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 2.13.39. 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 `http://www.lilypond.org/`.

Copyright © 1999–2010 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 2.13.39

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 5
Nom des notes dans d'autres langues 7
1.1.2 Modification de plusieurs hauteurs 8
Vérifications d'octave 8
Transposition 9
1.1.3 Gravure des hauteurs 12
Clefs 12
Armure 16
Marques d'octaviation 17
Instruments transpositeurs 18
Altérations accidentelles automatiques 20
Ambitus 27
1.1.4 Têtes de note 29
Têtes de note spécifiques 29
Têtes de note avec nom de note 31
Têtes de note à forme variable 33
Improvisation 35
1.2 Rythme 36
1.2.1 Écriture du rythme 36
Durées 37
Nolets 38
Changement d'échelle des durées 43
Liaisons de prolongation 44
1.2.2 Écriture des silences 47
Silences 47
Silences invisibles 49
Silences valant une mesure 50
1.2.3 Gravure du rythme 54
Métrique 54
Indication métronomique 58
Levées 61
Musique sans métrique 61
Notation polymétrique 63
Découpage automatique des notes 66
Gravure de lignes rythmiques 66
1.2.4 Barres de ligature 69
Barres de ligature automatiques 69
Définition des règles de ligature automatique 71
Barres de ligature manuelles 77
Liens de croches en soufflet 80
1.2.5 Mesures 81
Barres de mesure 81
Numéros de mesure 86

Vérification des limites et numéros de mesure . . . 89
Indications de repère . . . 90
1.2.6 Fonctionnalités rythmiques particulières . . . 92
Notes d'ornement . . . 92
Alignement et cadences . . . 97
Gestion du temps . . . 97
1.3 Signes d'interprétation . . . 98
1.3.1 Signes d'interprétation attachés à des notes . . . 99
Articulations et ornements . . . 99
Nuances . . . 101
Personnalisation des indications de nuance . . . 106
1.3.2 Signes d'interprétation sous forme de courbe . . . 108
Liaisons d'articulation . . . 108
Liaisons de phrasé . . . 110
Signes de respiration . . . 112
Chutes et sauts . . . 113
1.3.3 Signes d'interprétation sous forme de ligne . . . 114
Glissando . . . 114
Arpèges . . . 115
Trilles . . . 118
1.4 Répétitions et reprises . . . 120
1.4.1 Répétition d'un long passage . . . 120
Répétitions courantes . . . 120
Indications de reprise manuelles . . . 125
Répétitions explicites . . . 127
1.4.2 Autres types de répétition . . . 129
Répétitions de mesure . . . 129
Répétitions en trémolo . . . 131
1.5 Notes simultanées . . . 133
1.5.1 Monophonie . . . 133
Notes en accords . . . 133
Répétition d'accords . . . 134
Expressions simultanées . . . 135
Clusters . . . 136
1.5.2 Plusieurs voix . . . 136
Polyphonie sur une portée . . . 137
Styles de voix . . . 139
Résolution des collisions . . . 140
Regroupement automatique de parties . . . 143
Saisie de musique en parallèle . . . 147
1.6 Notation sur la portée . . . 149
1.6.1 Gravure des portées . . . 149
Initialisation de nouvelles portées . . . 149
Regroupement de portées . . . 151
Imbrication de regroupements de portées . . . 155
Séparation des systèmes . . . 156
1.6.2 Modification de portées individuelles . . . 157
Symbole de la portée . . . 157
Portées d'ossia . . . 160
Masquage de portées . . . 163
1.6.3 Écriture de parties séparées . . . 167
Noms d'instrument . . . 167
Citation d'autres voix . . . 170
Mise en forme d'une citation . . . 174

1.7 Annotations éditoriales ... 177
1.7.1 Dans la portée ... 177
Indication de la taille de fonte musicale ... 177
Doigtés ... 179
Dictée à trous ... 181
Coloration d'objets ... 182
Parenthèses ... 183
Hampes ... 184
1.7.2 Hors de la portée ... 184
Info-bulle ... 185
Quadrillage temporel ... 186
Crochets d'analyse ... 188
1.8 Texte ... 189
1.8.1 Ajout de texte ... 189
Commentaires textuels ... 189
Indication textuelle avec extension ... 191
Indications textuelles ... 193
Texte indépendant ... 197
1.8.2 Mise en forme du texte ... 198
Introduction au formatage de texte ... 198
Sélection de la fonte et de la taille ... 200
Alignement du texte ... 202
Éléments graphiques dans du texte formaté ... 206
Notation musicale dans du texte formaté ... 208
Texte avec sauts de page ... 211
1.8.3 Fontes ... 211
Tout savoir sur les fontes ... 212
Attribution d'une fonte en particulier ... 213
Choix des fontes par défaut ... 214

2 Notation spécialisée ... 215

2.1 Musique vocale ... 215
2.1.1 Vue d'ensemble de la musique vocale ... 215
Références en matière de musique vocale ... 215
Opéra ... 215
Cahier de chants ... 215
Musique parlée ... 216
Chants ... 216
Musique vocale ancienne ... 216
2.1.2 Saisie des paroles ... 216
Tout savoir sur les paroles ... 216
Écriture de chants simples ... 218
Travail avec des paroles et variables ... 219
2.1.3 Alignement des paroles sur une mélodie ... 219
Durée automatique des syllabes ... 220
Durée explicite des syllabes ... 221
Plusieurs syllabes sur une note ... 221
Plusieurs notes pour une même syllabe ... 222
Saut de notes ... 223
Traits d'union et de prolongation ... 224
Paroles et reprises ... 224
2.1.4 Situations particulères en matière de paroles ... 224
Paroles alternatives ... 224
Paroles indépendantes des notes ... 225

Espacement des syllabes . . . 226
Centrage des paroles entre les portées . . . 227
2.1.5 Couplets . . . 227
Numérotation des couplets . . . 227
Indication de nuance dans les couplets . . . 228
Indication du personnage et couplets . . . 228
Rythme différent selon le couplet . . . 229
Paroles en fin de partition . . . 230
Paroles sur plusieurs colonnes en fin de partition . . . 231
2.2 Instruments utilisant des portées multiples . . . 233
2.2.1 Vue d'ensemble des claviers . . . 233
Généralités sur les instruments à clavier . . . 233
Changement de portée manuel . . . 234
Changement de portée automatique . . . 235
Lignes de changement de portée . . . 237
Hampes et changements de portée . . . 237
2.2.2 Piano . . . 239
Pédales de piano . . . 239
2.2.3 Accordéon . . . 240
Symboles de jeux . . . 240
2.2.4 Harpe . . . 244
Généralités sur la harpe . . . 244
Pédales de harpe . . . 244
2.3 Cordes non frettées . . . 245
2.3.1 Vue d'ensemble de la notation pour cordes non frettées . . . 245
Références en matière de cordes non frettées . . . 245
Indications d'archet . . . 246
Harmoniques . . . 246
Snap (Bartók) pizzicato . . . 247
2.4 Fretted string instruments . . . 248
2.4.1 Common notation for fretted strings . . . 248
References for fretted strings . . . 248
String number indications . . . 249
Default tablatures . . . 250
Custom tablatures . . . 257
Fret diagram markups . . . 259
Predefined fret diagrams . . . 268
Automatic fret diagrams . . . 277
Right-hand fingerings . . . 280
2.4.2 Guitar . . . 282
Indicating position and barring . . . 282
Indicating harmonics and dampened notes . . . 282
Indicating power chords . . . 284
2.4.3 Banjo . . . 285
Banjo tablatures . . . 285
2.5 Percussion . . . 286
2.5.1 Common notation for percussion . . . 286
References for percussion . . . 286
Basic percussion notation . . . 286
Drum rolls . . . 287
Pitched percussion . . . 288
Percussion staves . . . 288
Custom percussion staves . . . 290
Ghost notes . . . 294

2.6 Wind instruments ... 294
2.6.1 Common notation for wind instruments ... 294
References for wind instruments ... 295
Fingerings ... 296
2.6.2 Bagpipes ... 298
Bagpipe definitions ... 298
Bagpipe example ... 298
2.6.3 Woodwinds ... 300
2.6.3.1 Woodwind diagrams ... 300
2.7 Chord notation ... 307
2.7.1 Chord mode ... 308
Chord mode overview ... 308
Common chords ... 309
Extended and altered chords ... 310
2.7.2 Displaying chords ... 313
Printing chord names ... 313
Customizing chord names ... 315
2.7.3 Figured bass ... 320
Introduction to figured bass ... 320
Entering figured bass ... 321
Displaying figured bass ... 323
2.8 Ancient notation ... 326
2.8.1 Overview of the supported styles ... 327
2.8.2 Ancient notation—common features ... 328
Pre-defined contexts ... 328
Ligatures ... 328
Custodes ... 329
Figured bass support ... 329
2.8.3 Typesetting mensural music ... 329
Mensural contexts ... 329
Mensural clefs ... 330
Mensural time signatures ... 331
Mensural note heads ... 332
Mensural flags ... 332
Mensural rests ... 333
Mensural accidentals and key signatures ... 334
Annotational accidentals (*musica ficta*) ... 334
White mensural ligatures ... 335
2.8.4 Typesetting Gregorian chant ... 336
Gregorian chant contexts ... 336
Gregorian clefs ... 337
Gregorian accidentals and key signatures ... 337
Divisiones ... 338
Gregorian articulation signs ... 338
Augmentum dots (*morae*) ... 339
Gregorian square neume ligatures ... 340
2.8.5 Working with ancient music—scenarios and solutions ... 346
Incipits ... 347
Mensurstriche layout ... 347
Transcribing Gregorian chant ... 347
Ancient and modern from one source ... 350
Editorial markings ... 350
2.9 World music ... 350
2.9.1 Common notation for non-Western music ... 350

Extending notation and tuning systems 351
2.9.2 Arabic music 352
References for Arabic music 352
Arabic note names 352
Arabic key signatures 353
Arabic time signatures 354
Arabic music example 356
Further reading for Arabic music 356
2.9.3 Turkish classical music 357
References for Turkish classical music 357
Turkish note names 357

3 General input and output 359

3.1 Input structure 359
3.1.1 Structure of a score 359
3.1.2 Multiple scores in a book 360
3.1.3 Multiple output files from one input file 361
3.1.4 Output file names 362
3.1.5 File structure 363
3.2 Titles and headers 364
3.2.1 Creating titles 365
3.2.2 Custom titles 368
3.2.3 Reference to page numbers 369
3.2.4 Table of contents 370
3.3 Working with input files 371
3.3.1 Including LilyPond files 371
3.3.2 Different editions from one source 372
Using variables 373
Using tags 374
3.3.3 Text encoding 376
3.3.4 Displaying LilyPond notation 378
3.4 Controlling output 378
3.4.1 Extracting fragments of music 378
3.4.2 Skipping corrected music 379
3.4.3 Alternative output formats 379
3.4.4 Replacing the notation font 379
3.5 MIDI output 380
3.5.1 Creating MIDI files 380
Instrument names 381
3.5.2 MIDI block 382
3.5.3 What goes into the MIDI output? 383
Supported in MIDI 383
Unsupported in MIDI 383
3.5.4 Repeats in MIDI 384
3.5.5 Controlling MIDI dynamics 384
Dynamic marks 384
Overall MIDI volume 385
Equalizing different instruments (i) 386
Equalizing different instruments (ii) 387
3.5.6 Percussion in MIDI 388

