ... 356
collision et numéro de mesure ... 116
collision verticale ... 611
collision, ignorer ... 178, 188
colonnes de texte ... 271
colonnes, texte ... 262
`color` ... 242
coloration d'objet ... 242
coloration de note ... 242
colorier des voix ... 182
coloring text ... 800
colorisation d'objet ... 242
colorisation de note ... 242
`\column` ... 271, 756
`\column-lines` ... 801
Combinaison de notes en accords ... 174
combinaison de parties ... 189
combinateur de parties ... 189
`\combine` ... 274, 757
comma ... 506
commandes d'alignement du texte ... 273
commentaire ... 510, 514
commentaire textuel ... 263
Common Practice Period ... 10, 501
`common-shortest-duration` ... 613
Completion_heads_engraver ... 84
`Completion_heads_engraver` ... 83
Completion_rest_engraver ... 84
`Completion_rest_engraver` ... 83
complexe, métrique ... 81
composite, métrique ... 80
`\compound-meter` ... 779
`\compoundMeter` ... 81, 850
`\compressEmptyMeasures` ... 230
`\compressMMRests` ... 65, 66, 230, 850
compteur, pourcent, visibilité ... 169
compteur, reprise en pourcent ... 169
`\concat` ... 757
concatenating text ... 757
concert pitch ... 29
condenser les silences ... 69
condition et markup ... 527
conducteur ... 332
Conducteurs et parties ... 543
conga ... 807
`\consists` ... 637, 644
`Construction d'un markup en Scheme` ... 138, 539
contemporain, glissando ... 147
`\context` ... 630, 639
`\context` dans un bloc `\layout` ... 639
`context-spec-music` ... 186
ContextChange ... 357
contexte implicite ... 647
contexte, création ... 630, 644
contexte, définition en MIDI ... 561
contexte, durée de vie ... 634
contexte, maintien actif ... 634
contexte, modification des propriétés par défaut ... 638
contexte, référencement ... 630
Contextes et graveurs ... 181, 628
contextes, ordonnancement ... 646
Contexts ... 597, 599, 602, 628
Contexts and engravers ... 628
contrôle de barre de mesure et reprise ... 158
contrôle des hauteurs ... 10
controlling general text alignment ... 759
`controlpitch` ... 10
Conventions de nommage des objets et propriétés ... 811
copyright ... 550
corde à vide, indication ... 366
corde numérotée ... 370
corde, numéro ... 366
corde, numéro, positionnement ... 237
corde, saut, tablature ... 377
cordes d'orchestre ... 366
cordes frettées et accordages prédéfinis ... 389
cordes frettées, carrure d'accord ... 407
cordes frettées, doigtés main droite ... 417

cordes frettées, harmonique ... 419
cordes frettées, indication de la position et du barré ... 418
cordes frettées, note étouffée ... 419
cordes, écriture pour ... 366
cornemuse ... 434
cornemuse : exemple ... 435
Correction des collisions d'objets ... 356, 357
couche ... 180
couleur ... 242
couleur d'objet ... 676
couleur rgb ... 243
couleur rvb ... 243
couleur, code css ... 242
couleur, note d'un accord ... 244
couleurs, liste ... 716
`countPercentRepeats` ... 169
coup de gratte, indication ... 85
couplet, numéro ... 319
Couplets ... 328
courbes ... 138
cowbell ... 807
`\cr` ... 130
crash cymbal ... 807
creating a table ... 802
creating empty text object ... 797
creating horizontal space, in text ... 761
creating text fraction ... 795
creating vertical space, in text ... 769, 799
`\cresc` ... 131
crescendo ... 136
crescendo ... 130
`crescendo-event` ... 223
`crescendoSpanner` ... 135
`crescendoText` ... 135
`\crescHairpin` ... 131
`\crescTextCresc` ... 131
crochet ... 251, 475
crochet d'arpeggio interportée ... 361
crochet de n-olet, positionnement ... 51
crochet de phrasé ... 251
crochet de regroupement de notes ... 251
crochet de regroupement, imbrication ... 202
crochet de regroupement, portée unique ... 201
crochet de reprise raccourci ... 163
crochet de reprise, plusieurs portées ... 163
crochet interportée ... 361
crochet rectiligne ... 100
crochet vertical ... 199
crochets, note entre ... 244
croix, tête de note ... 40
`cross` ... 40
cross-staff tremolo ... 172
`\crossStaff` ... 360, 850
css, code couleur ... 242
cue-notes ... 338
`\cueClef` ... 224, 850
`\cueClefUnset` ... 224, 850
`\cueDuring` ... 224, 850
`\cueDuringWithClef` ... 224, 850
CueVoice ... 229, 338
`CueVoice` ... 224
cuica ... 807
`currentBarNumber` ... 110, 125
custodes ... 471
`\customTabClef` ... 779
Custos ... 472
custos ... 469, 472
custos ... 471
Custos_engraver ... 471
cymbal, various ... 807

D

décalage ... 659
décalage de note ... 183
décalage de voix ... 183
décalage horizontal dans un *markup* ... 269
décalage vertical dans un *markup* ... 270
décallage, note ... 187
décoration du texte ... 273
défaut, durée ... 49
défaut, hauteur ... 49
défaut, vertical, positionnement (-) ... 667
définition d'une barre de mesure ... 107
définition de sortie ... 628, 812
délimitation, systèmes imbriqués ... 202
désinence et tablature ... 377
D.S al Fine ... 119
`Définition d'une nouvelle commande de liste de markups` ... 279
Déplacement d'objets ... 268, 273
dacalé, grupetto ... 128
dash patterns, slur ... 141
`\dashBang` ... 128
`\dashDash` ... 128
`\dashDot` ... 128
`\dashHat` ... 128
`\dashLarger` ... 128
`\dashPlus` ... 128
`\dashUnderscore` ... 128
`\deadNote` ... 41, 850
`\deadNotesOff` ... 41
`\deadNotesOn` ... 41
`debug-beam-scoring` ... 583
`debug-slur-scoring` ... 583
`debug-tie-scoring` ... 583
`Debugging scoring algorithms` ... 583
`\decr` ... 130
`\decresc` ... 131
decrescendo ... 136
decrescendo ... 130
`decrescendoSpanner` ... 135
`decrescendoText` ... 135
`default` ... 29, 31, 529
`\default` ... 529
default, style d'altérations ... 31
`default-staff-staff-spacing` ... 596
Default_bar_line_engraver ... 82
`\default` ... 117
`defaultBarType` ... 109
`\defaultchild` ... 647
`\defaultTimeSignature` ... 70
`\defineBarLine` ... 107, 850
demi-bémol ... 6, 9, 502
demi-dièse ... 6, 9
`\deminutum` ... 484, 490
`\denies` ... 644, 645, 646
`\descendens` ... 484, 490

dessin des symboles de portée 670
dessus, direction (^) 667
dièse 6
diagramme d'accord pour instrument fretté 393
diagramme d'accord, table alternative 412
diagramme de fret 393
diagramme de fret et transposition 405
diagramme de fret personnalisé 401
diagramme de fret personnalisé, ajout 406
diagramme de fret pour gaucher 396
diagramme de fret prédéfini, définition 409
diagramme de fret, personnalisation 408
diagramme de fret, positionnement 401
diagramme personnalisé de fret 393, 401
`\dim` 131, 446
dimension 669
`\dimHairpin` 131
diminuendo 130
`\dimTextDecr` 131
`\dimTextDecresc` 131
`\dimTextDim` 131
`\dir-column` 757
direction, bas (_) 667
direction, défaut (-) 667
direction, haut (^) 667
`\discant` 789
`\displayLilyMusic` 569, 850
`\displayMusic` 850
`\displayScheme` 851
disponibilité des fontes 283
distance absolue 669
distance entre deux portées de piano 360
distance entre les portées 595
distance extensible 669
distance relative 669
divisi, portées 215
divisio 482
divisio 481
`\divisioMaior` 482
`\divisioMaxima` 482
`\divisioMinima` 482
division de note 83
division de portée 215
division de silence 83
division de voix 331
division et paroles 316
divisiones 481
`dodecaphonic` 35
dodecaphonic, style d'altération 35
dodecaphonic, style d'altérations 35
dodecaphonic, style néo-moderne 35
dodecaphonic, style néomoderne 35
`dodecaphonic-first` 35
`dodecaphonic-no-repeat` 35
doigté 236
doigté d'accord 237
doigté et hampe 238
doigté et silence multimesure 69
doigté main droite, positionnement 417
doigté ou numéro de corde 370
doigté, « pouce » 127
doigté, ajout au diagramme de fret 415
doigté, dans la portée 238
doigté, positionnement 237
doigtés main droite et cordes frettées 417
doigtés, symboles pour vents 432
doigtés, table 433
doigtés,diagrammes pour bois 440
doigte, glissé 239
doit 146
`\dorian` 23
dorien 23
DotColumn 50
Dots 50
`\dotsDown` 49
`\dotsNeutral` 49
`\dotsUp` 49
doublée, legato d'accords 140
double bémol 6
double barre 102
double dièse 6
double flat 8
double pause 61
double point 49
double sharp 8
Double_percent_repeat_engraver 170
`\doubleflat` 780
doublement pointée, note 49
DoublePercentEvent 170
DoublePercentRepeat 170
DoublePercentRepeatCounter 170
DoubleRepeatSlash 170
`\doublesharp` 780
`doubleSlurs` 140
`\downbow` 127, 366, 806
`\downmordent` 127, 805
`\downprall` 127, 805
`\draw-circle` 274, 772
`\draw-dashed-line` 772
`\draw-dotted-line` 773
`\draw-hline` 773
`\draw-line` 274, 773
`\draw-squiggle-line` 774
drawing beam, within text 771
drawing box, with rounded corners 775
drawing box, with rounded corners, around text 778
drawing circle, within text 772
drawing dashed line, within text 772
drawing dotted line, within text 773
drawing ellipse, around text 774
drawing line, across a page 773
drawing line, within text 773
drawing oval, around text 775
drawing path 776
drawing solid box, within text 775
drawing squiggled line, within text 774
drawing triangle, within text 778
`\dropNote` 450, 851
`\drummode` 197, 423, 666
`drumPitchNames` 428
`drumPitchTable` 429
`\drums` 423, 666
DrumStaff 198, 430
`DrumStaff` 197
`drumStyleTable` 428
DrumVoice 430
durée 48
durée d'un silence 61
durée isolée 49

durée par défaut ... 49
durée, ligne de ... 466
durées, échelonnement ... 56
Duration names notes and rests ... 50
`\dwn` ... 502
`\dynamic` ... 136, 747
`dynamic-event` ... 223
Dynamic_performer ... 559, 561, 562, 563
`\dynamicDown` ... 132
DynamicLineSpanner ... 132, 136
`DynamicLineSpanner` ... 132, 135
`\dynamicNeutral` ... 132
Dynamics ... 136
DynamicText ... 136
`\dynamicUp` ... 132

E

easy play, notation, numéro ... 42
easy play, tête de note ... 41
`\easyHeadsOff` ... 41
`\easyHeadsOn` ... 41
Editorial annotations ... 236, 238, 242, 244, 245, 246, 249, 251
effets MIDI ... 566
electric snare ... 807
`\ellipse` ... 774
Emmentaler, fonte ... 718
encadrement du texte ... 273
encapsulated postscript ... 553
enclosing text in box, with rounded corners ... 778
enclosing text within a box ... 746
`\endcr` ... 130
`\enddecr` ... 130
`\endSpanners` ... 675, 851
Engravers and Performers ... 628
enjambement de portée, arpège ... 152
ensemble, musique d' ... 332
entête ... 515
entête de page ... 522
entête, impression ... 582
Episema ... 483
Episema_engraver ... 483
EpisemaEvent ... 483
`\episemFinis` ... 482
episemFinis, grégorien, articulation ... 482
`\episemInitium` ... 482
episemInitium, grégorien, articulation ... 482
EPS, format de sortie ... 553
`\epsfile` ... 275, 775
espace ... 515
espace dans les paroles ... 295
espace, dans les paroles ... 288
`Espacement` ... 613, 614
espacement au sein d'un système ... 595
espacement autour du texte ... 274
espacement des paroles ... 306
espacement entre les portées ... 595
espacement horizontal ... 612
espacement horizontal, affinage ... 696
espacement horizontal, modification ... 615
espacement vertical ... 595
espacement, affichage des valeurs ... 624
espacement, modification en cours de partition ... 614
espacement, ornement ... 123
espressivo ... 131
`\espressivo` ... 127, 131, 805
`\etc` ... 700
`\eventChords` ... 851
exception, chiffrage d'accord ... 455
Exemple concret ... 186, 188, 355
exemple de musique arabe ... 505
`\expandEmptyMeasures` ... 230
expansion de reprise ... 157
`explicitClefVisibility` ... 679
`explicitKeySignatureVisibility` ... 679
exposant ... 267
expression *markup* ... 263
expression MIDI ... 566
Expressions musicales imbriquées ... 209, 212, 647
Expressive marks ... 61, 129, 136, 138, 141, 143, 145, 146, 150, 153, 156, 259, 419
extenseur ... 257, 299
extenseur, mise en forme ... 257
extenseur, modification ... 694
extenseur, nuance, personnalisation ... 258
extension avec rupture, modification ... 694
`extra-offset` ... 596
extraction, fragment ... 552
`\eyeglasses` ... 795
`Ez_numbers_engraver` ... 42