4 Spacing issues 390

4.1 Paper and pages 390
4.1.1 Paper size 390
4.1.2 Page formatting 391
Fixed vertical dimensions 391
Flexible vertical dimensions 391
Horizontal dimensions 394
Other layout variables 395
4.2 Music layout 398
4.2.1 Setting the staff size 398
4.2.2 Score layout 399
4.3 Breaks 399
4.3.1 Line breaking 399
4.3.2 Page breaking 401
4.3.3 Optimal page breaking 402
4.3.4 Optimal page turning 402
4.3.5 Minimal page breaking 403
4.3.6 Explicit breaks 403
4.3.7 Using an extra voice for breaks 405
4.4 Vertical spacing 407
4.4.1 Flexible vertical spacing within systems 407
Within-system spacing properties 407
Spacing of ungrouped staves 410
Spacing of grouped staves 411
Spacing of non-staff lines 412
4.4.2 Explicit staff and system positioning 413
4.4.3 Vertical collision avoidance 418
4.5 Horizontal spacing 419
4.5.1 Horizontal spacing overview 419
4.5.2 New spacing area 421
4.5.3 Changing horizontal spacing 421
4.5.4 Line length 424
4.5.5 Proportional notation 424
4.6 Fitting music onto fewer pages 431
4.6.1 Displaying spacing 431
4.6.2 Changing spacing 433

5 Changing defaults 435

5.1 Interpretation contexts 435
5.1.1 Contexts explained 435
Score - the master of all contexts 435
Top-level contexts - staff containers 435
Intermediate-level contexts - staves 436
Bottom-level contexts - voices 436
5.1.2 Creating contexts 437
5.1.3 Keeping contexts alive 438
5.1.4 Modifying context plug-ins 440
5.1.5 Changing context default settings 442
5.1.6 Defining new contexts 443
5.1.7 Aligning contexts 445
5.2 Explaining the Internals Reference 446
5.2.1 Navigating the program reference 446
5.2.2 Layout interfaces 447

5.2.3 Determining the grob property ... 448
5.2.4 Naming conventions ... 449
5.3 Modifying properties ... 449
5.3.1 Overview of modifying properties ... 449
5.3.2 The `\set` command ... 450
5.3.3 The `\override` command ... 452
5.3.4 The `\tweak` command ... 453
5.3.5 `\set` vs. `\override` ... 455
5.3.6 Modifying alists ... 455
5.4 Useful concepts and properties ... 457
5.4.1 Input modes ... 457
5.4.2 Direction and placement ... 458
5.4.3 Context layout order ... 460
5.4.4 Distances and measurements ... 460
5.4.5 Staff symbol properties ... 460
5.4.6 Spanners ... 461
Using the `spanner-interface` ... 461
Using the `line-spanner-interface` ... 463
5.4.7 Visibility of objects ... 466
Removing the stencil ... 466
Making objects transparent ... 466
Painting objects white ... 466
Using break-visibility ... 467
Special considerations ... 469
5.4.8 Line styles ... 470
5.4.9 Rotating objects ... 471
Rotating layout objects ... 471
Rotating markup ... 472
5.5 Advanced tweaks ... 472
5.5.1 Aligning objects ... 472
Setting `X-offset` and `Y-offset` directly ... 473
Using the `side-position-interface` ... 473
Using the `self-alignment-interface` ... 474
Using the `break-alignable-interface` ... 475
5.5.2 Vertical grouping of grobs ... 477
5.5.3 Modifying stencils ... 477
5.5.4 Modifying shapes ... 478
Modifying ties and slurs ... 478
5.6 Using music functions ... 479
5.6.1 Substitution function syntax ... 479
5.6.2 Substitution function examples ... 480

Annexe A Notation manual tables ... 482

A.1 Chord name chart ... 482
A.2 Common chord modifiers ... 483
A.3 Predefined fretboard diagrams ... 486
A.4 MIDI instruments ... 494
A.5 List of colors ... 495
A.6 The Feta font ... 496
Clef glyphs ... 496
Time Signature glyphs ... 496
Number glyphs ... 497
Accidental glyphs ... 497
Default Notehead glyphs ... 498

Special Notehead glyphs ... 498
Shape-note Notehead glyphs ... 499
Rest glyphs ... 503
Flag glyphs ... 504
Dot glyphs ... 504
Dynamic glyphs ... 504
Script glyphs ... 505
Arrowhead glyphs ... 507
Bracket-tip glyphs ... 507
Pedal glyphs ... 507
Accordion glyphs ... 508
Vaticana glyphs ... 508
Medicaea glyphs ... 509
Hufnagel glyphs ... 510
Mensural glyphs ... 510
Neomensural glyphs ... 512
Petrucci glyphs ... 513
Solesmes glyphs ... 514
A.7 Note head styles ... 514
A.8 Text markup commands ... 515
A.8.1 Font ... 515
A.8.2 Align ... 524
A.8.3 Graphic ... 537
A.8.4 Music ... 543
A.8.5 Instrument Specific Markup ... 546
A.8.6 Other ... 550
A.9 Text markup list commands ... 554
A.10 List of articulations ... 555
Articulation scripts ... 555
Ornament scripts ... 556
Fermata scripts ... 556
Instrument-specific scripts ... 556
Repeat sign scripts ... 556
Ancient scripts ... 556
A.11 Percussion notes ... 557
A.12 Technical glossary ... 558
alist ... 558
callback ... 558
closure ... 558
glyph ... 558
grob ... 559
immutable ... 559
interface ... 559
lexer ... 559
mutable ... 559
output-def ... 560
parser ... 560
parser variable ... 560
prob ... 560
simple closure ... 560
smob ... 561
stencil ... 561
A.13 All context properties ... 561
A.14 Layout properties ... 571

A.15 Available music functions 587
A.16 Predefined type predicates 591
R5RS primary predicates 591
R5RS secondary predicates 591
Guile predicates 592
LilyPond scheme predicates 592
LilyPond exported predicates 592
A.17 Scheme functions 593

Annexe B Cheat sheet 614

Annexe C Grammaire de LilyPond 618

Annexe D GNU Free Documentation License 637

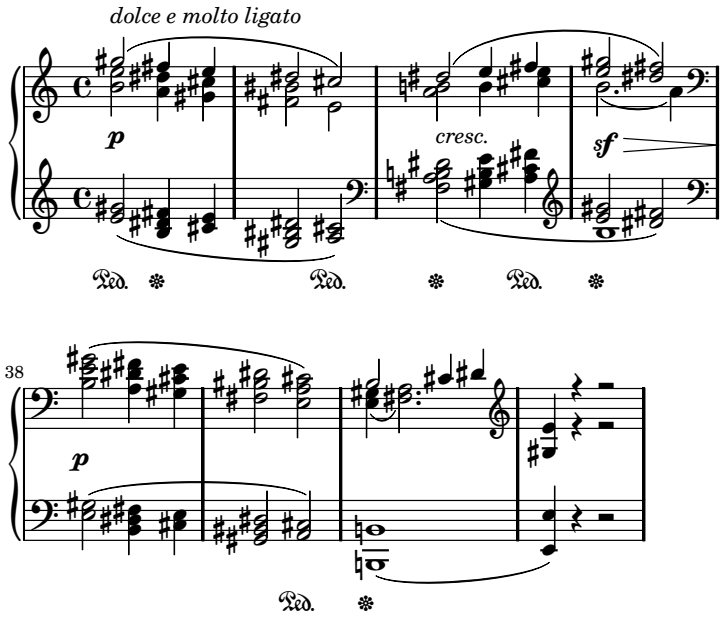
Annexe E Index des commandes LilyPond 644

Annexe F Index de LilyPond 653

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.
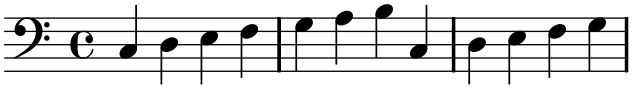
1.1.1 Écriture des hauteurs de note

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.
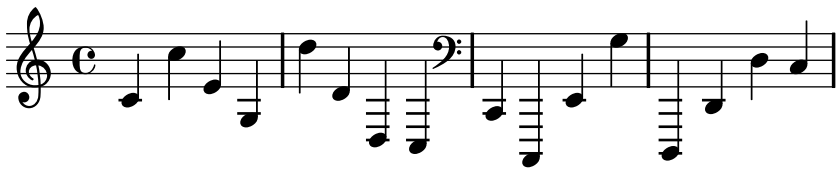
```
\clef bass
c d e f
g a b c
d e f g
```



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' c'' e' g
d'' d' d c
\clef bass
c, c,, e, g
```

```
d,, d, d c
```



Voir aussi

Glossaire musicologique : Section “Noms des notes” dans *Glossaire*.

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

Octaves relatives

On spécifie les octaves en ajoutant `'` ou `,` aux noms de hauteur. En recopiant de la musique, on a vite fait de mettre une note à la mauvaise octave ; ce genre d'erreur est difficile à retrouver. Le mode d'écriture en octave relative prévient ces erreurs dans la mesure où elles deviennent beaucoup plus évidentes : une seule erreur décale le reste de la pièce à une mauvaise octave.
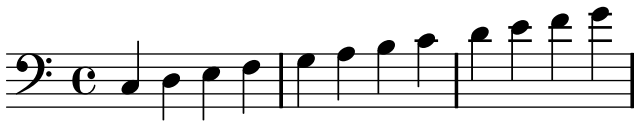
`\relative` *`hauteur_de_référence expression_musicale`*

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.
- Lorsque l'on entre en mode `\relative`, une hauteur absolue de départ peut être spécifiée, et agira dès lors comme si elle précédait la première note de *expression_musicale*. Nous vous recommandons de faire référence à un do (`c`).
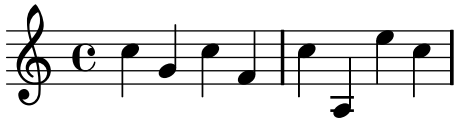
Voici le mode `\relative` en action.

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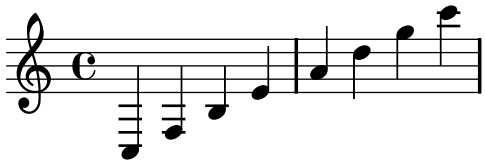


On utilise les signes de changement d'octave pour les intervalles dépassant la quarte.

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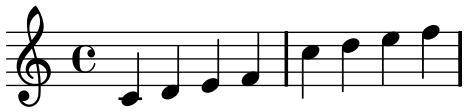


Bien que ne comportant aucun signe de changement d'octave, une séquence de notes peut tout à fait couvrir un intervalle important.

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



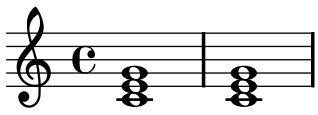
Lorsque plusieurs blocs `\relative` sont imbriqués, c'est la dernière clause `\relative` qui est prise en compte.

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



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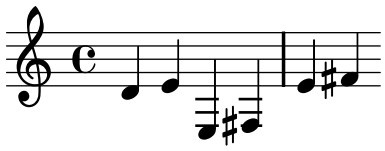
```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` n'est pas permis au sein d'un bloc `\chordmode`.

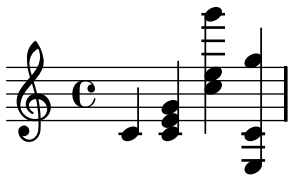
Pour utiliser le mode d'octave relative dans de la musique transposée, une clause `\relative` additionnelle doit être placée au sein du bloc `\transpose`.