F

`\f` ... 129
F, clef ... 740
fa, clef de ... 18
fall ... 146
fantôme, note ... 244
fantôme, note, percussion ... 430
fausse note, percussion ... 430
`\featherDurations` ... 102, 851
fermé ... 432
`\fermata` ... 127, 780, 806
Ferneyhough, soufflet ... 134
Feta, glyphes ... 718
feuille de chant ... 452
Feuilles de style ... 549
`\ff` ... 129
`\fff` ... 129
`\ffff` ... 129
`\fffff` ... 129
fifth ... 6
figured bass ... 460
FiguredBass ... 217, 462, 464
`figuredBassAlterationDirection` ... 462
`figuredBassPlusDirection` ... 462
`\figuremode` ... 459, 666
`\figures` ... 459, 666
`\fill-line` ... 271, 758
`\fill-with-pattern` ... 541, 758
`\filled-box` ... 274, 775
fin alternative ... 157
fin de citation ... 228
fin de ligne, repère, positionnement ... 260
fin de réplique ... 228
finalis ... 481
`\finalis` ... 482

\finger ... 236, 747, 851
finger-glide-interface ... 241
finger-interface ... 649
Finger_glide_engraver ... 241
FingerGlideSpanner ... 241
Fingering ... 238, 371, 648, 649
fingering-event ... 238, 648
fingering-glide-event ... 241
Fingering_engraver ... 238, 648, 651
FingeringEvent ... 238, 648
FingeringGlideEvent ... 241
fingeringOrientations ... 237
first-page-number ... 581
\first-visible ... 795
\fixed ... 2, 851
flag ... 470, 476
flageolet ... 806
\flageolet ... 127, 432, 806
flageolet, taille ... 432
flat ... 8
\flat ... 780
Flex ... 811
\flexa ... 490
Flexibilité architecturale ... 254, 256
floor tom tom ... 807
fois, première ... 158
followVoice ... 359
Fonctions de rappel ... 685
Fonctions de substitution intermédiaires ... 661, 695
Fonctions musicales ... 698, 699
Fonctions pour markups ... 265
fondamentale ... 446
Font ... 268, 282, 286
font, non-text in markup ... 286
font-encoding ... 286
font-interface ... 236, 649, 797
font-interface ... 236, 286
font-size ... 232, 236
\fontCaps ... 747
FontConfig ... 279
fonte ... 810
fonte Emmentaler ... 718
fonte, changement ... 265
fonte, définition de la taille ... 585
fonte, famille ... 267
fonte, fichiers externes ... 279
fonte, généralités ... 279
fonte, localisation ... 279
fontes disponibles ... 283
fontes musicales, modification ... 554
fontes, choix par défaut ... 284
\fontsize ... 265, 747
fontSize ... 232
\footnote ... 529, 795, 851
footnote, note de bas de page ... 529
footnote-separator-markup ... 583
Footnote_engraver ... 537
FootnoteEvent ... 537
FootnoteItem ... 537
FootnoteSpanner ... 537
Forbid_line_break_engraver ... 84
Forbid_line_break_engraver ... 55
forget ... 36
forget, style d'altérations ... 36
format de repère ... 118
formatage du numéro de mesure ... 113
Formatting text ... 837, 846
forme semai ... 504
four-string-banjo ... 422
\fp ... 129
\fraction ... 795
fragment, extraction ... 552
\freeBass ... 789
French, clef ... 740
\frenchChords ... 453
Frenched score ... 333
Frenched scores ... 212
Frenched staff ... 212, 217
Frenched staves ... 333
fret ... 372
fret et transposition ... 405
fret, ajout de diagramme personnalisé ... 406
fret, ajout de doigté au diagramme ... 415
fret, diagramme ... 393, 403
fret, diagramme automatique ... 414
fret, diagramme avec nom d'accord ... 405
fret, diagramme personnalisé ... 393, 401
fret, diagramme pour gaucher ... 396
fret, mandoline ... 403
fret, positionnement de giagramme ... 401
fret, ukulele ... 403
\fret-diagram ... 394, 785
fret-diagram et *markup* ... 394
fret-diagram-interface ... 401, 403, 408, 414, 417
fret-diagram-interface ... 401
\fret-diagram-terse ... 396, 786
fret-diagram-terse markup ... 396
\fret-diagram-verbose ... 398, 786
fret-diagram-verbose markup ... 398
FretBoards ... 403
Fretted strings ... 371, 389, 393, 403, 414, 417, 418, 419, 420, 421, 422
\fromproperty ... 795
Funk, tête de note ... 43
\funkHeads ... 43
\funkHeadsMinor ... 44
fusion de notes ... 183

G

Gabarits préprogrammés ... 329
galbe des liaisons ... 691
gaucher, diagramme de fret ... 396
\general-align ... 270, 759
\germanChords ... 453
glide ... 239
glissé, doigté ... 239
glissando ... 150
Glissando ... 150, 675
glissando ... 146
\glissando ... 146
glissando contemporain ... 147
glissando d'accords et tablature ... 386
glissando et reprise ... 149, 165
glissando et tablature ... 387
glissando tronqué ... 148
glissando, indication en tablature ... 386
glissando, marque temporelle ... 147
\glissandoMap ... 146

globale, variable ... 812
glyphe ... 810
glyphe musical ... 119
glyphes Feta ... 718
glyphes Parmesan ... 718
gouttière ... 578
grégorien, altération ... 481
grégorien, armure ... 481
grégorien, articulation, *accentus* ... 482
grégorien, articulation, *circulus* ... 482
grégorien, articulation, *episemFinis* ... 482
grégorien, articulation, *episemInitium* ... 482
grégorien, articulation, *ictus* ... 482
grégorien, articulation, *semicirculus* ... 482
grégorien, articulations ... 482
grégorien, clef ... 480
grégorien, ligature de neumes carrés ... 483
grégorien, transcription ... 495
grégorien, transcription moderne ... 346
`\grace` ... 119, 851
grace notes ... 123
Grace_auto_beam_engraver ... 123
Grace_beam_engraver ... 123
Grace_engraver ... 124
Grace_spacing_engraver ... 124
GraceMusic ... 123
grammaire de LilyPond ... 812
grand staff ... 202
GrandStaff ... 36, 202
Graphic ... 275, 690
Graphical Object Interfaces ... 39, 811
graphique, intégration ... 275
graphique, objet ... 810
graphique, tracé d'objet ... 273
graphisme dans la notation ... 274
graphisme, tracé ... 273
gras ... 265
graveur, affectation à un contexte ... 644
gravure manuelle, liaison de tenue ... 60
GregorianTranscriptionStaff ... 198
`GregorianTranscriptionStaff` ... 197
grid-line-interface ... 251
grid-point-interface ... 251
Grid_line_span_engraver ... 251
`Grid_line_span_engraver` ... 249
Grid_point_engraver ... 251
`Grid_point_engraver` ... 249
`gridInterval` ... 249
GridLine ... 251
GridPoint ... 251
grob ... 649, 810
grob, blanchiment ... 59
grob, positionnement vertical ... 667
grob, propriétés ... 654
grob, surimpression ... 59
grob, visibilité ... 675
grob-interface ... 649, 811, 813
`grob-interface` ... 811
`\grobdescriptions` ... 851
grobs, superposition ... 676
groupement de mesures ... 95
groupement de note manuel ... 99
groupement de pulsations ... 95
groupetto ... 805
`grow-direction` ... 102
grupetto décalé ... 128
guidon ... 471
guillemets dans les paroles ... 295
guillemets en mode *markup* ... 264
guillemets, dans les paroles ... 288
guiro ... 807
guitare basse, tablature ... 389
guitare, coup de gratte ... 85
guitare, grille d'accords ... 85
guitare, tête de note ... 40
guitare, tablature pour ... 369

H

Hairpin ... 136, 659, 660
hairpin ... 136
Hal Leonard ... 41
half-open high hat ... 127, 807
`\halfopen` ... 127, 428, 806
`\halign` ... 268, 760
hammer on ... 388
hampe ... 245
hampe barrée ... 122
hampe descendante ... 245
hampe et doigté ... 238
hampe et tablature ... 381
hampe horizontale ... 381
hampe invisible ... 245
hampe montante ... 245
hampe neutre ... 245
hampe, enjambement de portées ... 360
hampe, orientation ... 245
hampe,ligne médiane, direction ... 245
handclap ... 807
Harmonia Sacra, tête de note ... 43
`\harmonic` ... 41, 367, 375
`\harmonicByFret` ... 375, 851
`\harmonicByRatio` ... 375, 851
`\harmonicNote` ... 851
harmonics ... 368
`\harmonicsOff` ... 367
`\harmonicsOn` ... 367, 851
harmonique et cordes frettées ... 419
harmonique et tablature ... 382
harmonique naturel ... 367
harmonique, indication en tablature ... 375
harmonique, tête de note ... 40
harmoniques artificiels ... 367
harmoniques, « flageolet » ... 127
`\harp-pedal` ... 787
harpe ... 364
harpe sacrée, tête de note ... 43
harpe, diagramme de pédales ... 364
harpe, pédale ... 364
hauteur ... 1
hauteur explicite, trille ... 155
hauteur isolée ... 49
hauteur par défaut ... 49
hauteur, nom par défaut ... 6
Hauteurs et armure ... 6, 8, 22, 25, 504
`\haydnturn` ... 127, 805
`\hbracket` ... 273, 775
`\hcenter-in` ... 761
`\header` ... 513

Henze, point d'orgue 127
\henzelongfermata 127, 806
\henzeshortfermata 127, 806
\hide 676, 852
\hideKeySignature 434
\hideNotes 241
\hideSplitTiedTabNotes 374
\hideStaffSwitch 359
high bongo 807
high conga 807
high hat 807
high hat, half open 127
high hat, open 127
high hat, pédale 127
high timbale 807
high tom tom 807
horizontal, alignement des paroles 307
horizontal, espacement 612
horizontal-bracket-interface 253
horizontal-bracket-text-interface 253
horizontal-shift 579
Horizontal_bracket_engraver 253
Horizontal_bracket_engraver 251
HorizontalBracket 253
HorizontalBracketText 253
HorizontalBracketText 252
horizontale, hampe 381
horizontally centering text 756
\hspace 269, 761
hufnagel 468, 469
\huge 232, 268, 748
hymne 340, 346

I

\ictus 482, 807
ictus, grégorien, articulation 482
identificateurs 514
\iij 485
\IIJ 485
\ij 485
\IJ 485
image, intégration 275
imbrication de reprise 165
imbrication de systèmes 202
implicite, contexte 647
importing stencil, into text 799
impression de caractère réservé 264
impression de caractères spéciaux 264
impression, ordre 676
improvisation 47
improvisation arabe 504
\improvisationOff 47, 85
\improvisationOn 47, 85
\in 669
inaltérable, objet 811
inaltérable, propriété 811
incipit 494
\incipit 494, 852
\inclinatum 484, 490
\include 514, 541
inclusion de fichier 541
incomplète, mesure 77
indépendant, texte 262
indent 219, 579, 617
indentation 579
indication d'archet 366
indication d'octave relative 2
indication de corde à vide 366
indication de nuance, personnalisation 136
indication du barré 394
indication textuelle 259
indication, trémolo 171
indice 267
individuelle, partition 217
info-bulle 248
\inherit-acceptability 852
inlining an Encapsulated PostScript image 775
inner-margin 579
inserting music, into text 783
inserting PostScript directly, into text 777
inserting URL link, into text 779
\inStaffSegno 161, 852
Instanciation explicite des voix 181, 182
instrument à vent 431
instrument MIDI, égalisation 560
Instrument Specific Markup 365, 403
instrument transpositeur 12
instrument, centrage du nom 218
instrument, changement de nom 219
instrument, citation 335
instrument, nom 217
instrument, nom abrégé 217
instrument, nom complexe 218
instrument, nom d' 562
instrument, nom, autres contextes 219
instrument, script spécifique 806
instrument-specific-markup-interface 444, 797
InstrumentName 221
\instrumentSwitch 852
intégration d'objet graphique 273
intégration de graphique 275
inter-portée, ligature 355
inter-portée, notes 355
interface 811
interfaces de rendu 649
Interfaces pour programmeurs 684
interportée, hampe 360
interportée, trémolo 171
interportées, barre de mesure 107
interval 6
intervalle de comma 506
intervalle medium 501
inversion 14
\inversion 14, 852
\invertChords 450, 852
invisible, barre de mesure 103
invisible, note 241
\ionian 23
ionien 23
iraq 504
isolée, durée 49
isolée, hauteur 49
\italianChords 453
\italic 265, 748
italique 265
item-interface 649

J

J'entends des Voix ... 183, 426, 428
jazz, accord ... 453
justifié, texte ... 272
`\justified-lines` ... 278, 801
`\justify` ... 272, 763
`\justify-field` ... 762
`\justify-line` ... 762
`\justify-string` ... 763
justifying lines of text ... 801
justifying text ... 763