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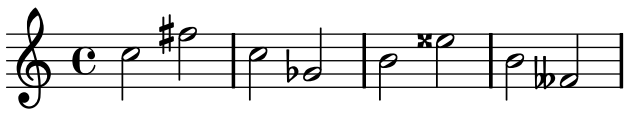


Si l'expression précédente est un accord, c'est la première note de l'accord qui détermine l'emplacement de la première note du prochain accord. À l'intérieur de l'accord, les notes sont placées relativement à celle qui précède. Examinez avec attention l'exemple suivant, et tout particulièrement le positionnement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

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



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 8.

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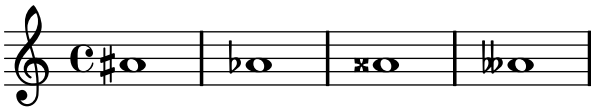
Référence des propriétés internes : Section "RelativeOctaveMusic" dans *Référence des propriétés internes*.

Altérations

> **Note :** Les nouveaux utilisateurs sont parfois déroutés par la gestion des altérations et de l'armure. Pour LilyPond, 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 "Altérations et armure" dans *Manuel d'initiation*.
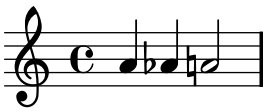
Dans la notation par défaut, un *dièse* est formé en ajoutant `is` après le nom de note, un *bémol* en ajoutant `es`. Les *double-dièses* et *double-bémols* sont obtenus en ajoutant respectivement `isis` ou `eses` au nom de note. Ce sont les noms de note hollandais. Pour les autres langues, consultez [Nom des notes dans d'autres langues], page 7.

```
ais1 aes aisis aeses
```



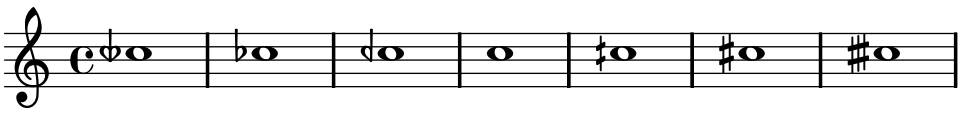
Un bécarre annule l'effet d'une altération, qu'elle soit accidentelle ou à l'armure. Cependant, dans la syntaxe des noms de note, les bécarres ne s'écrivent pas avec un suffixe ; un simple nom de note donnera une note bécarre.

```
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 :
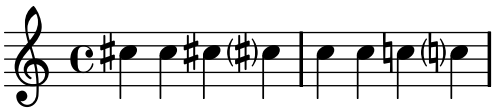
```
ceseh1 ces ceh c cih cis cisih
```



Les micro-intervalles sont aussi exportés dans le fichier MIDI.
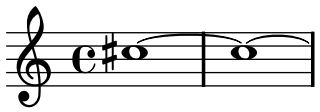
Normalement, les altérations sont imprimées automatiquement, mais il se peut que vous vouliez les imprimer manuellement. On peut forcer l'impression d'une altération, dite « de précaution », en ajoutant un point d'exclamation `!` après la hauteur de note. Une altération entre parenthèses peut être obtenue en ajoutant un point d'interrogation `?` après la hauteur de note. Ces signes peuvent aussi être utilisés pour imprimer des bécarres.
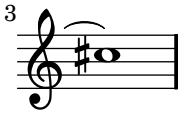
```
cis cis cis! cis? c c c! c?
```



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

```
cis1 ~ cis ~
\break
cis
```



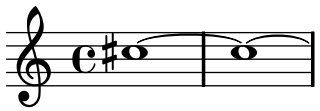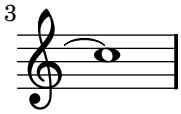


Morceaux choisis

*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
}
```
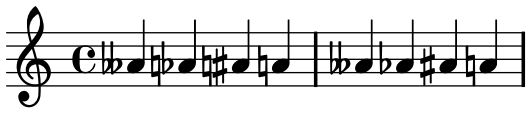




Suppression des bécarres superflus

En accord avec les règles standards de l'écriture musicale, on grave un bécarre avant un dièse ou un bémol si on a besoin d'annuler une altération précédente. Pour modifier ce comportement, assignez la propriété `extraNatural` du contexte `Staff` à 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 “Altérations et armure” dans *Manuel d’initiation*.

Manuel de notation : [Altérations accidentelles automatiques], page 20, ⟨undefined⟩ [Altérations accidentelles suggérées], page ⟨undefined⟩, [Nom des notes dans d’autres langues], page 7.

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 pour plusieurs autres langues. Pour les utiliser, incluez le fichier d’initialisation spécifique à la langue voulue. Par exemple, pour les langues romanes, ajoutez `\include "italiano.ly"` au début du fichier source.

Les fichiers de langue disponibles ainsi que les noms de note utilisés sont les suivants :

Fichier linguistique	**Nom des notes**
‘`nederlands.ly`’	c d e f g a bes b
‘`arabic.ly`’	do re mi fa sol la sib si
‘`catalan.ly`’	do re mi fa sol la sib si
‘`deutsch.ly`’	c d e f g a b h
‘`english.ly`’	c d e f g a bf b
‘`espanol.ly`’	do re mi fa sol la sib si
‘`italiano.ly`’	do re mi fa sol la sib si
‘`norsk.ly`’	c d e f g a b h
‘`portugues.ly`’	do re mi fa sol la sib si
‘`suomi.ly`’	c d e f g a b h
‘`svenska.ly`’	c d e f g a b h
‘`vlaams.ly`’	do re mi fa sol la sib si
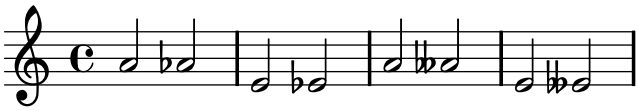
et les suffixes d’altération correspondants :

Fichier linguistique	**dièse**	**bémol**	**double dièse**	**double bémol**
‘`nederlands.ly`’	-is	-es	-isis	-eses
‘`arabic.ly`’	-d	-b	-dd	-bb
‘`catalan.ly`’	-d/-s	-b	-dd/-ss	-bb
‘`deutsch.ly`’	-is	-es	-isis	-eses
‘`english.ly`’	-s/-sharp	-f/-flat	-ss/-x/-sharpsharp	-ff/-flatflat
‘`espanol.ly`’	-s	-b	-ss/-x	-bb
‘`italiano.ly`’	-d	-b	-dd	-bb

‘norsk.ly’	-iss/-is	-ess/-es	-ississ/-isis	-essess/-eses
‘portugues.ly’	-s	-b	-ss	-bb
‘suomi.ly’	-is	-es	-isis	-eses
‘svenska.ly’	-iss	-ess	-ississ	-essess
‘vlaams.ly’	-k	-b	-kk	-bb

Notez qu’en hollandais, en allemand, en norvégien et en suédois, un *la* altéré de *bémol* se note `aes` et `aeses`. Ces formes sont contractées en `as` et `ases` (ou plus communément `asas`). Dans certains fichiers linguistiques, seules ces formes abrégées ont été définies.

```
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 plusieurs fichiers linguistiques. Les préfixes *semi-* et *sesqui-* correspondent au *demi-* et *trois demis*. À noter qu’aucune définition n’existe à ce jour pour le norvégien, le suédois, le catalan et l’espagnol.

Fichier linguistique	**semi-dièse**	**semi-bémol**	**sesqui-dièse**	**sesqui-bémol**
‘nederlands.ly’	-ih	-eh	-isih	-eseh
‘arabic.ly’	-sd	-sb	-dsd	-bsb
‘deutsch.ly’	-ih	-eh	-isih	-eseh
‘english.ly’	-qs	-qf	-tqs	-tqf
‘espanol.ly’	-cs	-cb	-tcs	-tcb
‘italiano.ly’	-sd	-sb	-dsd	-bsb
‘portugues.ly’	-sqt	-bqt	-stqt	-btqt

Voir aussi

Glossaire musicologique : Section “Nom des notes” dans *Glossaire*.

Manuel de notation : ⟨undefined⟩ [Noms de note et altérations non-occidentaux], page ⟨undefined⟩.

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

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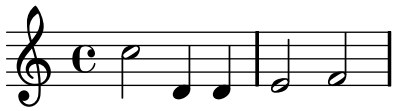
Cette partie traite de la manière de modifier les hauteurs de note.

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 !
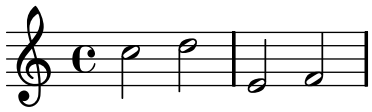
Une note peut être suivie de `=`*apostrophes/virgules* pour indiquer à quelle octave absolue elle devrait être. Dans l’exemple suivant, le `d` générera un avertissement, puisqu’on attend un `d''` mais qu’on obtient un `d'` – il n’y a qu’une tierce entre `b'` et `d''`. Sur la partition, l’octave sera corrigée pour donner un `d''` et la prochaine note sera calculée en fonction de `d''` et non de `d'`.

```
\relative c'' {
  c2 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 doît en mode relatif. Dans le cas contraire, un message sera émis et la note précédente ne sera pas modifiée. Les notes suivantes seront, par contre, positionnées relativement à *hauteur_référence*.
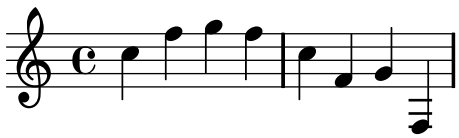
```
\relative c'' {
  c2 d
  \octaveCheck c'
  e2 f
}
```



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

```
\relative c'' {
  c4 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 :
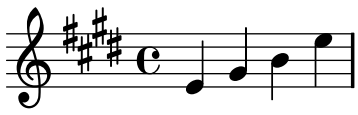
`\transpose` *note_de_départ note_d_arrivée expression_musicale*

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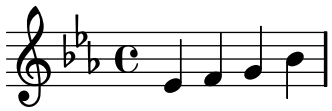
Prenons comme exemple une pièce écrite en ré majeur. Si cette pièce est un peu trop basse pour l'interprète, elle peut être transposée en mi majeur. Vous noterez que l'armure est automatiquement modifiée.

```
\transpose d e {
  \relative c' {
    \key d \major
    d4 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.
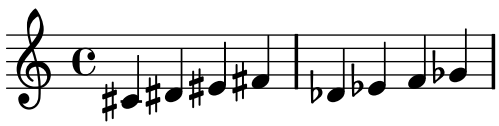
```
\transpose a c' {
  \relative c' {
    \key c \major
    c4 d e g
  }
}
```



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' { 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 18.

Morceaux choisis

Transposition et réduction du nombrer 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 appliquable 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 (eq? n 6) (eq? n 2)))
       (set! a (- a 2))
       (set! n (+ n 1)))
      ((and (< a -1) (or (eq? n 0) (eq? 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))))
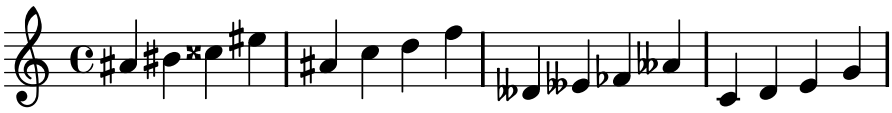
#(define (naturalize music)
   (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
     (if (pair? es)
         (ly:music-set-property!
          music 'elements
          (map (lambda (x) (naturalize x)) es)))
     (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 (parser location 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 : [Octaves relatives], page 2, [Instruments transpositeurs], page 18.