K

Keep_alive_together_engraver ... 217
`keepAliveInterfaces` ... 213
`\keepWithTag` ... 544, 852
`\key` ... 22, 44, 852
key signature ... 477, 481
key-signature-interface ... 25
Key_engraver ... 25
Key_performer ... 25
Keyboards ... 355, 357, 359, 360, 362, 363
Keyboards, Claviers ... 364
KeyCancellation ... 25
KeyChangeEvent ... 25
KeySignature ... 25, 477, 481, 504
kiévien ... 490
kiévienne, altération ... 492
kiévienne, clef ... 491
kiévienne, ligature ... 492
kiéviennne, tête de note ... 491
Kiev ... 490
Kievan ... 490
kievan notation ... 491, 492
Kievan, clef ... 473
kievan, clef ... 740
`\kievanOff` ... 491
`\kievanOn` ... 491
`KievanStaff` ... 490
`KievanVoice` ... 490
`\killCues` ... 228, 852
kirchenpause ... 231
kurd ... 504

L

La partition est une (unique) expression musicale composée ... 510
La propriété outside-staff-priority ... 611
`\label` ... 538, 852
laissez vibrer ... 61
laissez vibrer ... 58
`\laissezVibrer` ... 58
LaissezVibrerTie ... 61
LaissezVibrerTieColumn ... 61
`landscape`, papier ... 573
`\language` ... 852
`\languageRestore` ... 852
`\languageSaveAndChange` ... 852
langue, nom de note ... 8
`\large` ... 232, 268, 748
`\larger` ... 265, 268, 748
`last-bottom-spacing` ... 577
layer (calque) ... 676
`\layout` ... 513, 583, 628, 639
`layout-set-staff-size` ... 585
lead sheet ... 452
ledger line ... 208
ledger-line-spanner-interface ... 41
Ledger_line_engraver ... 41
LedgerLineSpanner ... 41
`\left-align` ... 268, 764
left-aligning text ... 764
`\left-brace` ... 796
`\left-column` ... 764
`left-margin` ... 578
legato ... 139
Les expressions musicales en clair ... 510
Les voix contiennent la musique ... 182, 188
levée ... 77
levée dans une reprise ... 158
lexer ... 811
`\lheel` ... 127, 806
liaison d'articulation ... 139
liaison de n-olet ... 52
liaison de phrasé ... 141
liaison de prolongation ... 57
liaison de prolongation et reprise avec alternative ... 58
liaison de prolongation, apparence ... 59
liaison de prolongation, pointillés ... 59
liaison de prolongation, positionnement ... 59
liaison de prolongation, tirets ... 59
liaison de tenue ... 57
liaison de tenue et accord ... 58
liaison de tenue et altération ... 7
liaison de tenue et répétition ... 58
liaison doublée, pour legato d'accords ... 140
liaison et reprise ... 165
liaison et texte ... 140
liaison, au-dessous des notes ... 139
liaison, au-dessus des notes ... 139
liaison, dans les paroles ... 295
liaison, densité des tirets ... 140, 142
liaison, inclusion de texte ... 140
liaison, laissez vibrer ... 58
liaison, modification ... 690
liaison, orientation manuelle ... 139
liaison, style de trait ... 139, 142
liaison, tirets ... 139, 142
liaison, tirets et trait continu ... 139, 142
liaison, trait plein ... 139, 142
liaison, trait pointillé ... 139, 142
liaisons d'articulation multiples ... 139
liaisons d'articulation simultanées ... 139
liaisons de phrasé multiples ... 142
liaisons de phrasé simultanées ... 142
lien de croche ... 87
ligature ... 469, 471, 479, 490, 493
ligature ... 470, 492
ligature ancienne ... 478
ligature blanche ... 478
ligature coudée ... 88
ligature coudée, personnalisation ... 88
ligature en fin de partition ... 96
ligature en fin de voix polyphonique ... 96
ligature en soufflet ... 102
ligature et cadence ... 78

ligature et mélisme ... 88
ligature et métrique ... 70
ligature et musique polymétrique ... 80
ligature et neumes ... 483
ligature et paroles ... 90
ligature et saut de ligne ... 88
ligature inter-portée ... 355
ligature manuelle ... 87, 99
ligature manuelle et note d'ornement ... 99
ligature manuelle, orientation ... 99
ligature, définition de règles ... 87
ligature, in text ... 757
ligature, indication des subdivisions ... 95
ligature, n-olet, saut de ligne ... 55
ligature, `\partCombine` et `\autoBeamOff` ... 89
ligature, personnalisation ... 87
ligature, subdivision ... 94
ligne de commentaire ... 510, 514
ligne de prolongation, nuance, masquage ... 135
ligne médiane, hampe, direction ... 245
ligne supplémentaire ... 205
ligne, indication de durée ... 466
ligne, longueur ... 617
ligne, passer à la suivante ... 588
lignes ... 146
lignes de portée, épaisseur ... 205
lignes de portée, nombre ... 205
`LilyPond grammar` ... 812
line ... 208
`\line` ... 764
line-spanner-interface ... 675
`line-width` ... 272, 577, 617
`\linea` ... 484, 490
LineBreakEvent ... 592
`\lineprall` ... 127, 805
liste associative ... 810
liste associative, modification ... 664
liste des couleurs ... 716
liturgie et musique ... 340
`\locrian` ... 23
locrien ... 23
longa ... 50, 63
`\longa` ... 48, 61
`\longfermata` ... 127, 806
longueur d'un silence multimesure ... 66
longueur de note ... 48
Longueur et épaisseur des objets ... 212, 669
longueur minimale, soufflet ... 133
`\lookup` ... 796
losange, tête de note ... 40, 367
low bongo ... 807
low conga ... 807
low timbale ... 807
low tom tom ... 807
`\lower` ... 269, 765
lowering text ... 765
`\ltoe` ... 127, 806
luth, personnalisation ... 423
luth, tablature ... 422
`ly:add-context-mod` ... 863
`ly:add-file-name-alist` ... 863
`ly:add-interface` ... 863
`ly:add-listener` ... 863
`ly:add-option` ... 863
`ly:all-grob-interfaces` ... 863
`ly:all-options` ... 863
`ly:all-stencil-expressions` ... 863
`ly:angle` ... 863
`ly:assoc-get` ... 863
`ly:axis-group-interface::add-element` ... 863
`ly:basic-progress` ... 863
`ly:book-add-bookpart!` ... 863
`ly:book-add-score!` ... 864
`ly:book-book-parts` ... 864
`ly:book-header` ... 864
`ly:book-paper` ... 864
`ly:book-process` ... 864
`ly:book-process-to-systems` ... 864
`ly:book-scores` ... 864
`ly:book-set-header!` ... 864
`ly:book?` ... 863
`ly:bp` ... 864
`ly:bracket` ... 864
`ly:broadcast` ... 864
`ly:camel-case->lisp-identifier` ... 864
`ly:chain-assoc-get` ... 864
`ly:check-expected-warnings` ... 864
`ly:cm` ... 864
`ly:command-line-code` ... 864
`ly:command-line-options` ... 864
`ly:connect-dispatchers` ... 865
`ly:context-current-moment` ... 865
`ly:context-def-lookup` ... 865
`ly:context-def-modify` ... 865
`ly:context-def?` ... 865
`ly:context-event-source` ... 865
`ly:context-events-below` ... 865
`ly:context-find` ... 865
`ly:context-grob-definition` ... 865
`ly:context-id` ... 865
`ly:context-matched-pop-property` ... 865
`ly:context-mod-apply!` ... 865
`ly:context-mod?` ... 865
`ly:context-name` ... 865
`ly:context-now` ... 865
`ly:context-parent` ... 865
`ly:context-property` ... 865
`ly:context-property-where-defined` ... 866
`ly:context-pushpop-property` ... 866
`ly:context-set-property!` ... 866
`ly:context-unset-property` ... 866
`ly:context?` ... 865
`ly:debug` ... 866
`ly:default-scale` ... 866
`ly:dimension?` ... 866
`ly:dir?` ... 866
`ly:directed` ... 866
`ly:disconnect-dispatchers` ... 866
`ly:dispatcher?` ... 866
`ly:duration->string` ... 866
`ly:duration-dot-count` ... 866
`ly:duration-factor` ... 866
`ly:duration-length` ... 866
`ly:duration-log` ... 866
`ly:duration-scale` ... 867
`ly:duration<?` ... 866
`ly:duration?` ... 866
`ly:effective-prefix` ... 867
`ly:encode-string-for-pdf` ... 867
`ly:engraver-announce-end-grob` ... 867

ly:engraver-make-grob ... 867
ly:error ... 867
ly:event-deep-copy ... 867
ly:event-property ... 867
ly:event-set-property! ... 867
ly:event? ... 867
ly:expect-warning ... 867
ly:extract-subfont-from-collection ... 867
ly:find-file ... 867
ly:font-config-add-directory ... 867
ly:font-config-add-font ... 867
ly:font-config-display-fonts ... 867
ly:font-config-get-font-file ... 868
ly:font-design-size ... 868
ly:font-file-name ... 868
ly:font-get-glyph ... 868
ly:font-glyph-name-to-charcode ... 868
ly:font-glyph-name-to-index ... 868
ly:font-index-to-charcode ... 868
ly:font-magnification ... 868
ly:font-metric? ... 868
ly:font-name ... 868
ly:font-sub-fonts ... 868
ly:format ... 868
ly:format-output ... 868
ly:generic-bound-extent ... 869
ly:get-all-function-documentation ... 869
ly:get-all-translators ... 869
ly:get-cff-offset ... 869
ly:get-context-mods ... 869
ly:get-font-format ... 869
ly:get-option ... 869
ly:get-spacing-spec ... 869
ly:gettext ... 869
ly:grob-alist-chain ... 869
ly:grob-array->list ... 869
ly:grob-array-length ... 869
ly:grob-array-ref ... 869
ly:grob-array? ... 869
ly:grob-basic-properties ... 869
ly:grob-chain-callback ... 869
ly:grob-common-refpoint ... 870
ly:grob-common-refpoint-of-array ... 870
ly:grob-default-font ... 870
ly:grob-extent ... 870
ly:grob-get-vertical-axis-group-index ... 870
ly:grob-interfaces ... 870
ly:grob-layout ... 870
ly:grob-object ... 870
ly:grob-original ... 870
ly:grob-parent ... 870
ly:grob-pq<? ... 870
ly:grob-properties? ... 870
ly:grob-property ... 870
ly:grob-property-data ... 870
ly:grob-pure-height ... 870
ly:grob-pure-property ... 870
ly:grob-relative-coordinate ... 870
ly:grob-robust-relative-extent ... 870
ly:grob-script-priority-less ... 871
ly:grob-set-nested-property! ... 871
ly:grob-set-object! ... 871
ly:grob-set-parent! ... 871
ly:grob-set-property! ... 871
ly:grob-spanned-rank-interval ... 871
ly:grob-staff-position ... 871
ly:grob-suicide! ... 871
ly:grob-system ... 871
ly:grob-translate-axis! ... 871
ly:grob-vertical<? ... 871
ly:grob? ... 869
ly:gulp-file ... 871
ly:gulp-file-utf8 ... 871
ly:has-glyph-names? ... 871
ly:hash-table-keys ... 871
ly:inch ... 871
ly:input-both-locations ... 871
ly:input-file-line-char-column ... 872
ly:input-location? ... 872
ly:input-message ... 872
ly:input-warning ... 872
ly:interpret-music-expression ... 872
ly:intlog2 ... 872
ly:item-break-dir ... 872
ly:item-get-column ... 872
ly:item? ... 872
ly:iterator? ... 872
ly:length ... 872
ly:lily-lexer? ... 872
ly:lily-parser? ... 872
ly:line-interface::line ... 872
ly:listened-event-class? ... 872
ly:listened-event-types ... 872
ly:listener? ... 872
ly:make-book ... 872
ly:make-book-part ... 873
ly:make-context-mod ... 873
ly:make-dispatcher ... 873
ly:make-duration ... 873
ly:make-global-context ... 873
ly:make-global-translator ... 873
ly:make-grob-properties ... 873
ly:make-moment ... 873
ly:make-music ... 873
ly:make-music-function ... 873
ly:make-music-relative! ... 873
ly:make-output-def ... 873
ly:make-page-label-marker ... 874
ly:make-page-permission-marker ... 874
ly:make-pango-description-string ... 874
ly:make-paper-outputter ... 874
ly:make-pitch ... 874
ly:make-prob ... 874
ly:make-rotation ... 874
ly:make-scale ... 874
ly:make-scaling ... 874
ly:make-score ... 874
ly:make-spring ... 874
ly:make-stencil ... 874
ly:make-stream-event ... 874
ly:make-transform ... 875
ly:make-translation ... 875
ly:make-unpure-pure-container ... 875
ly:message ... 875
ly:minimal-breaking ... 875
ly:minimal-breaking ... 593
ly:mm ... 875
ly:module->alist ... 875
ly:module-copy ... 875
ly:modules-lookup ... 875