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`.

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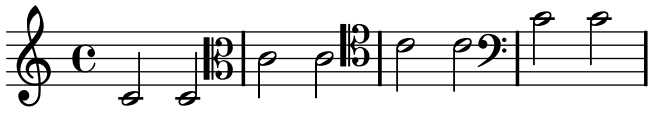
Nous allons voir dans cette partie comment influer sur la gravure des hauteurs.

Clefs

La clé indique quelles lignes de la portée correspondent à quelles hauteurs. Elle peut changer au fil du morceau, comme dans l’exemple suivant qui indique le do médium dans différentes clefs :

```
\clef treble
c2 c
\clef alto
c2 c
```

```
\clef tenor
c2 c
\clef bass
c2 c
```



LilyPond met à votre disposition d'autres clefs :
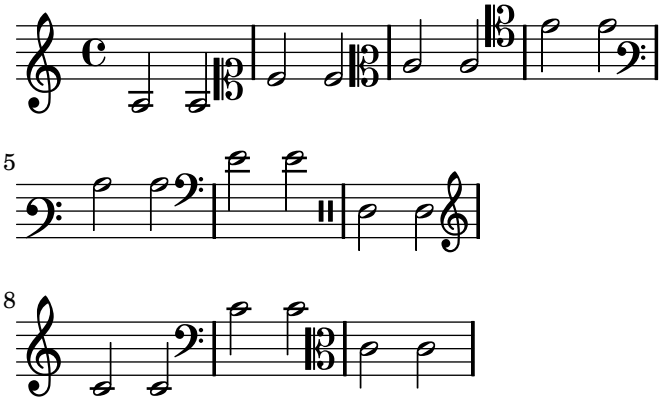
```
\clef french
c2 c
\clef soprano
c2 c
\clef mezzosoprano
c2 c
\clef baritone
c2 c

\break

\clef varbaritone
c2 c
\clef subbass
c2 c
\clef percussion
c2 c
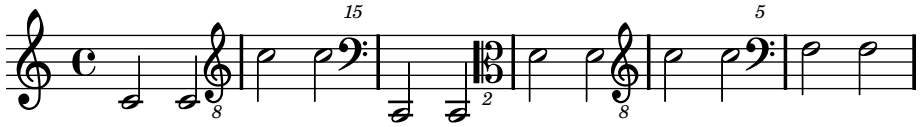
\break

\clef G   % synonym for treble
c2 c
\clef F   % synonym for bass
c2 c
\clef C   % synonym for alto
c2 c
```



En ajoutant `_8` ou `^8` au nom de la clé, 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
c2 c
\clef "treble_8"
c2 c
\clef "bass^15"
c2 c
\clef "alto_2"
c2 c
\clef "G_8"
c2 c
\clef "F^5"
c2 c
```



Il existe d'autres clefs, destinées à un usage particulier. Voir à ce sujet ⟨undefined⟩ [Tablatures par défaut], page ⟨undefined⟩ et ⟨undefined⟩ [Tablatures personnalisées], page ⟨undefined⟩.

Morceaux choisis

Affinage des propriétés d'une clef

La commande `\clef "treble_8"` équivaut à définir `clefGlyph`, `clefPosition` – qui contrôle la position verticale de la clé – `middleCPosition` et `clefOctavation`. Une clé est imprimée lorsque l'une de ces propriétés, hormis `middleCPosition`, est modifiée. Les exemples suivant font apparaître des possibilités de réglage manuel de ces propriétés.

Modifier le glyphe, la position de la clef ou son octaviation ne changera pas la position des notes ; il faut pour y parvenir modifier aussi la position du do médium. Le positionnement est relatif à la ligne médiane, un nombre positif faisant monter, chaque ligne ou interligne comptant pour 1. La valeur de `clefOctavation` devrait être de 7, -7, 15 ou -15, bien que rien ne vous empêche de lui affecter une autre valeur.

Lorsqu'un changement de clef intervient en même temps qu'un saut de ligne, la nouvelle clef est imprimer à la fois en fin de ligne et au début de la suivante. Vous pouvez toujours supprimer cette « clef de précaution » en affectant la valeur `end-of-line-invisible` à la propriété `explicitClefVisibility` du contexte `Staff`. Le comportement par défaut sera réactivé par `\unset Staff.explicitClefVisibility`.

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 sont préservées, ce qui n'est pas le cas pour la deuxième ligne.

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

{
  % The default treble clef
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  \set Staff.middleCPosition = #6
  c'1
  % The baritone clef
```

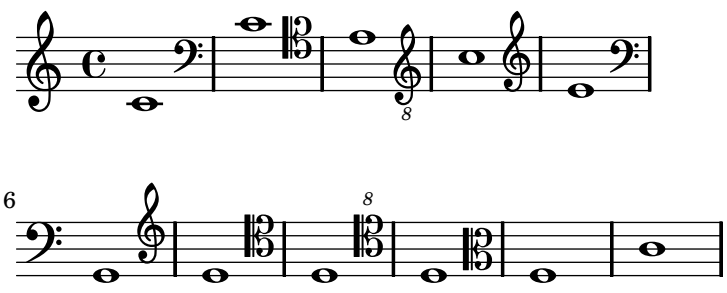
```
  \set Staff.clefGlyph = #"clefs.C"
  \set Staff.clefPosition = #4
  \set Staff.middleCPosition = #4
  c'1
  % The standard choral tenor clef
  \set Staff.clefGlyph = #"clefs.G"
  \set Staff.clefPosition = #-2
  \set Staff.clefOctavation = #-7
  \set Staff.middleCPosition = #1
  c'1
  % A non-standard clef
  \set Staff.clefPosition = #0
  \set Staff.clefOctavation = #0
  \set Staff.middleCPosition = #-4
  c'1 \break

  % The following clef changes do not preserve
  % the normal relationship between notes 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.clefOctavation = #7
  c'1
  \set Staff.clefOctavation = #0
  \set Staff.clefPosition = #0
  c'1

  % Return to the normal clef:

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



Voir aussi

Manuel de notation : ⟨undefined⟩ [Tablatures par défaut], page ⟨undefined⟩, ⟨undefined⟩ [Tablatures personnalisées], page ⟨undefined⟩.

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 “OctavateEight” dans *Référence des propriétés internes*, Section “clef-interface” dans *Référence des propriétés internes*.

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 “Altérations et armure” dans *Manuel d’initiation*.
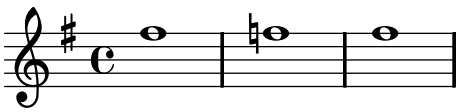
L’armure indique la tonalité dans laquelle la pièce doit être jouée. Elle comprend un ensemble d’altérations (dièses ou bémols) à la clef, c’est-à-dire au début de la portée. Elle peut varier en cours de morceau.

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`.

```
\key g \major
fis1
f
fis
```
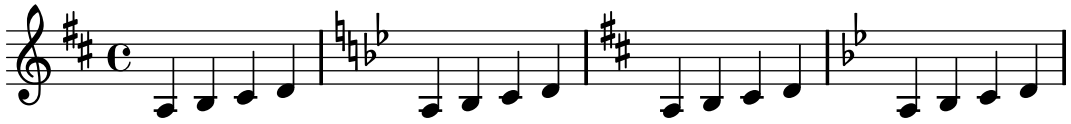


Morceaux choisis

*Suppression des bécarres superflus après un changement de*tonalité

Après un changement de tonalité, un bécarre est imprimé pour annuler toute altération précédente. Ceci peut être supprimé en réglant à `"false"` 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 `keySignature` d'un contexte `Staff`.

Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété. Il s'agit en l'occurence de définir une liste :
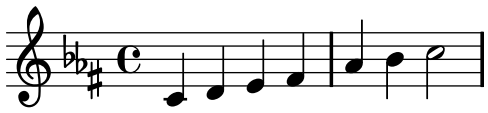
`\set Staff.keySignature = #`(((octave . pas) . altération) ((octave . pas) . altération) ...)`

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 ,FLAT ,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 quel que soit l'octave.

Voici, par exemple, comment générer une gamme par ton :

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Voir aussi

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

Manuel d'initiation : Section “Altérations et armure” dans *Manuel d'initiation*.

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

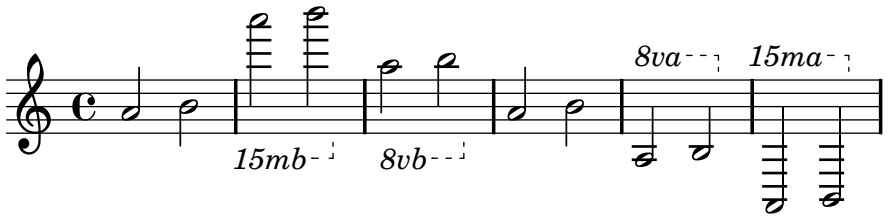
Référence des propriétés internes : Section “KeyChangeEvent” dans *Référence des propriétés internes*, Section “Key_engraver” dans *Référence des propriétés internes*, Section “Key_performer” dans *Référence des propriétés internes*, Section “KeyCancellation” dans *Référence des propriétés internes*,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

```
a2 b
\ottava #-2
a2 b
\ottava #-1
a2 b
\ottava #0
a2 b
```

```
\ottava #1
a2 b
\ottava #2
a2 b
```

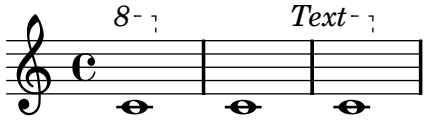


Morceaux choisis

Texte des marques d'octaviation

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

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



Voir aussi

Glossarire musicologique : Section "octaviation" dans *Glossaire*.

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 170.

`\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
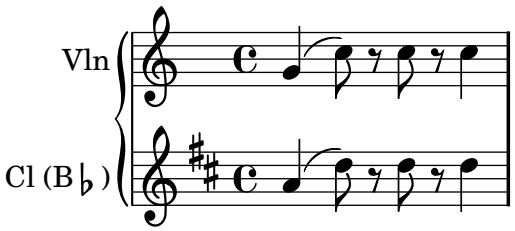
mentionnée en *mode absolu*. Par exemple, lorsque vous saisissez une partition en notes réelles, toutes les voix devraient être en ut ; si un instrument joue un ton au dessus, il faudra lui ajouter un `\transposition d'`. La commande `\transposition` s'utilise **si et seulement si** les notes à saisir **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" {
    \relative c'' {
      \set Staff.instrumentName = #"Vln"
      \set Staff.midiInstrument = #"violin"
      % not strictly necessary, but a good reminder
      \transposition c'

      \key c \major
      g4( c8) r c r c4
    }
  }
  \new Staff = "clarinet" {
    \relative c'' {
      \set Staff.instrumentName = \markup { Cl (B\flat) }
      \set Staff.midiInstrument = #"clarinet"
      \transposition bes

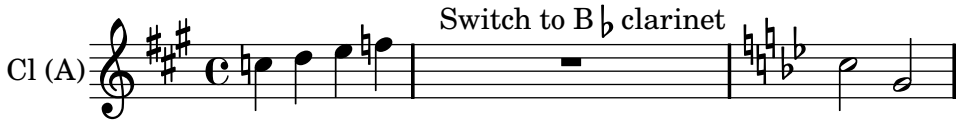
      \key d \major
      a4( d8) r d r d4
    }
  }
>>
```



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

```
\set Staff.instrumentName = #"Cl (A)"
\key a \major
\transposition a
c d e f
\textLengthOn
s1*0^\markup { Switch to B\flat clarinet }
R1

\key bes \major
\transposition bes
c2 g
```



Voir aussi

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

Manuel de notation : [Citation d’autres voix], page 170, [Transposition], page 9.