ly:moment-add ... 875
ly:moment-div ... 875
ly:moment-grace ... 875
ly:moment-grace-denominator ... 875
ly:moment-grace-numerator ... 875
ly:moment-main ... 876
ly:moment-main-denominator ... 876
ly:moment-main-numerator ... 876
ly:moment-mod ... 876
ly:moment-mul ... 876
ly:moment-sub ... 876
ly:moment<? ... 875
ly:moment? ... 875
ly:music-compress ... 876
ly:music-deep-copy ... 876
ly:music-duration-compress ... 876
ly:music-duration-length ... 876
ly:music-function-extract ... 876
ly:music-function-signature ... 876
ly:music-function? ... 876
ly:music-length ... 876
ly:music-list? ... 876
ly:music-mutable-properties ... 876
ly:music-output? ... 876
ly:music-property ... 877
ly:music-set-property! ... 877
ly:music-start ... 877
ly:music-transpose ... 877
ly:music? ... 876
ly:note-column-accidentals ... 877
ly:note-column-dot-column ... 877
ly:note-head::stem-attachment ... 877
ly:number->string ... 877
ly:one-line-auto-height-breaking ... 877
ly:one-line-auto-height-breaking ... 594
ly:one-line-breaking ... 877
ly:one-line-breaking ... 594
ly:one-page-breaking ... 877
ly:one-page-breaking ... 594
ly:optimal-breaking ... 877
ly:optimal-breaking ... 593
ly:option-usage ... 877
ly:otf->cff ... 877
ly:otf-font-glyph-info ... 877
ly:otf-font-table-data ... 878
ly:otf-font? ... 877
ly:otf-glyph-count ... 878
ly:otf-glyph-list ... 878
ly:output-def-clone ... 878
ly:output-def-lookup ... 878
ly:output-def-parent ... 878
ly:output-def-scope ... 878
ly:output-def-set-variable! ... 878
ly:output-def? ... 878
ly:output-description ... 878
ly:output-find-context-def ... 878
ly:output-formats ... 878
ly:outputter-close ... 878
ly:outputter-dump-stencil ... 878
ly:outputter-dump-string ... 878
ly:outputter-output-scheme ... 878
ly:outputter-port ... 878
ly:page-marker? ... 878
ly:page-turn-breaking ... 878
ly:page-turn-breaking ... 594
ly:pango-font-physical-fonts ... 879
ly:pango-font? ... 879
ly:paper-book-header ... 879
ly:paper-book-pages ... 879
ly:paper-book-paper ... 879
ly:paper-book-performances ... 879
ly:paper-book-scopes ... 879
ly:paper-book-systems ... 879
ly:paper-book? ... 879
ly:paper-column::break-align-width ... 879
ly:paper-column::print ... 879
ly:paper-fonts ... 879
ly:paper-get-font ... 879
ly:paper-get-number ... 879
ly:paper-outputscale ... 879
ly:paper-score-paper-systems ... 879
ly:paper-system-minimum-distance ... 879
ly:paper-system? ... 879
ly:parse-file ... 880
ly:parse-init ... 880
ly:parse-string-expression ... 880
ly:parsed-undead-list! ... 880
ly:parser-clear-error ... 880
ly:parser-clone ... 880
ly:parser-define! ... 880
ly:parser-error ... 880
ly:parser-has-error? ... 880
ly:parser-include-string ... 880
ly:parser-lookup ... 880
ly:parser-output-name ... 880
ly:parser-parse-string ... 880
ly:parser-set-note-names ... 880
ly:performance-headers ... 880
ly:performance-write ... 880
ly:pitch-alteration ... 881
ly:pitch-diff ... 881
ly:pitch-negate ... 881
ly:pitch-notename ... 881
ly:pitch-octave ... 881
ly:pitch-quartertones ... 881
ly:pitch-semitones ... 881
ly:pitch-steps ... 881
ly:pitch-tones ... 881
ly:pitch-transpose ... 881
ly:pitch<? ... 881
ly:pitch? ... 880
ly:pointer-group-interface::add-grob ... 881
ly:position-on-line? ... 881
ly:prob-immutable-properties ... 881
ly:prob-mutable-properties ... 881
ly:prob-property ... 881
ly:prob-property? ... 881
ly:prob-set-property! ... 881
ly:prob-type? ... 882
ly:prob? ... 881
ly:programming-error ... 882
ly:progress ... 882
ly:property-lookup-stats ... 882
ly:protects ... 882
ly:pt ... 882
ly:pure-call ... 882
ly:randomize-rand-seed ... 882
ly:register-stencil-expression ... 882
ly:register-translator ... 882
ly:relative-group-extent ... 882

ly:rename-file ... 882
ly:reset-all-fonts ... 882
ly:round-filled-box ... 882
ly:round-filled-polygon ... 882
ly:run-translator ... 883
ly:score-add-output-def! ... 883
ly:score-embedded-format ... 883
ly:score-error? ... 883
ly:score-header ... 883
ly:score-music ... 883
ly:score-output-defs ... 883
ly:score-set-header! ... 883
ly:score? ... 883
ly:separation-item::print ... 883
ly:set-default-scale ... 883
ly:set-grob-creation-callback ... 883
ly:set-grob-modification-callback ... 883
ly:set-middle-C! ... 884
ly:set-option ... 884
ly:set-origin! ... 884
ly:set-property-cache-callback ... 884
ly:skyline-empty? ... 884
ly:skyline-pair? ... 884
ly:skyline? ... 884
ly:smob-protects ... 884
ly:solve-spring-rod-problem ... 884
ly:source-file? ... 884
ly:source-files ... 884
ly:spanner-bound ... 884
ly:spanner-broken-into ... 885
ly:spanner-set-bound! ... 885
ly:spanner? ... 884
ly:spawn ... 885
ly:spring-set-inverse-compress-strength! ... 885
ly:spring-set-inverse-stretch-strength! ... 885
ly:spring? ... 885
ly:staff-symbol-line-thickness ... 885
ly:staff-symbol-staff-radius ... 885
ly:staff-symbol-staff-space ... 885
ly:stderr-redirect ... 885
ly:stencil-add ... 885
ly:stencil-aligned-to ... 885
ly:stencil-combine-at-edge ... 885
ly:stencil-empty? ... 885
ly:stencil-expr ... 885
ly:stencil-extent ... 885
ly:stencil-in-color ... 886
ly:stencil-outline ... 886
ly:stencil-rotate ... 886
ly:stencil-rotate-absolute ... 886
ly:stencil-scale ... 886
ly:stencil-stack ... 886
ly:stencil-translate ... 886
ly:stencil-translate-axis ... 886
ly:stencil? ... 885
ly:stream-event? ... 886
ly:string-percent-encode ... 886
ly:string-substitute ... 886
ly:system-font-load ... 886
ly:text-interface::interpret-markup ... 887
ly:transform->list ... 887
ly:transform? ... 887
ly:translate-cpp-warning-scheme ... 887
ly:translator-context ... 887
ly:translator-description ... 887
ly:translator-group? ... 887
ly:translator-name ... 887
ly:translator? ... 887
ly:transpose-key-alist ... 887
ly:ttf->pfa ... 887
ly:ttf-ps-name ... 887
ly:type1->pfa ... 887
ly:unit ... 887
ly:unpure-call ... 887
ly:unpure-pure-container-pure-part ... 888
ly:unpure-pure-container-unpure-part ... 888
ly:unpure-pure-container? ... 887
ly:usage ... 888
ly:verbose-output? ... 888
ly:version ... 888
ly:warning ... 888
ly:warning-located ... 888
ly:wide-char->utf-8 ... 888
\lydian ... 23
lydien ... 23
LyricCombineMusic ... 295, 302
LyricExtender ... 299
LyricHyphen ... 299
\lyricmode ... 288, 289, 667
Lyrics ... 217, 291, 295, 302, 330, 819
\lyrics ... 667
\lyricsto ... 289, 291, 292
LyricText ... 289, 327, 340

M

m ... 446
mélisme ... 296, 299
mélisme et ligature ... 88
mélodie alternative ... 323
mélodie d'une portée à une autre ... 359
mélodie, affichage du rythme seul ... 84
métadonnées MIDI ... 528
métadonnées PDF ... 528
métrique ... 69
métrique arabe ... 504
métrique composite ... 80, 81
métrique double ... 80
métrique en cours de mesure ... 77
métrique et ligature ... 70
métrique par défaut ... 70
métrique polymétrique ... 80
métrique, numérateur seulement ... 72
métrique, retour au propriétés par défaut ... 71
métrique, style ... 70, 474
métrique, visibilité ... 69
métronome, indication ... 73
Méthodes de retouche ... 55, 656, 658
Métriques anciennes ... 70
magnification->font-size ... 232, 585
\magnify ... 265, 748
magnifying text ... 748
\magnifyMusic ... 232, 852
\magnifyStaff ... 585, 853
magstep ... 232, 585, 669
main droite, doigté, positionnement ... 417
main droite, doigtés pour cordes frettées ... 417
maj ... 446
majeur ... 23

\major ... 23
majorSevenSymbol ... 454
makam ... 506
makam ... 500, 506
makam, exemple ... 507
makamlar ... 501, 506
makamlar ... 500, 506
make-dynamic-script ... 137
make-dynamic-script ... 137
make-pango-font-tree ... 284
\makeClusters ... 178, 853
\makeDefaultStringTuning ... 853
mandoline, tablature ... 389
manuel, saut de page ... 592
manuelle, barre de mesure ... 103
Manuels ... 1
\map-markup-commands ... 801
maqam ... 504
maqam ... 500, 501, 506
maracas ... 807
marcato ... 127
\marcato ... 127, 805
marge, texte qui dépasse ... 257
\mark ... 117, 259, 853
Mark_engraver ... 119, 262
Mark_engraver ... 261
\markalphabet ... 796
MarkEvent ... 119, 262
\markLengthOff ... 75, 260
\markLengthOn ... 75, 260
\markletter ... 797
\markup ... 254, 259, 262, 263, 667
markup conditionnel ... 527
markup et décalage horizontal ... 269
markup et décalage vertical ... 270
markup et fret-diagram ... 394
markup et paroles ... 289
markup et silence multimesure ... 68
markup multiligne ... 271
markup multipage ... 278
markup text, line width ... 272
markup, centrage sur la page ... 271
markup, commandes d'alignement du texte ... 273
markup, encadrement du texte ... 273
markup, expression ... 263
markup, inclusion de musique ... 275
markup, inclusion de partition ... 277
markup, objet ... 254
markup, ornementation du texte ... 273
markup, rembourrage du texte ... 274
markup, rotation ... 683
markup, syntaxe ... 263
markup, tempo ... 76
markup, texte au kilomètre ... 272
markup, texte justifié ... 272
markup, texte multipage ... 278
markup-markup-spacing ... 576
markup-system-spacing ... 576
\markuplist ... 262, 278
\markupMap ... 853
markups, alignement ... 268
marque temporelle, glissando ... 147
masquée, note ... 241
masquage de hampe ... 245
masquage de portée ... 212
masquage, nuance, ligne de prolongation ... 135
matériel d'orchestre ... 217
max-systems-per-page ... 580
maxima ... 50, 63
\maxima ... 48, 61
Measure_grouping_engraver ... 95
measureLength ... 90, 125
measurePosition ... 77, 125
Medicaea, Editio ... 468, 469
\medium ... 749
melisma ... 296, 299
\melisma ... 296
\melismaEnd ... 296
mensural ... 468, 469
mensural notation ... 469, 470, 473, 474, 475, 476, 477
mensural, clef ... 740
mensural, transcription ... 498
mensurale noire, clef ... 473
mensurale, clef ... 473
MensuralStaff ... 198
MensuralStaff ... 197, 472, 477
MensuralVoice ... 472, 477
mensuration, signe ... 474
Mensurstriche ... 494
\mergeDifferentlyDottedOff ... 183
\mergeDifferentlyDottedOn ... 183
\mergeDifferentlyHeadedOff ... 183
\mergeDifferentlyHeadedOn ... 183
merging text ... 757, 765
mesure à compter ... 65
mesure de silence ... 61
mesure entière de silence ... 65
mesure incomplète ... 77
mesure tronquée et numéro de mesure ... 112
mesure, numéro de ... 125
mesure, numérotation ... 110
mesure, numérotation et reprise ... 164
mesure, répétition de ... 168
mesure, subdivision ... 95
mesure, vérification des limites ... 116
meter ... 82
metronome ... 77
metronome mark ... 77
MetronomeMark ... 77
metronomic indication ... 77
mezzosoprano, clef ... 740
\mf ... 129
micro-intervalle, tablature ... 392
microtonalité ... 9
mid tom tom ... 807
middleCPosition ... 26
\midi ... 513, 628
MIDI ... 559, 562
MIDI ... 556
MIDI et reprise ... 563
MIDI et transposition ... 28
MIDI metadata ... 528
MIDI, égalisation par défaut, adaptation ... 560
MIDI, éléments non pris en compte ... 557
MIDI, éléments pris en compte ... 556
MIDI, canaux ... 564
MIDI, définition de contexte ... 561
MIDI, gestion des nuances ... 558
MIDI, indications de nuance ... 558
MIDI, instrument ... 562, 567

MIDI, le bloc ... 557
MIDI, nuance personnalisée ... 558
MIDI, pistes ... 564
MIDI, réglage du volume ... 559
MIDI, un canal par voix ... 565
`midiBalance` ... 566
`midiChannelMapping` ... 564
`midiChorusLevel` ... 566
`midiDrumPitches` ... 429
`midiExpression` ... 566
`midiPanPosition` ... 566
`midiReverbLevel` ... 566
`min-systems-per-page` ... 580
mineur ... 23
`minimum-Y-extent` ... 596
`minimumFret` ... 372, 416
`minimumPageTurnLength` ... 595
`minimumRepeatLengthForPageTurn` ... 595
`\minor` ... 23
`minorChordModifier` ... 455
mirroring markup ... 778
mise en forme ... 583, 585
`mixed` ... 362
`\mixolydian` ... 23
mixolydien ... 23
`\mm` ... 669
modèle de musique arabe ... 505
Modèles pour ensemble vocal ... 304, 306, 329, 330, 347, 350, 632
Modèles pour quatuor à cordes ... 366
modale, inversion ... 17
modale, transformation ... 15
modale, transposition ... 16
`\modalInversion` ... 17, 853
`\modalTranspose` ... 16, 853
mode ... 23
`mode` ... 812
mode ancien ... 23
mode markup et caractères spéciaux ... 264
mode *markup* et guillemets ... 264
mode relatif et accord ... 174
mode relatif et `\autoChange` ... 358
`modern` ... 32
modern cautionary, style d'altération ... 32
modern voice, style d'altération de précaution ... 32
modern, style d'altération ... 31, 32
modern, style d'altération de précaution ... 32
`modern-cautionary` ... 32
modern-cautionary, style d'altération ... 31
`modern-voice` ... 32
`modern-voice-cautionary` ... 32
moderntab, clef ... 391
Modification de listes associatives ... 576, 597
Modification des propriétés d'un contexte ... 643
modification du style par défaut des barres de mesure ... 109
modification, respiration, signe ... 144
modifier des propriétés ... 652
`\mordent` ... 127, 805
motet ... 329
mouvements, plusieurs ... 510
`\mp` ... 129
multi-measure rest ... 68
multi-measure rest, within text, by duration ... 782
multi-measure rest, within text, by `duration-scale` ... 780
`\multi-measure-rest-by-number` ... 780
multiligne, *markup* ... 271
multiligne, texte ... 271
MultiMeasureRest ... 69, 232
MultiMeasureRestNumber ... 69, 232
MultiMeasureRestScript ... 69, 232
`MultiMeasureRestScript` ... 66
MultiMeasureRestText ... 69, 232
`MultiMeasureRestText` ... 66
multimesure, note, contraction ... 230
multimesure, note, expansion ... 230
multimesure, silence, contraction ... 230
multimesure, silence, expansion ... 230
multimesure, silence, nombre ... 231
multimesure, silence, style ... 231
multipage, *markup* ... 278
multiple, liaison de phrasé ... 142
multiples voix ... 183
multiples, liaisons d'articulation ... 139
multivoix, altérations ... 32, 34
Music ... 278
Music classes ... 224
musica ficta ... 477
musicale, citation ... 335
musicale, fonte ... 286
`\musicglyph` ... 119, 781
`\musicMap` ... 853
musicologie, crochet d'analyse ... 251
`musicQuotes` ... 812
musique échelonnée ... 56
musique ancienne, masquage de portée ... 212
musique ancienne, transcription ... 201, 498
musique arabe ... 501
musique arabe, exemple ... 505
musique dans un *markup* ... 275
musique en parallèle ... 194
musique entremêlée ... 194
musique et note de bas de page ... 529
musique non mesurée ... 78, 125
musique non mesurée et saut de ligne ... 79
musique non mesurée et saut de page ... 79
musique ottomane ... 506
musique répétitive ... 157
musique turque ... 506
mute bongo ... 807
mute conga ... 807
mute timbale ... 807

N

`\n` ... 129
n-olet ... 51
n-olet, chiffrage inhabituel ... 53
n-olet, formatage ... 52
n-olet, liaison ... 52
n-olet, ligature, saut de ligne ... 55
n-olet, modification du chiffre ... 53
n-olet, positionnement du crochet ... 51
n-olet, regroupment ... 51
n-olet, visibilité du crochet ... 52, 54
n-olet, visibilité du nombre ... 52
n-olets successifs ... 52
N.C., symbole ... 451

`\name` ... 644
`\natural` ... 781
naturel ... 6
naturel, harmonique ... 367
`neo-modern` ... 34
neo-modern, style d'altérations ... 34
`neo-modern-cautionary` ... 34
neo-modern-cautionary, style d'altérations ... 34
`neo-modern-voice` ... 34
neo-modern-voice, style d'altération ... 34
`neo-modern-voice-cautionary` ... 34
neomensural ... 469
neume carré et ligature ... 483
`\new` ... 630
New_fingering_engraver ... 238, 648
`\newSpacingSection` ... 614
niente, al, soufflet ... 134
no reset, style d'altérations ... 35
no-chord, symbole ... 451
`no-reset` ... 35
`\noBeam` ... 99
`\noBreak` ... 588
nom d'instrument ... 217
nom d'instrument, autres contextes ... 219
nom d'instrument, centrage ... 218
nom d'instrument, complexe ... 218
nom de note ... 1
nom de note, hollandais ... 6
nom de note, langue ... 8
nom de note, par défaut ... 6
nom de personnage ... 321
nom du chanteur ... 321
nombre de portées ... 209
noms d'instrument, centrés ... 218
non mesurée, musique ... 125
non musical, symbole ... 274
non traditionnelle, armure ... 24
non-ASCII, caractères ... 549
Non-imbrication des crochets et liaisons ... 141, 143
non-text font, in markup ... 286
NonMusicalPaperColumn ... 614
`nonstaff-nonstaff-spacing` ... 596
`nonstaff-relatedstaff-spacing` ... 596
`nonstaff-unrelatedstaff-spacing` ... 596
`\noPageBreak` ... 592, 853
`\noPageTurn` ... 595, 853
`\normal-size-sub` ... 749
`\normal-size-super` ... 267, 749
`\normal-text` ... 750
`\normalsize` ... 232, 268, 750
notation dans un *markup* ... 275
notation et graphisme ... 274
notation facile ... 41
Notation proportionnelle ... 613
notation, explication ... 248
notation, fonte ... 286
notation, taille ... 232
`\note` ... 782
note étouffée et cordes frettées ... 419
note colorisée dans un accord ... 244
note d'ornement ... 119
note d'ornement en fin de note ... 120
note d'ornement et retouche ... 121
note d'ornement, mise en forme ... 121
note d'ornement, synchronisation ... 124
note en couleur ... 242
note fantôme ... 244
note head ... 475, 491
note invisible ... 241
note masquée ... 241
note penchée ... 47
note pointée ... 49
note pointée, déplacement horizontal ... 187
note pointée, nombre de points ... 50
note profilée ... 43
note profilée, Aiken, variante fine ... 45
note silencieuse ... 61
note tenue au début du système suivant, masquage de l'altération ... 7
note value ... 50
note, décalage ... 183
note, décallage ... 187
note, division ... 83
note, impression du nom ... 246
note, longueur ... 48
note, nom selon la langue ... 8
note, prolongation ... 466
note, tête ancienne ... 475
note, taille standard ... 236
note, within text, by duration ... 782
note, within text, by `log` and `dot-count` ... 781
`\note-by-number` ... 781
note-collision-interface ... 833, 836, 838
note-event ... 41, 43, 47
`note-event` ... 223
note-head-interface ... 41, 43, 47
Note_head_line_engraver ... 360
Note_heads_engraver ... 41, 43, 47, 84, 646
`Note_heads_engraver` ... 83
Note_name_engraver ... 248
`Note_name_engraver` ... 246
Note_spacing_engraver ... 242
NoteCollision ... 188
NoteColumn ... 188
NoteHead ... 41, 43, 47
`\notemode` ... 667
NoteName ... 248
`noteNameFunction` ... 246
NoteNames ... 248
`NoteNames` ... 246
`noteNameSeparator` ... 246
notes inter-portée ... 355
Notes simultanées ... 188
notes simultanées et altération ... 36
notes, espacement horizontal ... 614
notes, nom en arabe ... 501
NoteSpacing ... 242, 613, 614
nouvelle portée ... 197
nuance personnalisée ... 136
nuance textuelle, style ... 135
nuance, alignement vertical ... 135
nuance, extension, personnalisation ... 258
nuance, ligne de prolongation, masquage ... 135
nuance, MIDI, personnalisation ... 558
nuances ... 129
nuances éditoriales ... 136
nuances entre les portées d'un système pianistique ... 354
nuances entre parenthèses ... 136
nuances successives sur une note tenue ... 131

nuances suggérées ... 136
nuances, positionnement vertical ... 132
\null ... 269, 797
NullVoice ... 318
numéro de corde ... 370
numéro de corde ou doigté ... 370
numéro de corde, positionnement ... 237
numéro de couplet ... 319
numéro de mesure ... 110, 113, 125
numéro de mesure à intervalle régulier ... 110
numéro de mesure et cadence ... 78
numéro de mesure et collision ... 116
numéro de mesure et mesure tronquée ... 112
numéro de mesure et reprise ... 164, 165
numéro de mesure, alignement ... 115
numéro de mesure, formatage ... 113
numéro de mesure, supression ... 115
numéro de page, référencement ... 537
numéro, notation *easy play* ... 42
numérotation des mesures, suppression ... 78
numérotation des pages ... 581
\number ... 750
\numericTimeSignature ... 70

O

objet altérable ... 811
objet de propriété ... 812
objet de rendu ... 810
objet en couleur ... 242
objet graphique, interface ... 811
objet graphique, propriétés ... 654
objet graphique, tracé ... 273
objet *markup* ... 254
objet Scheme ... 813
objet texte ... 254
objet, couleur ... 676
objet, rotation ... 683
objet, surimpression ... 676
objet, visibilité ... 675
Objets et interfaces ... 537, 811
objets graphiques ... 649
Objets inclus dans la portée ... 668, 669
octavation ... 28
octave absolue ... 1
octave et clef ... 18
octave relative et accord ... 5
octave relative, indication ... 2
octave, spécification ... 2
octave, vérification ... 10
\octaveCheck ... 10, 853
octaviation ... 25, 682
octaviation optionnelle ... 18
octaviation, allure du prolongateur ... 27
octaviation, texte ... 26
octaviation, une seule voix ... 26
\offset ... 659, 853
offset (décalage) ... 659
\omit ... 675, 853
\on-the-fly ... 527, 797
on-the-fly (à la volée) ... 527
\once ... 654, 656, 661, 695, 854
\oneVoice ... 179
opéra ... 332
opérette ... 332
\open ... 127, 366, 432, 806
open bongo ... 807
open conga ... 807
open high hat ... 127, 807
open timbale ... 807
opotionnelle, octaviation ... 18
Options avancées de lilypond ... 555
oratorio ... 329
orchestrale, musique ... 332
orchestre, cordes ... 366
orchestre, matériel d' ... 217
orchestre, notation pour ... 332
orenement, espacement ... 123
Organisation des fichiers LilyPond ... 515, 522
Organisation du code source avec des variables ... 196, 335, 543, 548, 549, 633, 634
orgue, indication, pédalier ... 127
orgue, marque de pédale ... 806
orgue, point d' ... 806
\oriscus ... 484, 490
ornement ... 119
ornement et ligature ... 99
ornement et paroles ... 322
ornement, « grupetto barré » ... 127
ornement, « grupetto Haydn » ... 127
ornement, « grupetto renversé » ... 127
ornement, « grupetto » ... 127
ornement, « mordant ascendant » ... 127
ornement, « mordant descendant » ... 127
ornement, « mordant » ... 127
ornement, « signum congruentiae » ... 127
ornement, « trille » ... 127
ornement, script ... 805
ornements ... 805
ossia ... 212
ossia ... 209, 217
ossia, positionnement ... 211
ottava ... 25
\ottava ... 25, 854
ottava-bracket-interface ... 28
Ottava_spanner_engraver ... 28
OttavaBracket ... 28
ottavation ... 26
ottavation-numbers ... 25
ottavation-ordinals ... 25
ottavation-simple-ordinals ... 25
ottavationMarkups ... 25
ottomane, musique classique ... 500
outer-margin ... 579
output-count ... 812
output-def ... 812
output-suffix ... 812
outside-staff-horizontal-padding ... 611
outside-staff-padding ... 611
outside-staff-priority ... 611
ouvert ... 432, 806
ouvert, soufflet ... 134
\oval ... 775
\overlay ... 765
\override ... 654, 658, 797
override ponctuel ... 656
override, annulation des effets ... 655
\override-lines ... 801
OverrideProperty ... 652

\overrideProperty ... 658, 854
\overrideTimeSignatureSettings ... 70, 854
overriding property within text markup ... 797
\overtie ... 750
overtie-ing text ... 750