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 <<
  #(set-accidental-style '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 :

```
#(set-accidental-style 'voice 'StaffGroup)
```

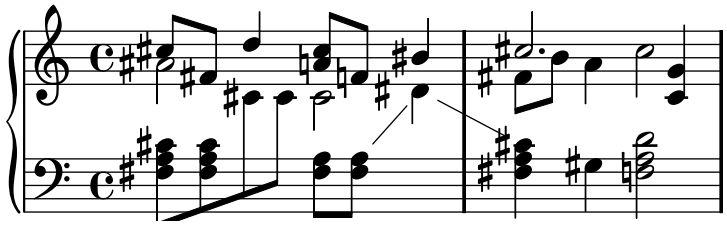
Nous vous présentons ci-après les différentes régles d’altération prises en charge. Pour les besoins de la démonstration, nous partirons de l’exemple suivant :

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

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative c' {
      <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 {
    <<
      \context Staff = "up" {
        #(set-accidental-style 'default)
        \musicA
      }
      \context Staff = "down" {
        #(set-accidental-style 'default)
        \musicB
      }
    >>
  }
```

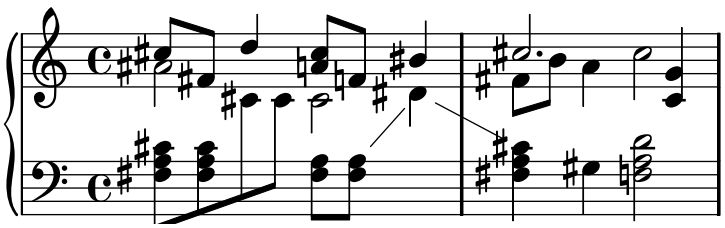


Notez bien que pour appliquer le même style aux deux portées, seules les dernières lignes de cet exemple nous intéressent.

```
  \new PianoStaff {
    <<
      \context Staff = "haut" {
        %%% voici la ligne à modifier en conséquence :
        #(set-accidental-style 'default 'Score)
        \musicA
      }
      \context Staff = "bas" {
        \musicB
      }
    >>
  }
```

`default`

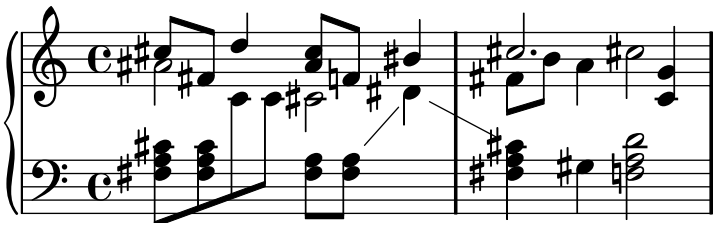
C'est la règle d'impression par défaut, qui se rapporte à l'usage en vigueur au XVIIIe siècle : les altérations accidentelles sont valables tout une mesure, et uniquement à leur propre octave. C'est la raison pour laquelle il n'y a pas de bécarre avant le *si* de la deuxième mesure, ni avant le dernier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



`voice`

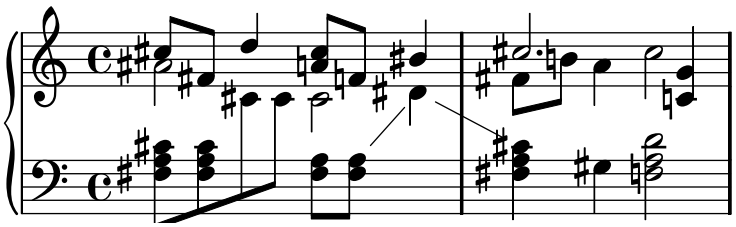
En principe, LilyPond se souvient de toutes les altérations présentes sur la portée (contexte `Staff`). Avec cette règle, cependant, les altérations sont indépendantes pour chacune des voix tout en obéissant à la règle `default`.

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`.



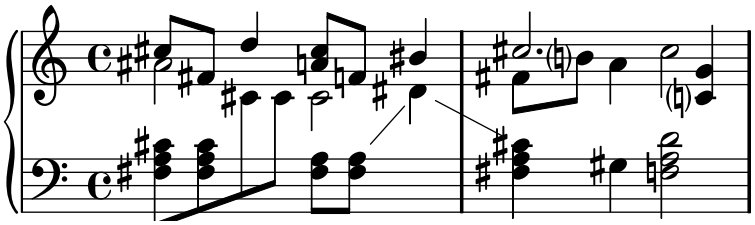
`modern`

Cette règle est la plus courante au XXe siècle. Les altérations accidentelles sont imprimées comme avec le style `default`, mais 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`

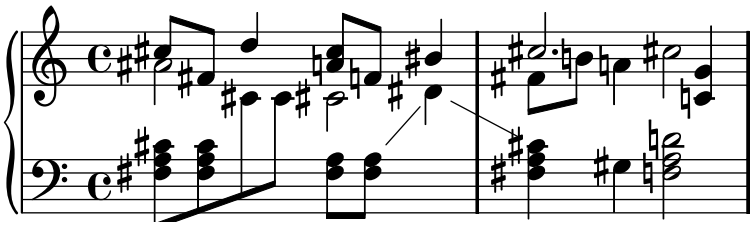
Cette règle est équivalente à `modern`, mais les bécarres de précaution (absents dans la règle `default`) sont imprimés de façon particulière : soit plus petit, soit (par défaut) entre parenthèses – il est possible de le définir au moyen de la propriété `cautionary-style` de l'objet `AccidentalSuggestion`.



`modern-voice`

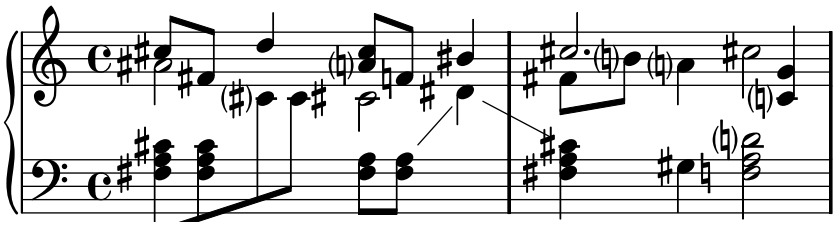
Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix, mais les autres voix d'un même contexte `Staff` en *tiennent compte* cette fois. C'est pourquoi le *la* 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 qu'il avait auparavant concerné la main droite.



`modern-voice-cautionary`

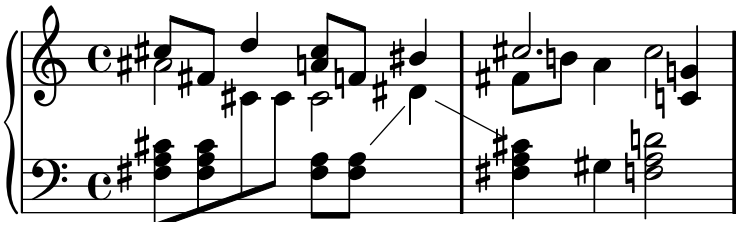
Cette régle est similaire à la précédente, mais les altérations de précaution (celles que n'aurait pas ajoutées `voice`), sont imprimées de façon particulière. On retrouve donc toutes les altérations qu'imprimerait `default`, mais certaines sont considérées comme étant « de précaution ».



`piano`

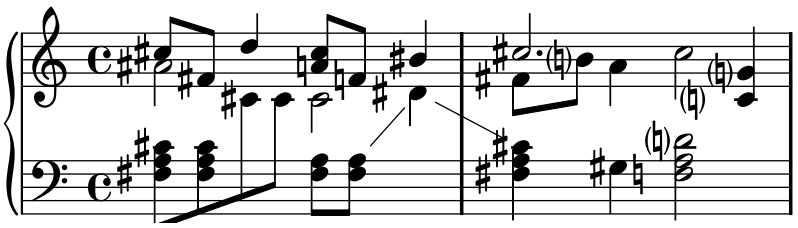
Cette règle est communément employée pour les partitions de piano au XXe siècle. Très similaire à `modern` de par son comportement, elle s'en distingue en ce que les altérations tiennent compte des autre portées du contexte `GrandStaff` ou `PianoStaff`.

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



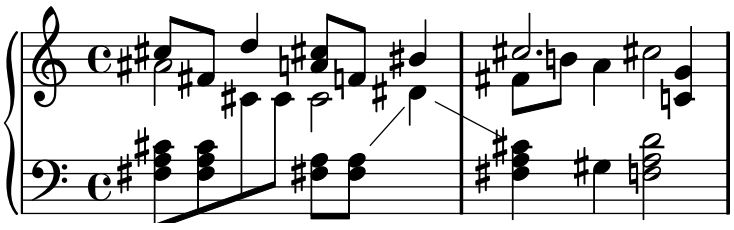
`piano-cautionary`

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

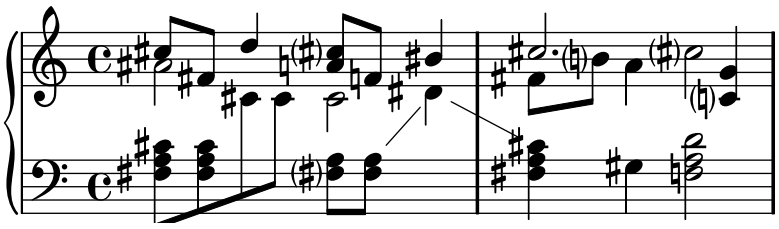


`neo-modern`

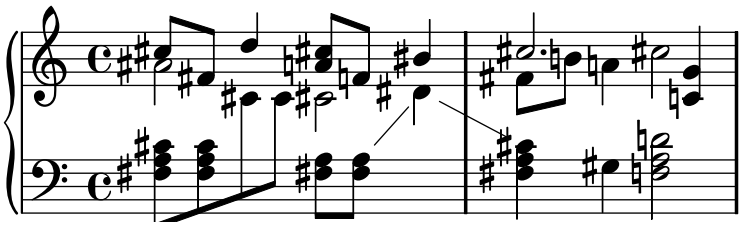
Cette règle suit les pratiques de la musique contemporaine : les altérations accidentelles apparaîssent comme dans le style `modern`, à ceci près qu'elles sont répétées dans la même mesure – sauf si elles concernent deux notes consécutives.



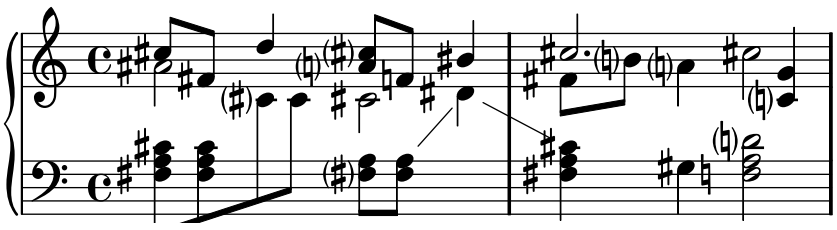
`neo-modern-cautionary`
: Identique au style `neo-modern`, mais les altérations de précaution sont imprimées différemment.



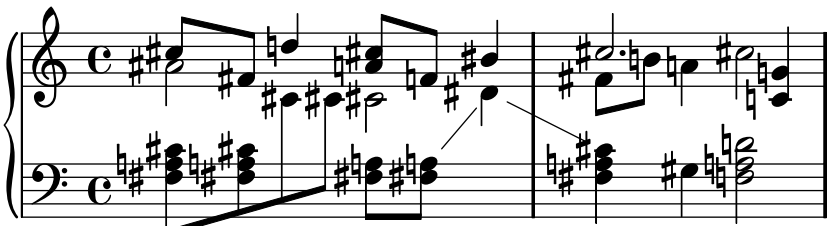
`neo-modern-voice`
: Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix comme avec le style `neo-modern` mais les autres voix dans le même contexte `Staff`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



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

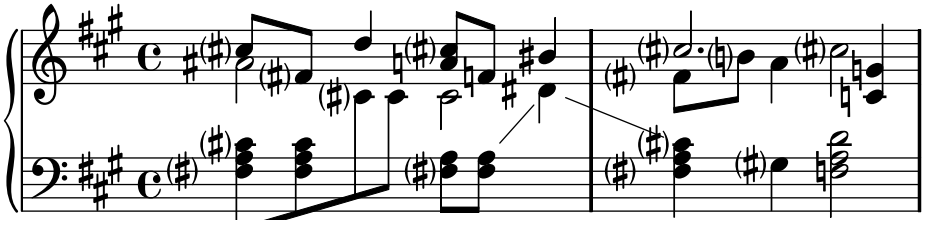


`dodecaphonic`
: Cette règle reproduit ce que certains compositeurs du début du XXe siècle ont introduit dans leur désir d'abolir la distinction entre les notes naturelles ou non. Ainsi, **chaque** note est affublée d'une altération, même si elle est naturelle.



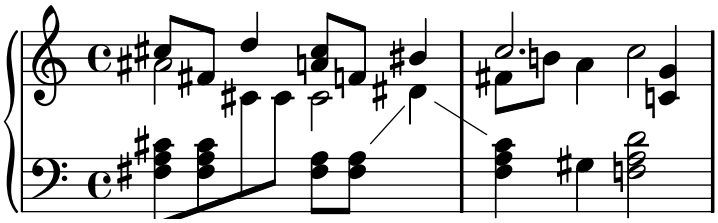
`teaching`

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.



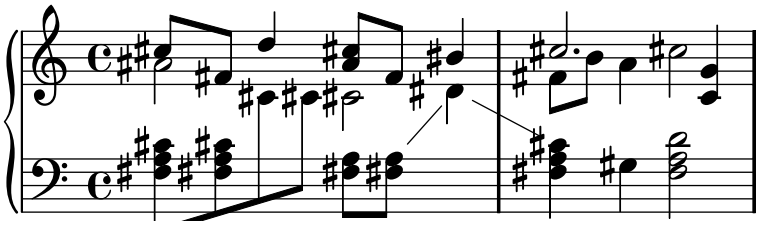
`no-reset`

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, quelque soit leur place dans la mesure, sont de ce fait imprimées en fonction de l'éventuelle armure. À l'inverse du style `dodecaphonic`, aucun bécarre ne sera ici imprimé.



Morceaux choisis

Le dodécaphonisme : toute note est altérée

Au début du XXème siècle, Schoenberg, Berg et Webern – la « Seconde » école de Vienne – imaginèrent de donner une importance comparable aux douze notes de la gamme chromatique, et éviter ainsi toute tonalité. Pour ce faire, ces compositions font apparaître une altération à chaque note, y compris un bécarre, pour mettre en exergue cette nouvelle approche de la théorie et du langage musicaux.

Voici comment obtenir une telle notation.