P

\p ... 129
pédale de harpe ... 364
pédale de piano ... 362
pédale high hat ... 127
pédale sostenuto ... 362
pédale sustain ... 362
pédale sustain, style ... 362
pédale, diagramme pour harpe ... 364
pédale, indication combinée de ... 362
pédale, indication graphique de ... 362
pédale, indication textuelle de ... 362
pédale, style d'indications de ... 362
pédalier, « pointe » ... 127
pédalier, « talon » ... 127
pédalier, orgue indication ... 127
\pad-around ... 274, 765
\pad-markup ... 274, 766
\pad-to-box ... 274, 766
\pad-x ... 274, 766
padding ... 650
padding text ... 766
padding text horizontally ... 766
page, entête ... 522
page, format ... 572
page, mise en forme ... 617
page, numéro de la première ... 581
page, numérotation ... 581
page, numérotation automatique ... 581
page, numérotation en chiffres romains ... 581
page, pied ... 522
page, première ... 581
page, référencement du numéro ... 537
page, saut ... 617
page-breaking ... 580
page-breaking-system-system-spacing ... 580
page-count ... 580
\page-link ... 797
page-number-type ... 582
\page-ref ... 538, 798
page-spacing-weight ... 580
\pageBreak ... 592, 854
\pageTurn ... 595, 854
\palmMute ... 854
\palmMuteOn ... 854
pan, MIDI ... 566
Pango ... 279
\paper ... 513, 572
paper-height ... 574
paper-width ... 577
papier, orientation ... 573
papier, taille ... 572
\parallelMusic ... 194, 854
parenthèses, altération entre ... 6
parenthèses, note entre ... 244
parentheses-interface ... 245
ParenthesesItem ... 245
Parenthesis_engraver ... 245
\parenthesize ... 244, 776, 854
parlato ... 339
parlato, tête de note ... 40
Parmesan, glyphes ... 718
paroles ... 288
paroles assignées à une voix ... 179
paroles communes à plusieurs voix ... 318
paroles divisées (reprises) ... 316
paroles et barre de mesure ... 308
paroles et liaison de prolongation ... 314
paroles et ligature ... 90
paroles et mélodies ... 291
paroles et *markup* ... 289
paroles et ornement ... 322
paroles et \partCombine ... 192
paroles et répétition ... 308
paroles, alignement horizontal ... 307
paroles, alignement sur la mélodie ... 289
paroles, alignement sur une mélodie épisodique .. 635
paroles, blanc ... 63
paroles, espacement version 2.12 ... 304
paroles, garder dans les marges ... 308
paroles, gestion de l'espacement ... 306
paroles, identificateur ... 300
paroles, mise en forme ... 288
paroles, positionnement ... 211, 302
paroles, reprise avec alternative ... 312
paroles, saut de note ... 63
paroles, variables ... 300
parser ... 812
parser, variable ... 812
part ... 193
\partCombine ... 189, 318, 855
\partCombine et paroles ... 192, 318
\partCombineApart ... 190
\partCombineAutomatic ... 190
\partCombineChords ... 190
\partCombineDown ... 855
\partCombineForce ... 855
partCombineListener ... 812
PartCombineMusic ... 193
\partCombineSoloI ... 190
\partCombineSoloII ... 190
\partCombineUnisono ... 190
\partCombineUp ... 855
\partial ... 77, 158, 855
partie vocale ... 329
parties combinées, texte ... 193
parties, combiner des ... 189
partition incluse dans un *markup* ... 277
Partition pour chœur à quatre voix mixtes .. 329, 330
\partocombine, texte ... 193
\path ... 776
path, drawing ... 776
\pattern ... 798
paysage, papier ... 573
PDF metadata ... 528
pedal high hat ... 807
pedalSustainStyle ... 362
percent ... 168
percent repeat ... 170
Percent_repeat_engraver ... 170
PercentRepeat ... 170
PercentRepeatCounter ... 170

PercentRepeatedMusic 170
Percussion 423, 424, 425, 428, 430
percussion, clef 424, 740
percussion, fausse note 430
percussion, note fantôme 430
percussion, note silencieuse 430
percussion, portée 197
percussions 423, 425
percussions, nom des notes 807
percussions, personnalisation 428
perse, makam 500
personnage, indication 333
personnalisation de tablature 389
Personnalisation des indications de nuance 254
personnalisation, diagramme de fret 408
`\pes` 490
petite note 119
petites notes 221, 434
petites notes, formater des 224
Petrucci 468, 470
Petrucci, clef 473, 740
phrasé, liaison de 141
phrasé, pour des paroles 296
PhrasingSlur 143
`\phrasingSlurDashed` 142
`\phrasingSlurDashPattern` 142, 855
`\phrasingSlurDotted` 142
`\phrasingSlurDown` 142
`\phrasingSlurHalfDashed` 142
`\phrasingSlurHalfSolid` 142
`\phrasingSlurNeutral` 142
`\phrasingSlurSolid` 142
`\phrasingSlurUp` 142
`\phrygian` 23
phrygien 23
`piano` 33
piano cautionary, style d'altérations 33
piano et altérations 33
piano et pédale 362
piano, nuances entre les portées 354
piano, style d'altérations 33
piano, système 199
piano, système pour 354
`piano-cautionary` 33
Piano_pedal_engraver 363
PianoPedalBracket 363
PianoStaff 36, 153, 202, 221, 330, 355
`PianoStaff` 354, 357
pied de page 515, 522
pied, indication 127
`pipe, symbole` 116
Pitch names 2, 6, 8, 10, 477
Pitch_squash_engraver 47, 87, 646, 824
`Pitch_squash_engraver` 85
`\pitchedTrill` 155, 855
Pitches .. 2, 6, 8, 10, 11, 14, 22, 25, 28, 29, 36, 40, 41, 43, 46, 47, 473, 504
`pitchnames` 812
pizzicato, Bartók 368
pizzicato, *snap* 368
placing horizontal brackets, around text 775
placing parentheses, around text 776
placing vertical brackets, around text 771
plat, soufflet 134
plein, trait de liaison 139, 142
plusieurs lignes de texte 272
plusieurs mouvements 510
plusieurs pages de texte 278
point 49
point d'arrêt 143, 806
point d'augmentation, modification du nombre 50
point d'orgue 119, 127, 806
point d'orgue et silence multimesure 66
point d'orgue sur une barre de mesure 259
point de contrôle et tweak 658
pointée, note 49
`\pointAndClickOff` 855
`\pointAndClickOn` 855
`\pointAndClickTypes` 855
pointe 806
pointillé, trait de liaison 139, 142
points de contrôle, courbe de Bézier 690
polices, choix par défaut 284
polymétrie 80
polymétrie et ligature 80
polymétrique, partition 637
polymetric 55, 82
polymetric time signature 82
polyphone et tablature 382
polyphonie 183
polyphonie, mêmes paroles 318
polyphonie, portée simple 179
polyphonie, voix additionnelle 186
polyphony 188
ponctuation 288
ponctuation et paroles 288
portée à la française 209
portée à quatre mesures 590
portée de batterie 197
portée de percussion 197
portée de piano 354
portée multiple 199
portée rythmique 197
portée simple 197
portée simple et polyphonie 179
portée unique avec crochet de regroupement 201
portée vide 212
portée, arpège, enjambement 152
portée, changement automatique 357
portée, définition de la taille 585
portée, division 215
portée, initialisation 197
portée, instanciation 197
portée, lignes de 205
portée, nouvelle 197
portée, reprise 205
portée, suspension 205
portée, transcription de grégorien 197
portées pour instrument à clés 354
portées pour instrument à clavier 354
portées, espacement 595
portées, groupe de 199
portées, nombre variable de 209
portées, regroupement 199
portées, regroupement et imbrication 202
portéesrythmique, masquage 212
portato 129
portato 127
`\portato` 127, 805
position, vertical, *grobs* 667

positionnemenet, diagramme de fret 401
Positionnement des objets 128, 129, 257
positionnement des paroles 302
positionnement des silences multimesure 67
positionnement vertical des *grobs* 667
Positionnement vertical des paroles 328
positionnement, doigté 237
positionnement, doigté main droite 417
positionnement, numéro de corde 237
positionnement, ossia 211
positionnement, paroles 211
postévénement 805
postscript 275
`\postscript` 275, 777
pouce *(thumb)* 805
pouce, doigté 237
pourcent, compteur de reprise 169
pourcent, répétition 168
pourcent, reprise isolée 169
poussé, indication d'archet 366
pousser l'archet 806
power chord 421
power chords 420
`\pp` 129
`\ppp` 129
`\pppp` 129
`\ppppp` 129
précaution, altération de 6
prédéfini, diagramme de fret, définition 409
présentation mensurale 201
`\prall` 127, 805
`\pralldown` 127, 805
`\prallmordent` 127, 805
`\prallprall` 127, 805
`\prallup` 127, 805
`\preBend` 377, 855
`\preBendHold` 377, 855
`predefinedDiagramTable` 412
`\predefinedFretboardsOff` 414
`\predefinedFretboardsOn` 414
première fois 158
première page 581
prima volta 159
`print-all-headers` 582
`print-first-page-number` 581
`print-page-number` 581
`printAccidentalNames` 246
`printNotesLanguage` 246
`printOctaveNames` 246
prob 812
prologation de note 466
prolongateur 299
prolongateur, octaviation 27
prolongation de texte 257
`\property-recursive` 798
`\propertyOverride` 855
`\propertyRevert` 855
PropertySet 652
`\propertySet` 856
`\propertyTweak` 856
`\propertyUnset` 856
proportionnel, espacement 617
propriété altérable 811
propriété commune 811
propriété objet 812
propriété partagée 811
propriétés 652
propriétés d'objet graphique 654
propriétés d'un grob 654
Propriétés des objets de rendu 811
Propriétés listées par interface 811
Psalmodie 350
psalmodie 340
psaume 346
`\pt` 669
pull off 388
pulsation, regroupement 95
pupitre, partition de 217
pure containers, Scheme 696
`\pushToTag` 548, 856
`\put-adjacent` 766
putting space around text 766

Q

`q`, répétition d'accord 175, 373
quadrillage temporel 249
quadrillage temporel, apparence 250
qualité d'accord 446
quart de ton 6, 8
quart de ton, tablature 392
quarter tone 8
`\quilisma` 484, 490
`quotedCueEventTypes` 223
`quotedEventTypes` 223
`\quoteDuring` 221, 224, 856
QuoteMusic 224

R

`r` 61
réglage fin d'un luth 423
réglage par défaut, modification 638
réglages, globalisation 549
répétition 105, 157
répétition courte 168
répétition de mesure 168
répétition et liaison de tenue 58
répétition, pourcent 168
répétition, utilisation de `q` 175, 373
réplique 221
réplique, fin 228
réplique, formatage 224
rétrograde, transformation 15
rôle 321
rôle, indication 333
`R` 65
Référence des propriétés internes 628
`Références bibliographiques` 453, 457
Répétitions et reprises 308
raccordement dans une balise 548
ragged, aligné 571
`ragged-bottom` 574
`ragged-last` 578, 617
`ragged-last-bottom` 574
`ragged-right` 578, 617
`\raise` 269, 767
`\raiseNote` 450, 856
raising text 767

rappel ... 810
rast ... 504
Ratisbona, Editio ... 469
rectangle en front de regroupement ... 200
rectiligne, crochet ... 100
`\reduceChords` ... 85, 856
referencing page label, in text ... 801
referencing page number, in text ... 797, 798
registre, symbole pour accordéon ... 363
regroupement de balises ... 544
regroupement de n-olets ... 51
regroupement, rectangle en front ... 200
regroupements de balises ... 547
RehearsalMark ... 119, 262
relatif ... 2
`\relative` ... 2, 14, 358, 856
RelativeOctaveCheck ... 11
RelativeOctaveMusic ... 6
religieuse, musique ... 340
reliure ... 578
`\remove` ... 637
`remove-empty` ... 213
`remove-first` ... 213
`remove-grace-property` ... 122
`remove-layer` ... 215
`\RemoveAllEmptyStaves` ... 212, 860
`\RemoveEmptyStaves` ... 212, 860
`\removeWithTag` ... 544, 856
renaissance, musique ... 201
rendu, interfaces de ... 649
rendu, objet de ... 810
repère manuel ... 118
repère, format ... 118
repère, indication de ... 117
repère, personnalisation ... 118
repère, positionnement en fin de ligne ... 260
repère, sous la portée ... 75
repère, style ... 118
repère, sur toutes les portées ... 261
repeat ... 165
`\repeat` ... 157, 158, 159
`\repeat percent` ... 168
`\repeat tremolo` ... 170
`\repeat unfold` ... 157
`\repeat volta` ... 158, 159
`repeatCommands` ... 166
`repeatCountVisibility` ... 169
RepeatedMusic ... 158, 165, 167
Repeats ... 158, 165, 167, 170, 172
RepeatSlash ... 170
RepeatSlashEvent ... 170
`\repeatTie` ... 58, 314
`\replace` ... 751
reprise ... 105, 157
reprise ambiguë ... 165
reprise avec alternative ... 157
reprise avec alternative et liaison de prolongation ... 58
reprise avec alternative et paroles ... 312
reprise avec levée ... 158
reprise courante ... 158
reprise développée ... 563
reprise de portée ... 205
reprise et accord ... 457
reprise et anacrouse ... 158
reprise et contrôle de barre de mesure ... 158
reprise et glissando ... 149, 165
reprise et liaison ... 165
reprise et numéro de mesure ... 164
reprise et paroles ... 308
reprise et *segno* ... 161
reprise manuelle ... 166
reprise, crochet raccourci ... 163
reprise, expansion ... 157
reprise, fin alternative ... 159
reprise, pourcent, compteur ... 169
reprise, script ... 806
reprises imbriquées ... 165
reprises successives, barre de mesure ... 163
`\resetRelativeOctave` ... 5, 856
respiration, indication ... 143
respiration, modification du symbole ... 144
`\responsum` ... 485
Rest ... 63
`\rest` ... 61, 782
rest, within text, by duration ... 782
rest, within text, by `log` and `dot-count` ... 782
`\rest-by-number` ... 782
`rest-event` ... 223
Rest_engraver ... 84
RestCollision ... 188
`restNumberThreshold` ... 231
`restrainOpenStrings` ... 372
retouche (*tweak*) ... 656
retouche de note d'ornement ... 121
Retouche de partition ... 628, 684
`Retouches complexes` ... 696
retour au propriétés par défaut de la métrique ... 71
`\retrograde` ... 15, 856
reverb MIDI ... 566
`\reverseturn` ... 127, 805
`\revert` ... 655
RevertProperty ... 652
`\revertTimeSignatureSettings` ... 71, 856
`\rfz` ... 129
rgb, couleur ... 243
`rgb-color` ... 243
`\rheel` ... 127, 806
Rhythmic_column_engraver ... 646
RhythmicStaff ... 47, 87, 198
`RhythmicStaff` ... 197
Rhythms ... 50, 55, 57, 61, 63, 64, 68, 73, 78, 79, 82, 84, 87, 89, 97, 101, 102, 110, 116, 117, 119, 123, 125, 126
ride bell ... 807
ride cymbal ... 807
`\right-align` ... 268, 767
right-aligning text ... 767
`\right-brace` ... 798
`\right-column` ... 767
`right-margin` ... 578
`\rightHandFinger` ... 417, 856
`\roman` ... 751
`\romanStringNumbers` ... 366, 370
`\rotate` ... 768
rotating text ... 768
`\rounded-box` ... 273, 778
`\rtoe` ... 127, 806
rvb, couleur ... 243
rythmique d'une mélodie ... 84

S

`s` ... 63
sélection de la taille (notation) ... 232
séparatuer d'accord ... 457
`\sacredHarpHeads` ... 43
`\sacredHarpHeadsMinor` ... 44
saisie, ignorer des passages ... 552
`\sans` ... 751
SATB ... 329
saut ... 145
saut de durée ... 63
saut de ligne ... 103
saut de ligne et cadences ... 79
saut de ligne et ligature ... 88
saut de ligne et musique non mesurée ... 79
saut de ligne manuel ... 588
saut de ligne régulier ... 590
saut de page ... 617
saut de page et cadences ... 79
saut de page et musique non mesurée ... 79
saut, gestion sur voix dédiée ... 591
scalable vector graphics ... 553
`\scale` ... 778
`\scaleDurations` ... 56, 80, 856
scaling markup ... 778
scaling text ... 768
Scheme, objet ... 813
Scheme, *pure container* ... 696
Scheme, *unpure container* ... 696
Scheme, variable ... 812
scordatura ... 25
Score ... 126, 814, 817
`\score` ... 509, 513, 783
`\score-lines` ... 801
`score-markup-spacing` ... 576
`score-system-spacing` ... 576
`scoreTitleMarkup` ... 523
Scottish highland bagpipe ... 434
Script ... 128, 129, 483
script ... 805
script et silence multimesure ... 66
script textuel, alignement vertical ... 135
Script_engraver ... 483
ScriptEvent ... 483
scripts, ordre vertical ... 128
seconda volta ... 159
seconde fois ... 158
segno ... 104, 106, 119
`\segno` ... 127, 806
segno et reprise ... 161
segno sur une barre de mesure ... 259
self-alignment-interface ... 649, 684
`self-alignment-X` ... 596
semai ... 505
semi-transparence et couleurs ... 243
`\semicirculus` ... 482, 807
semicirculus, grégorien, articulation ... 482
`\semiflat` ... 784
`\semiGermanChords` ... 453
`\semisharp` ... 784
septième ... 446
septième majeure, symbole ... 455
sesqui-bémol ... 9
sesqui-dièse ... 9
`\sesquiflat` ... 784
`\sesquisharp` ... 785
`\set` ... 90, 652, 658
`set-global-fonts` ... 284
`set-global-staff-size` ... 585
`set-octavation` ... 25
setting extent of text object ... 800
setting horizontal text alignment ... 760
setting subscript, in standard font size ... 749
setting superscript, in standard font size ... 749
`\settingsFrom` ... 857
`\sf` ... 129
`\sff` ... 129
`\sfz` ... 129
`\shape` ... 691, 857
sharp ... 8
`\sharp` ... 785
`\shiftDurations` ... 857
`\shiftOff` ... 183
`\shiftOn` ... 183
`\shiftOnn` ... 183
`\shiftOnnn` ... 183
`short-indent` ... 219, 579
`\shortfermata` ... 127, 806
`show-available-fonts` ... 283
`showFirstLength` ... 552, 812
`\showKeySignature` ... 434
`showLastLength` ... 552, 812
`\showStaffSwitch` ... 359
side-position-interface ... 649, 684
sidestick ... 807
signe « snappizzicato » ... 127
signe de mensuration ... 474
signe de respiration, modification ... 144
signe, « segno » ... 127
signet ... 537
signets ... 538
`\signumcongruentiae` ... 127, 807
sikah ... 504
silence ... 61
silence ancien ... 476
silence d'église ... 231
silence d'espacement ... 63
silence invisible ... 63
silence multimesure ... 61, 65
silence multimesure et doigté ... 69
silence multimesure et *markup* ... 68
silence multimesure et point d'orgue ... 66
silence multimesure, étiquette ... 66
silence multimesure, ajout de texte ... 66
silence multimesure, longueur ... 66
silence multimesure, positionnement ... 67
silence multimesure, script ... 66
silence multimesure, style ... 231
silence, décalage automatique ... 183
silence, division ... 83
silence, mesure entière ... 65
silence, spécification du positionnement vertical ... 61
Silences valant une mesure ... 230
silences, collision entre ... 69
silences, condenser les ... 69
silencieuse, note, percussion ... 430
simile ... 170
`\simple` ... 752
simple text string ... 752

simple text string, with tie characters ... 785
simultanée, liaison de phrasé ... 142
simultanées, liaisons d'articulation ... 139
Simultaneous notes .. 174, 178, 182, 183, 188, 193, 196
`\single` ... 532, 661, 857
`\skip` ... 63, 857
`skipBars` ... 231
SkipMusic ... 64
`skipTypesetting` ... 552
Slash_repeat_engraver ... 170
`slashChordSeparator` ... 454
slashed digit ... 798
`\slashed-digit` ... 798
`\slashedGrace` ... 119, 857
`\slashSeparator` ... 582
`\slashturn` ... 127, 805
Slur ... 141, 153
slur ... 141
slur, defining dash patterns ... 141
`slur-event` ... 223
`\slurDashed` ... 139
`\slurDashPattern` ... 140, 857
`\slurDotted` ... 139
`\slurDown` ... 139
`\slurHalfDashed` ... 139
`\slurHalfSolid` ... 139
`\slurNeutral` ... 139
`\slurSolid` ... 139
`\slurUp` ... 140
`\small` ... 232, 268, 752
`\smallCaps` ... 752
`\smaller` ... 265, 268, 752
smob ... 813
snap pizzicato ... 368
`\snappizzicato` ... 127, 806
snare ... 807
sol, clef de ... 18
Solesmes ... 469
solfège ... 246
solo ... 189
son ... 556
soprano, clef ... 740
sortie, définition ... 628, 812
sos. ... 362
sostenuto, pédale ... 362
SostenutoEvent ... 363
`\sostenutoOff` ... 362
`\sostenutoOn` ... 362
SostenutoPedal ... 363
SostenutoPedalLineSpanner ... 363
soufflet ... 130
soufflet et barre de mesure ... 132
soufflet Ferneyhough ... 134
soufflet ouvert ... 134
soufflet penché ... 683
soufflet plat ... 134
soufflet, al niente ... 134
soufflet, alignement directionnel
sur `NoteColumn` ... 133
soufflet, déplacement de l'extrémité ... 133
soufflet, ligature ... 102
soufflet, longueur minimale ... 133
souligné ... 265
`\sourcefileline` ... 514
`\sourcefilename` ... 514
sourdine ... 806
sous-ligature, orientation ... 95
`\southernHarmonyHeads` ... 43
`\southernHarmonyHeadsMinor` ... 44
`\sp` ... 129
spécification d'un repère ... 118
Spacing .. 575, 577, 579, 583, 585, 586, 592, 593, 594, 595, 600, 601, 602, 611, 612, 614, 615, 617, 624, 625, 627
`spacing` ... 613
spacing-spanner-interface ... 842, 844
SpacingSpanner ... 612, 613, 614, 615
`\spacingTweaks` ... 857
`Span_stem_engraver` ... 360
SpanBar ... 110
spatialisation (pan), MIDI ... 566
splash cymbal ... 807
`\spp` ... 129
Sprechgesang ... 339
stéréo MIDI, balance ... 566
staccatissimo ... 127
`\staccatissimo` ... 127, 805
staccato ... 129
staccato ... 127
`\staccato` ... 127, 805
stacking text in a column ... 756
Staff 36, 40, 82, 198, 202, 217, 221, 253, 613, 814
staff ... 198, 208, 212
Staff notation .. 77, 198, 202, 204, 205, 208, 212, 217, 221, 224, 229
`staff-affinity` ... 596
`staff-padding` ... 238
`\staff-space` ... 669
`staff-staff-spacing` ... 596
staff-symbol-interface ... 208
`Staff.midiInstrument` ... 562, 567
`Staff_collecting_engraver` ... 261
Staff_symbol_engraver ... 217
`Staff_symbol_engraver` ... 212
StaffGroup ... 116, 202, 204
`staffgroup-staff-spacing` ... 596
StaffGrouper ... 331, 597, 599, 601, 664
StaffSpacing ... 614
StaffSymbol ... 198, 208, 212
standard, taille de note ... 236
StanzaNumber ... 327
`start-repeat` ... 166
`startAcciaccaturaMusic` ... 122
`startAppoggiaturaMusic` ... 122
`startGraceMusic` ... 122
`\startGroup` ... 251
`\startStaff` ... 205, 209
`\startTrillSpan` ... 154
staves ... 198
`\stdBass` ... 790
`\stdBassIV` ... 791
`\stdBassV` ... 792
`\stdBassVI` ... 793
Stem ... 246, 362, 660
`Stem` ... 360
stem-interface ... 246
`stem-spacing-correction` ... 613
Stem_engraver ... 101, 246
`\stemDown` ... 245
`stemLeftBeamCount` ... 100
`\stemNeutral` ... 245

stemRightBeamCount ... 100
\stemUp ... 245
stencil ... 813
\stencil ... 799
stencil, suppression ... 675
stopAcciaccaturaMusic ... 122
stopAppoggiaturaMusic ... 122
stopGraceMusic ... 122
\stopGroup ... 251
\stopped ... 127, 428, 432, 806
\stopStaff ... 205, 209, 212
\stopTrillSpan ... 154
\storePredefinedDiagram ... 407, 412, 857
strictBeatBeaming ... 95
\string-lines ... 801
StringNumber ... 371
stringNumberOrientations ... 237
\stringTuning ... 390, 857
stringTunings ... 389, 403
StrokeFinger ... 418
strokeFingerOrientations ... 237, 417
\stropha ... 484, 490
Structuration de la saisie des notes ... 805
\strut ... 799
style d'accidentelle ... 29
style d'altération *modern* ... 32
style d'altération *modern accidental* ... 32
style d'altération *neo-modern-cautionary* ... 34
style d'altération *neo-modern-voice* ... 34
style de métrique ... 70
style de repère ... 118
style de trait, liaison ... 139, 142
style de voix ... 182
style, nuance textuelle ... 135
style, silence multimesure ... 231
style, tête de note ... 40
\styledNoteHeads ... 857
\sub ... 267, 753
subbass, clef ... 740
subdivideBeams ... 94
subdividing beams ... 94
subdivision de ligature ... 94
subscript text ... 753
substituant pour événement ... 174
substitution de doigt ... 236
substitution, fonction ... 699
suggestAccidentals ... 128, 477
\super ... 267, 753
superscript text ... 753
suppression, accords répétés ... 411
surimpression d'objet ... 676
surimpression de *grob* ... 59
sus ... 449
suspension de portée ... 205
sustain, pédale ... 362
sustain, style de pédale ... 362
SustainEvent ... 363
\sustainOff ... 362
\sustainOn ... 362
SustainPedal ... 363
SustainPedalLineSpanner ... 363
SVG, format de sortie ... 553
swing, script ... 568
swing.ly ... 568
syllabe, durée automatique ... 291
symbole arabe d'un demi-bémol ... 502
symbole de portée ... 205
symbole de septième majeure ... 455
symbole non musical ... 274
symbole, arpège ... 151
symboles de portée, dessin ... 670
synchronisation des notes d'ornement ... 124
syntaxe du mode *markup* ... 263
système ... 199
système choral ... 199
système pianistique ... 199
système, début de ... 199
système, espacement des portées ... 595
système, grand ... 199
système, indicateur de séparation ... 204
système, rectangle en front ... 200
systèmes imbriqués ... 202
system-count ... 580
system-separator-markup ... 582
system-system-spacing ... 576
systems-per-page ... 580
SystemStartBar ... 202, 204
SystemStartBrace ... 202, 204
SystemStartBracket ... 202, 204
SystemStartSquare ... 202, 204