```
\markup {
  This snippet is deprecated as of version 2.12 and
  will be removed from the documentation in 2.14.
}
```

This snippet is deprecated as of version 2.12 and will be removed from the documentation in 2.14

Voir aussi

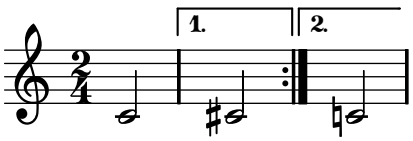
Snippets: Section “Hauteurs” dans *Morceaux choisis*.

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

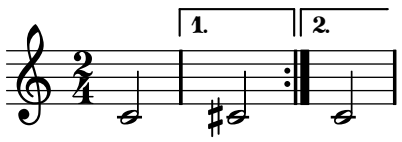
Les notes simultanées sont considérées comme des évènements séquentiels. Ceci implique que, dans un accord, les altérations accidentelles seront imprimées comme si les notes de l’accord apparaissaient l’une après l’autre, en fonction de l’ordre dans lequel elles ont été saisies – ce qui peut poser problème 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.

L’absence d’altération de précaution est déterminée par l’examen de la mesure précédente. Néanmoins, lorsqu’un bloc `\alternative` suit une section `\repeat volta N`, la logique voudrait que l’on regarde 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 (parser location music) (ly:music?) #{
  #(set-accidental-style 'forget)
  $music
  #(set-accidental-style 'modern)
#})
{
  #(set-accidental-style 'modern)
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
     cis'
     \forget c'
  }
}
```
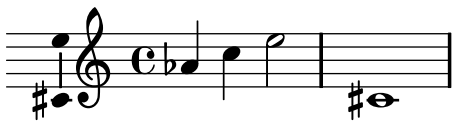


Ambitus

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 clé deux notes représentant la hauteur la plus basse et la plus haute. Les éventuelles altérations accidentelles seront automatiquement ajoutées.

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

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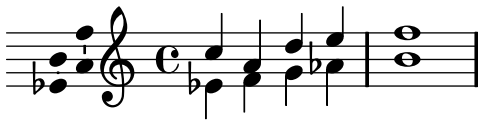


Morceaux choisis

Un ambitus par voix

L'`ambitus` peut être individualisé par voix. Il faut en pareil cas éviter qu'ils se chevauchent.
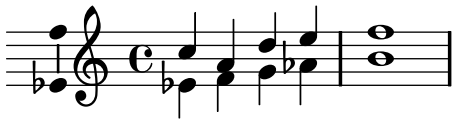
```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus #'X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



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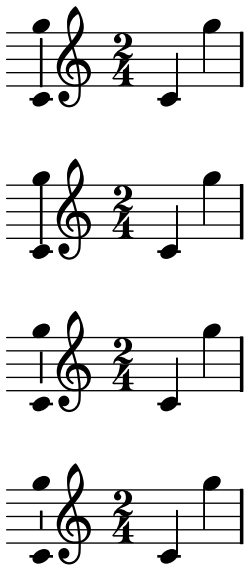
\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}

\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''
}
```



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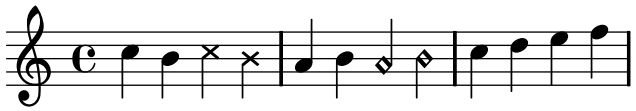
Nous allons voir dans ce chapitre comment modifier l’aspect des têtes de note.

Têtes de note spécifiques

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

```
c4 b
\override NoteHead #'style = #'cross
c4 b
\revert NoteHead #'style
```

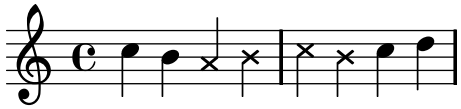
```
a b
\override NoteHead #'style = #'harmonic
a b
\revert NoteHead #'style
c4 d e f
```



Pour une liste exhaustive des styles de tête de note, consultez ⟨undefined⟩ [Styles de tête de note], page ⟨undefined⟩.

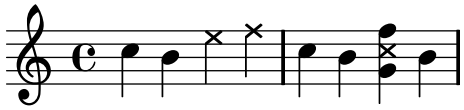
Certains instruments utilisent des têtes de note différentes à des fins spécifiques – des croix (style `cross`) pour le *parlato* des chanteurs ou les notes étouffées des guitares :

```
c4 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 :
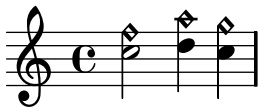
```
c4 b
\xNote { e f }
c b < g \xNote c f > b
```



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 qui indiquent les harmoniques des cordes :

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



Commandes prédéfinies

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

Voir aussi

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

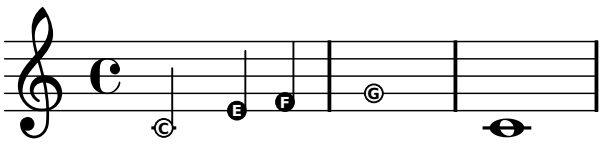
Manuel de notation : ⟨undefined⟩ [Styles de tête de note], page ⟨undefined⟩, [Notes en accords], page 133, ⟨undefined⟩ [Indication des harmoniques et notes étouffées], page ⟨undefined⟩.

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 ⟨undefined⟩ [Définition de la taille de portée], page ⟨undefined⟩.

```
#(set-global-staff-size 26)
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



Commandes prédéfinies

`\easyHeadsOn`, `\easyHeadsOff`.

Morceaux choisis

Numbers as easy note heads

Easy notation note heads use the `note-names` property of the `NoteHead` object to determine what appears inside the note head. By overriding this property, it is possible to print numbers representing the scale-degree.