T

ténor, clef ... 18
tête de note ... 232
tête de note allongée ... 47
tête de note en losange ... 367
tête de note et improvisation ... 47
tête de note, Aiken ... 43
tête de note, allure ... 43
tête de note, apprentissage ... 41
tête de note, Christian Harmony ... 43
tête de note, Funk ... 43
tête de note, Harmonia Sacra ... 43
tête de note, harpe sacrée ... 43
tête de note, spéciale ... 40
tête de note, style ... 182, 740
tête de note, Walker ... 43
tab, clef ... 740
Tab_note_heads_engraver ... 393
\tabChordRepeats ... 373, 857
\tabChordRepetition ... 857
\tabFullNotation ... 372
tablature ... 197, 369
tablature et désinence ... 377
tablature et glissando ... 386, 387
tablature et hampe ... 381
tablature et harmonique ... 382
tablature et indication d'harmonique ... 375
tablature et micro-intervalle ... 392
tablature et polyphonie ... 382
tablature et quart de ton ... 392
tablature par défaut ... 371
tablature pour banjo ... 369, 421
tablature pour guitare ... 369
tablature, accordages prédéfinis ... 389
tablature, base ... 371
tablature, clef ... 391
tablature, hammer on ... 388

tablature, luth ... 422
tablature, pull off ... 388
\table ... 802
table de doigtés ... 433
table des matières, personnalisation ... 540
\table-of-contents ... 541, 803
TabNoteHead ... 389
TabStaff ... 198, 389
TabStaff ... 197, 371
TabVoice ... 389
TabVoice ... 371
tag ... 544
\tag ... 544, 857
\tagGroup ... 547, 858
taille des notes ... 232
Taille des objets ... 212
taille et fonte ... 586
taille, flageolet ... 432
talon ... 806
tam tam ... 807
tambourine ... 807
\taor ... 434
taqasim ... 505
taqasim ... 504
teaching ... 35
teaching, style d'altérations ... 35
\teeny ... 232, 268, 753
tempo ... 73
\tempo ... 73
tempo en *markup* ... 76
tempo indication ... 77
tempo sous la portée ... 75
tempo, changement masqué ... 76
\temporary ... 661, 695, 858
temporel, quadrillage ... 249
temporelle, note de bas de page ... 531
temps, gestion du ... 125
tenor G, clef ... 740
tenor varC, clef ... 740
tenor, clef ... 740
tenue et arpeggio ... 60
tenue et nuances successives ... 131
tenue, gravure manuelle ... 60
tenue, liaison ... 57
tenuto ... 129
tenuto ... 127
\tenuto ... 127, 805
tessiture ... 37
test de mesure ... 116
Text ... 256, 257, 259, 261, 263, 265, 268, 273, 275, 278, 279, 284
text ... 362, 753
text column, left-aligned ... 764
text column, right-aligned ... 767
text, line width ... 272
text-interface ... 649, 797
text-script-interface ... 649
Text_engraver ... 646
texte ajouté ... 263
texte en colonnes ... 262, 271
texte en préambule ... 262
texte et alternative ... 167
texte et extenseur ... 257
texte et liaison ... 140
texte et rembourrage ... 274
texte hors marges ... 257
texte indépendant ... 262
texte indépendant et note de bas de page ... 534
texte indiquant le nombre de mesures vides ... 66
texte isolé ... 262
texte justifié ... 272
texte multiligne ... 271
texte sur plusieurs lignes ... 272
texte, alignement ... 268
texte, alignement horizontal ... 268
texte, alignement vertical ... 269
texte, autres langues ... 254
texte, centrage sur la page ... 271
texte, décoration ... 273
texte, encadrement ... 273
texte, inclusion dans une liaison ... 140
texte, maintien dans les marges ... 257
texte, mise en forme des extenseurs ... 257
texte, mise en forme des prolongations ... 257
texte, nuance, style ... 135
texte, objet ... 254
texte, octaviation ... 26
texte, taille ... 265
texte, top-level ... 262
\textLengthOff ... 66, 68, 257
\textLengthOn ... 66, 68, 135, 257
TextScript ... 129, 257, 263, 268, 273, 275, 278, 279, 444
TextSpanner ... 259, 675
\textSpannerDown ... 257
\textSpannerNeutral ... 257
\textSpannerUp ... 257
The Emmentaler font ... 781
\thumb ... 127, 237
tie ... 61, 84
Tie ... 61
\tie ... 754
tie-ing text ... 754
TieColumn ... 61, 694
TieColumn ... 60
\tied-lyric ... 785
\tieDashed ... 59
\tieDashPattern ... 59, 858
\tieDotted ... 59
\tieDown ... 59
\tieHalfDashed ... 59
\tieHalfSolid ... 59
\tieNeutral ... 59
\tieSolid ... 59
\tieUp ... 59
tieWaitForNote ... 60
timbale ... 807
\time ... 69, 90, 858
time signature ... 73
\times ... 858
TimeScaledMusic ... 56
TimeSignature ... 73, 82
timeSignatureFraction ... 80
Timing_translator ... 73, 78, 82, 110, 126, 817
\tiny ... 232, 268, 754
tiré, indication d'archet ... 366
tirer l'archet ... 806
tiret, trait de liaison ... 139, 142
tirettes d'accordéon, symboles ... 363
titre ... 515
tocFormatMarkup ... 541

`tocIndentMarkup` ... 541
`\tocItem` ... 541, 858
`tocItemMarkup` ... 541
`\tocItemWithDotsMarkup` ... 540
`tocTitleMarkup` ... 541
tom tom ... 807
Top ... 1, 628, 649
top-level, texte ... 262
`top-margin` ... 574
`top-markup-spacing` ... 577
`top-system-spacing` ... 577
`toplevel-bookparts` ... 812
`toplevel-scores` ... 812
Tout savoir sur les graveurs ... 84
trémolo ... 170
trémolo interportée ... 171
trémolo, indication de ... 171
trémolo, ligature de ... 170
tracé d'objet graphique ... 273
trait d'union ... 299
transcription de musique ancienne ... 201
transcription, grégorien ... 346, 495
transcription, musique ancienne ... 498
transformation rétrograde ... 15
`\translate` ... 270, 768
`\translate-scaled` ... 270, 768
translating text ... 768
Translation ... 648
transparence, semi ... 243
`\transparent` ... 799
transparent, objet ... 676
transparente, note ... 241
`\transpose` ... 11, 14, 858
`\transposedCueDuring` ... 228, 858
TransposedMusic ... 14
transposing instrument ... 29, 333
transpositeur, instrument ... 12
transposition ... 11
`\transposition` ... 28, 221, 858
transposition des hauteurs ... 11
transposition et clef ... 18
transposition et diagramme de fret ... 405
transposition et MIDI ... 28
transposition, instrument ... 28
Travail sur les fichiers d'entrée ... 510
tre corde ... 362
treble, clef ... 740
`\treCorde` ... 362
`tremolo` ... 170
tremolo, cross-staff ... 172
`tremoloFlags` ... 171
triade ... 446
triangle ... 807
`\triangle` ... 274, 778
trill ... 156
`\trill` ... 127, 154, 805
trille ... 154
trilles avec hauteur explicite ... 155
trilles avec hauteur explicite et altération ... 155
TrillSpanner ... 156, 675
triolet ... 51
triolet, formatage ... 52
triolet, liaison ... 52
triplet ... 55
`\tripletFeel` ... 568
tronqué, glissando ... 148
Tunable context properties ... 298, 299, 654
tuplet ... 55
`\tuplet` ... 51, 80, 858
`tuplet-slur` ... 52
TupletBracket ... 56
`\tupletDown` ... 51
`\tupletNeutral` ... 51
TupletNumber ... 56
`TupletNumber` ... 53
`\tupletSpan` ... 52, 858
`tupletSpannerDuration` ... 52
`\tupletUp` ... 51
turc, nom de note ... 506
`\turn` ... 127, 805
turque, exemple de makam ... 507
turque, makam ... 500
turque, musique ... 506
turque, musique classique ... 500
`Tutoriel Scheme` ... 628
`\tweak` ... 656, 658, 859
tweak (retouche, affinage) ... 656
tweak et point de contrôle ... 658
tweak, relation avec `\override` ... 658
Tweaks and overrides ... 684
`two-sided` ... 578
`\type` ... 644
type de caractère ... 810
`\typewriter` ... 754

U

U.C. ... 362
ukulele ... 394
ukulele, tablature ... 389
ultima volta ... 159
una corda ... 362
`\unaCorda` ... 362
UnaCordaEvent ... 363
UnaCordaPedal ... 363
UnaCordaPedalLineSpanner ... 363
unbreakable-spanner-interface ... 89
`\underline` ... 265, 754
underlining text ... 754
`\undertie` ... 755
undertie-ing text ... 755
`\undo` ... 661, 859
une pause par mesure ... 65
`unfold` ... 157
`\unfolded` ... 859
UnfoldedRepeatedMusic ... 158, 165
`\unfoldRepeats` ... 563, 859
Unfretted strings ... 366
`\unHideNotes` ... 241
Unicode ... 550
unpure containers, Scheme ... 696
`\unset` ... 653
`\upbow` ... 127, 366, 806
`\upmordent` ... 127, 805
`\upprall` ... 127, 805
`\upright` ... 755
ut, clef d' ... 18
UTF-8 ... 549
Utilisation en ligne de commande ... 552, 553

V

vérification d'octave ... 10
vérification des limites de mesure ... 116
varbaritone, clef ... 740
varC, clef ... 740
`\varcoda` ... 127, 806
variables ... 514
variables, utilisation de ... 543
variante ... 209
variante rythmique ... 316
Vaticana, Editio ... 468, 469
VaticanaStaff ... 198
VaticanaStaff ... 479
`VaticanaStaff` ... 197
VaticanaVoice ... 479
`\vcenter` ... 768
vents ... 431
vents, doigtés ... 432
vents, doigtés, diagramme ... 440
`\verbatim-file` ... 799
`\version` ... 514
`\versus` ... 485
vertical, alignement de nuance ... 135
vertical, alignement de script textuel ... 135
vertical, espacement ... 595, 617
vertical, ordre des scripts ... 128
vertical, positionnement des nuances ... 132
vertical, positionnement forcé des *grobs* ... 667
VerticalAxisGroup ... 217, 331, 597, 599, 600, 601, 602, 860
`VerticalAxisGroup` ... 596
vertically centering text ... 768
`\verylongfermata` ... 127, 806
`\veryshortfermata` ... 127, 806
vibraslap ... 807
vide, accord ... 122, 355
violin, clef ... 740
`\virga` ... 484, 490
`\virgula` ... 482
visibilité d'objets ... 675
visibilité d'une clef transposée ... 682
visibilité des hampes ... 245
visibilité des n-olets ... 52
Visibilité et couleur des objets ... 64, 217, 242, 347, 637, 675, 678, 682
visibilité, compteur, pourcent ... 169
visibilité, crochet de n-olet ... 54
Vocal music ... 288, 328, 330, 333, 338, 340
vocale, partition ... 332
vocalise ... 296
Voice ... 40, 47, 193, 224, 229, 295, 613, 651
`Voice` ... 179
`voice` ... 29, 31
voice, style d'altérations ... 31
VoiceFollower ... 360, 675
`\voiceFour` ... 179
`\voiceFourStyle` ... 182
`\voiceNeutralStyle` ... 182
`\voiceOne` ... 179
`\voiceOneStyle` ... 182
`\voices` ... 181, 859
`\voiceThree` ... 179
`\voiceThreeStyle` ... 182
`\voiceTwo` ... 179
`\voiceTwoStyle` ... 182
`\void` ... 569, 859
voix ... 179
voix dédiée aux sauts ... 591
voix entre deux portées ... 359
voix et ambitus ... 37
voix multiples ... 183
voix multiples et altérations ... 32, 34
voix, arpège, enjambement ... 153
voix, `\autoBeamOff` et `\partCombine` ... 89
voix, citation ... 224
voix, décalage ... 183
voix, division ... 331
voix, octaviation d'une seule ... 26
voix, polyphonie, additionnelle ... 186
voix, réplication ... 224
voix, style ... 182
volta ... 165
volta ... 158, 159
`volta` ... 158, 159, 859
volta, prima ... 159
volta, seconda ... 159
volta, ultima ... 159
Volta_engraver ... 453
`Volta_engraver` ... 163
VoltaBracket ... 165, 167
VoltaRepeatedMusic ... 165, 167
vowel transition ... 300
VowelTransition ... 300
voyelle, transition ... 299
`\vspace` ... 270, 769

W

Walker shape, tête de note ... 43
`\walkerHeads` ... 43
`\walkerHeadsMinor` ... 44
`whichBar` ... 109
whistle ... 807
`\whiteout` ... 799
whiteout (blanchiment) ... 676
`\whiteTriangleMarkup` ... 454
Winds ... 432, 434, 435, 436, 444
`\with` ... 637, 641
`\with-color` ... 242, 800
`\with-dimensions` ... 800
`\with-dimensions-from` ... 800
`\with-link` ... 801
`\with-outline` ... 801
`\with-url` ... 779
`\withMusicProperty` ... 859
woodblock ... 807
`\woodwind-diagram` ... 788
`\wordwrap` ... 272, 770
`\wordwrap-field` ... 769
`\wordwrap-internal` ... 803
`\wordwrap-lines` ... 278, 803
`\wordwrap-string` ... 770
`\wordwrap-string-internal` ... 803
World music ... 501, 502, 504, 505

X

x11, couleur 243, 244
`x11-color` 243, 244
`X-offset` 596
`\xNote` 40, 859
`\xNotesOff` 40
`\xNotesOn` 40