A simple engraver can be created to do this for every note head object it sees.

```
#(define Ez_numbers_engraver
   (list
    (cons 'acknowledgers
          (list
           (cons 'note-head-interface
                 (lambda (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))))))))

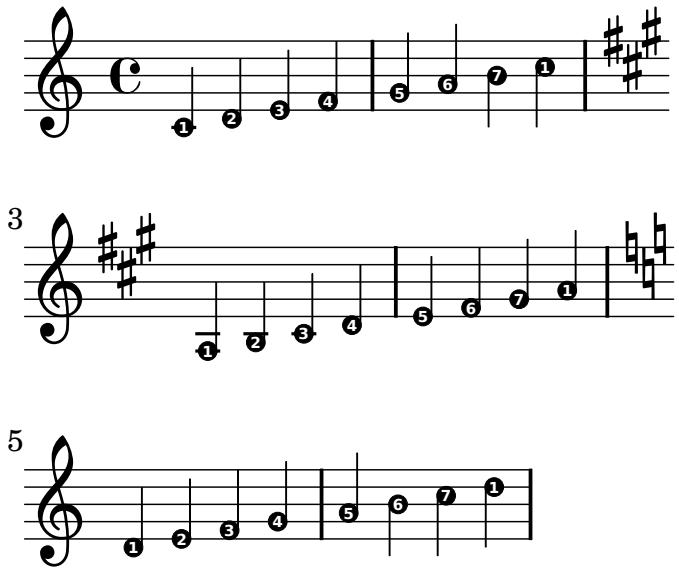
#(set-global-staff-size 26)

\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 : ⟨undefined⟩ [Définition de la taille de portée], page ⟨undefined⟩.
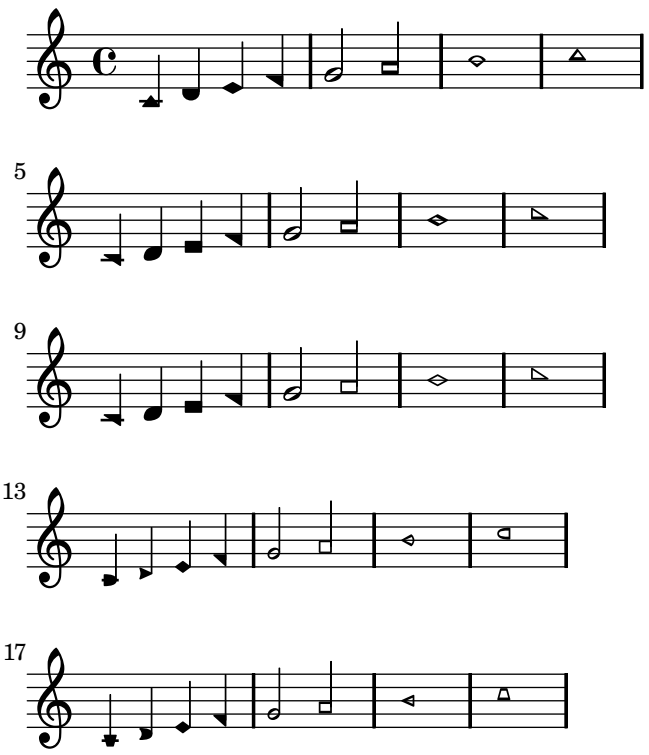
Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

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 XIXe siècle. Voici comment procéder :
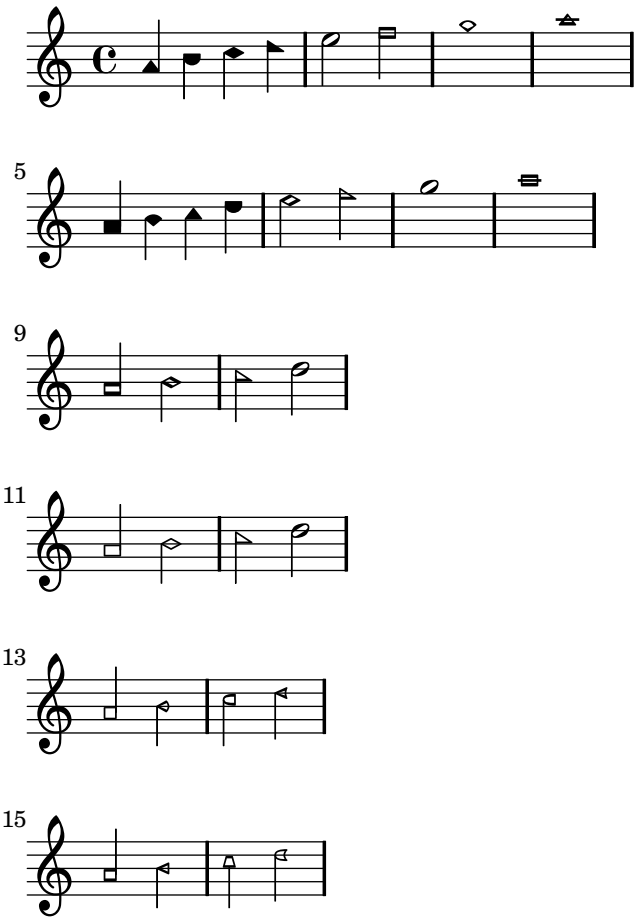
```
\aikenHeads
c, d e f g2 a b1 c \break
\sacredHarpHeads
c,4 d e f g2 a b1 c \break
\southernHarmonyHeads
c,4 d e f g2 a b1 c \break
\funkHeads
c,4 d e f g2 a b1 c \break
\walkerHeads
c,4 d e f g2 a b1 c \break
```



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 :

```
\key a \minor
\aikenHeads
a b c d e2 f g1 a \break
\aikenHeadsMinor
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`, `\funkHeads`, `\funkHeadsMinor`, `\sacredHarpHeads`. `\sacredHarpHeadsMinor`, `\southernHarmonyHeads`, `\southernHarmonyHeadsMinor`, `\walkerHeads`, `\walkerHeadsMinor`.

Morceaux choisis

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 déanciens recueils de chansons américaines.

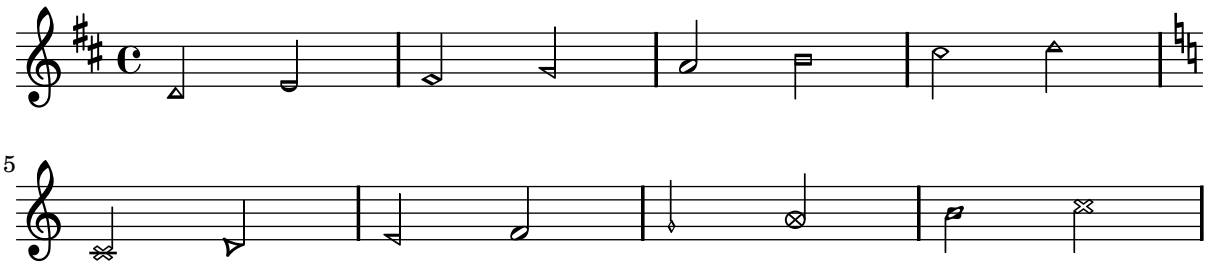
L'exemple suivant montre plusieurs manières de profiler les têtes de note, ainsi que la capacité de trnsposer tout en respectant la fonction harmonique de chaque note dans la gamme.

```
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 ⟨undefined⟩ [Styles de tête de note], page ⟨undefined⟩.

Voir aussi

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

Manuel de notation : ⟨undefined⟩ [Styles de tête de note], page ⟨undefined⟩.

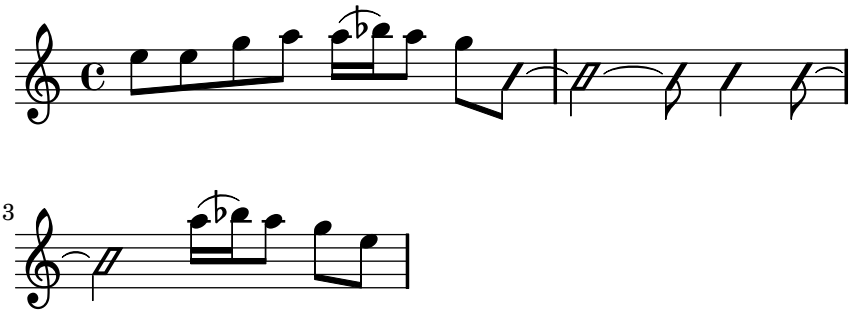
Référence des propriétés internes : Section “note-event” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “NoteHead” dans *Référence des propriétés internes*, Section “note-head-interface” dans *Référence des propriétés internes*.

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"
```

```
} {
  e8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  e2 ~ e8 f4 f8 ~
  f2
  \improvisationOff
  a16( bes) a8 g e
}
```



Commandes prédéfinies
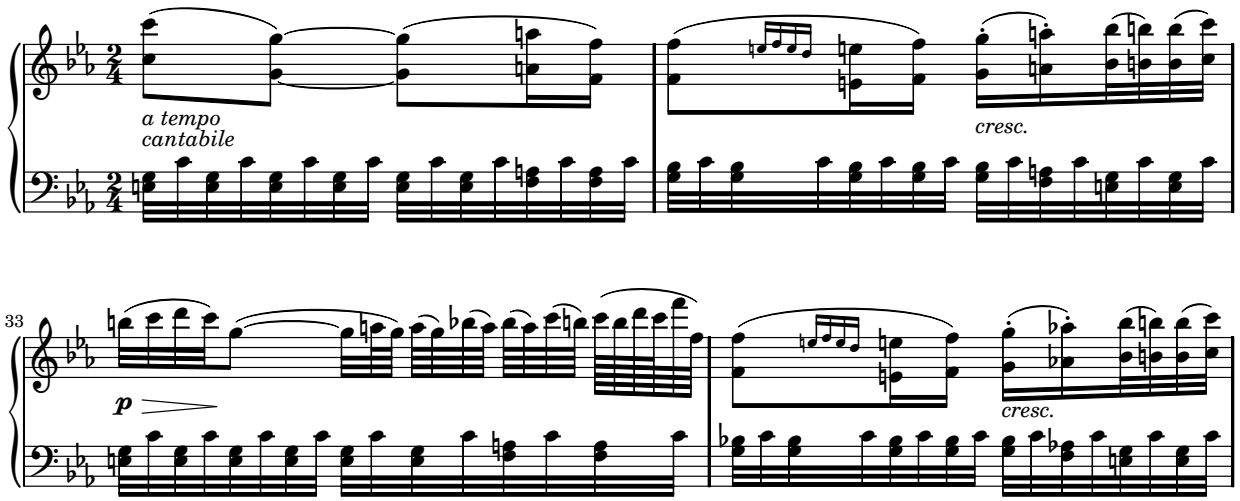
`\improvisationOn`, `\improvisationOff`.

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
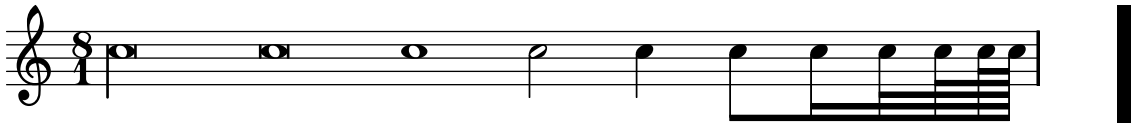


Cette section traite du rythme : durées, silences, barres de ligature et de mesure.

1.2.1 Écriture du rythme

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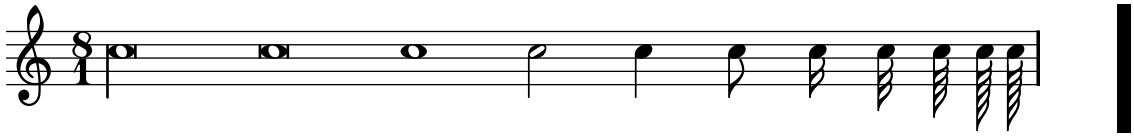
Dans les modes de notes, d'accords et de paroles, les durées sont écrites avec des chiffres et des points : les durées sont indiquées par leur valeur fractionnaire par rapport à la durée d'une ronde. Une noire, par exemple, qui équivaut à un 1/4 de ronde – *quarter note* en anglais – s'écrit `4`, alors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

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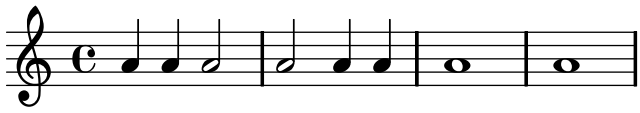
```
\time 8/1
\autoBeamOff
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
```



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 ⟨undefined⟩ [Notations anciennes], page ⟨undefined⟩.
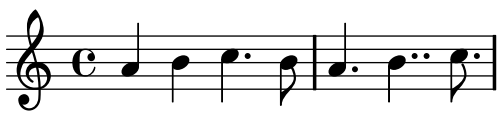
Si la durée d'une note n'est pas précisée, elle est alors assimilée à la durée de la note précédente. La valeur par défaut pour la première note est la noire (`4`).

```
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.

```
a4 b c4. b8 a4. b4.. c8.
```



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 44 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 215.

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 ⟨undefined⟩ [Notation proportionnelle], page ⟨undefined⟩.

Les points sont normalement haussés pour éviter les lignes de portée, sauf dans certaines polyphonies. Des commandes prédéfinies permettent de manuellement forcer une orientation particulière, comme indiqué au chapitre ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

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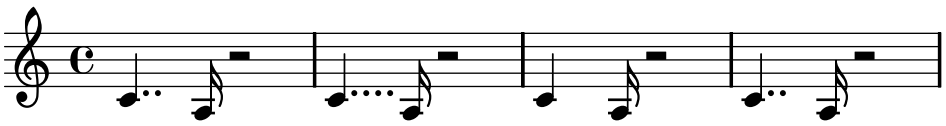
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Morceaux choisis

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

Voici comment modifier le nombre de points d'augmentation affectés à une note en particulier.

```
\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 69, [Liaisons de prolongation], page 44, [Hampes], page 184, Section 1.2.1 [Écriture du rythme], page 36, Section 1.2.2 [Écriture des silences], page 47, Section 2.1 [Musique vocale], page 215, ⟨undefined⟩ [Notations anciennes], page ⟨undefined⟩, ⟨undefined⟩ [Notation proportionnelle], page ⟨undefined⟩.

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 du trente-deuxième de soupir à la maxime (valant huit pauses).
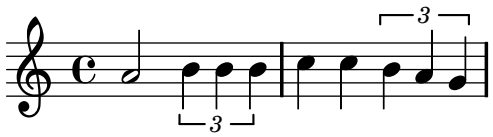
Nolets

Les nolets – triolets, quintolets, etc. – sont obtenus en multipliant toutes les durées d'une expression musicale par une fraction.

```
\times fraction { expression_musicale }
```

La durée de l'*expression_musicale* sera multipliée par la fraction. Le dénominateur de cette fraction sera imprimé au-dessus ou au-dessous des notes, parfois avec un crochet. Le nolet le plus courant est le triolet, dans lequel 3 notes ont la durée de 2, et où les notes durent donc 2/3 de leur valeur écrite.
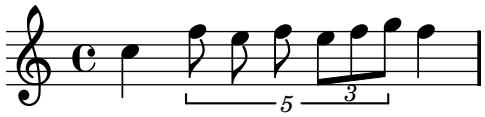
```
a2 \times 2/3 { b4 b b }
c4 c \times 2/3 { b4 a g }
```



Des commandes prédéfinies permettent de déroger au positionnement automatique du crochet en surplomb ou au-dessous des notes – voir le chapitre ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

Les nolets peuvent être imbriqués ; par exemple,

```
\autoBeamOff
c4 \times 4/5 { f8 e f \times 2/3 { e[ f g] } } f4 |
```



Lorsque, dans une imbrication, les nolets débutent au même instant, il vous faut recourir à la commande `\tweak`.

Vous pouvez interférer sur la durée des notes sans imprimer de crochet, comme indiqué au chapitre [Changement d'échelle des durées], page 43.

Commandes prédéfinies

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

Morceaux choisis
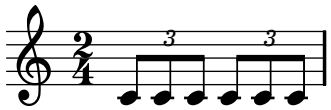
Plusieurs triolets avec une seule commande `\times`

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

Dans l'exemple suivant, deux triolets sont imprimés avec une seule fonction `\times`.

Pour plus d'information sur `make-moment`, voir la section appropriée du manuel de notation.

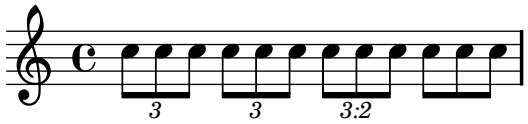
```
\relative c' {
  \time 2/4
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```



Modifier l'apparence du chiffre de nolet

L'apparence du chiffre est déterminée par la propriété `text` dans `TupletNumber`. La valeur par défaut imprime seulement le dénominateur, mais si elle est définie par la fonction `tuplet-number::calc-fraction-text`, la fraction entière *num*:*den* sera imprimée à la place.

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



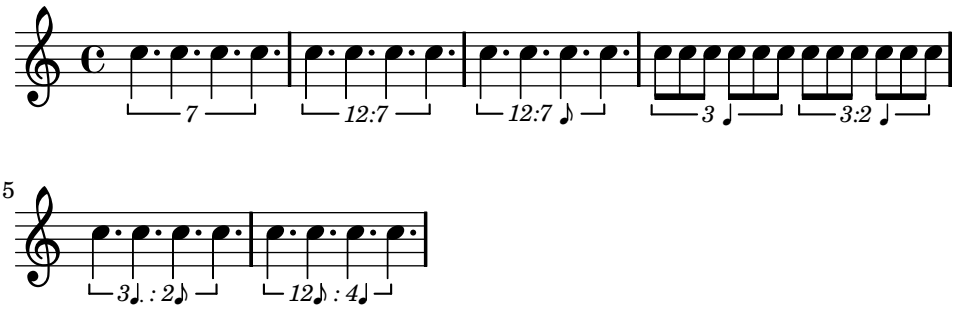
Nolets au chiffrage inhabituel

LilyPond sait aussi gérer des nolets 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 au chiffre.

```
\relative c'' {
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7) "8")
  \times 2/3  { c4. c4. c4. c4. }

  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text "4")
  \times 2/3  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber #'text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text "4")
  \times 2/3  { c8 c8 c8 c8 c8 c8 }

  \once \override TupletNumber #'text =
    #(tuplet-number::fraction-with-notes "4." "8")
  \times 2/3  { c4. c4. c4. c4. }
  \once \override TupletNumber #'text =
    #(tuplet-number::non-default-fraction-with-notes 12 "8" 4 "4")
  \times 2/3  { c4. c4. c4. c4. }
}
```
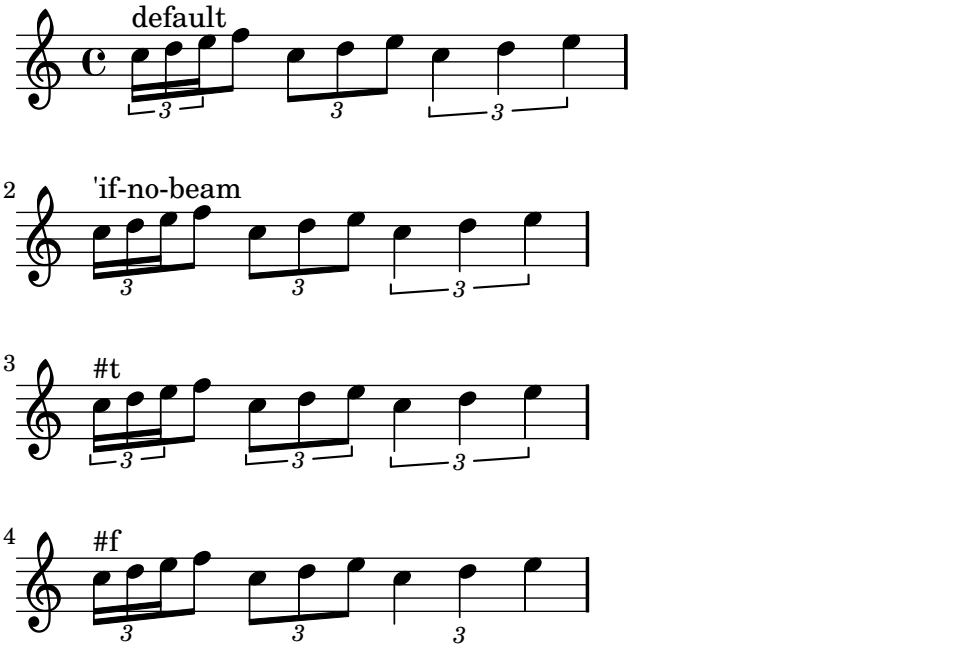


Contrôle de l'impression des crochets de nolet

Selon la tradition, les crochets indicateurs de nolet sont toujours imprimés sauf dans le cas où ils seraient de la même longuer 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, et `#'if-no-beam` les imprimera en l'absence de ligature.

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

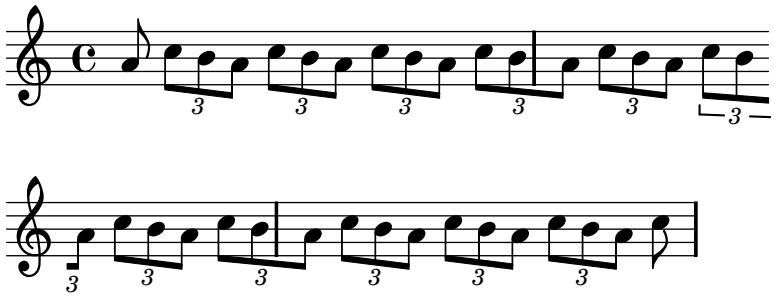
\new Voice {
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket #'bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket #'bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket #'bracket-visibility = ##f
    << \music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



Saut de ligne au milieu d'un nolet avec ligature

Cet exemple peu académique démontre comment il est possible d'insérer un saut de ligne dans un nolet 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 { \times 2/3 { c[ b a] } }
  % Insert a manual line break within a tuplet
  \times 2/3 { c[ b \bar "" \break a] }
  \repeat unfold 5 { \times 2/3 { c[ b a] } }
  c8
}
```



Voir aussi

Glossaire musicologique : Section "triolet" dans *Glossaire*, Section "nolet" dans *Glossaire*, Section "polymétrie" dans *Glossaire*.

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

Manuel de notation : [Gestion du temps], page 97, [Changement d'échelle des durées], page 43, ⟨undefined⟩ [La commande d'affinage (*tweak*)], page ⟨undefined⟩, [Notation polymétrique], page 63.

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*.

Problèmes connus et avertissements

Lorsqu'un nolet en début de portée est précédé d'une note d'ornement, celle-ci doit se placer avant la commande `\times` pour éviter toute erreur. Ailleurs dans la partition, les notes d'ornement peuvent se placer au sein de l'expression constituant le nolet.

Lorsqu'un nolet en début de mouvement est combiné avec une indication de tempo – grâce à la commande `\tempo` –, l'expression contenant le nolet doit impérativement faire partie d'un
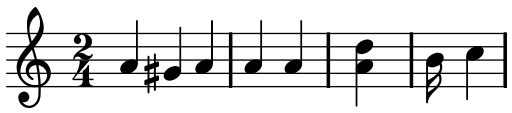
bloc `\new Voice` comme indiqué au chapitre Section “Les voix contiennent la musique” dans *Manuel d'initiation*.

Changement d'échelle des durées

Vous pouvez altérer la durée des notes, silences ou accords en leur joignant une fraction *N/M*, donnant « **N/M* » – ou « **N* » si *M=1*. Ceci 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.

Dans l'exemple suivant, les trois premières notes prennent exactement deux temps, mais aucun triolet n'est imprimé.
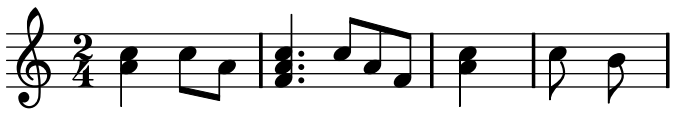
```
\time 2/4
% Alter durations to triplets
a4*2/3 gis4*2/3 a4*2/3
% Normal durations
a4 a4
% Double the duration of chord
<a d>4*2
% Duration of quarter, appears like sixteenth
b16*4 c4
```



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 la fraction *numérateur/dénominateur*. N'oubliez pas de préserver les espaces de part et d'autre du point. Voici un exemple illustrant la manière de comprimer ou étirer de la musique :

```
\time 2/4
% Normal durations
<c a>4 c8 a
% Scale music by *2/3
\scaleDurations #'(2 . 3) {
  <c a f>4. c8 a f
}
% Scale music by *2
\scaleDurations #'(2 . 1) {
  <c' a>4 c8 b
}
```



Cette technique est tout à fait appropriée à la notation polymétrique – voir [Notation polymétrique], page 63.

Voir aussi

Manuel de notation : [Nolets], page 38, [Silences invisibles], page 49, [Notation polymétrique], page 63.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.
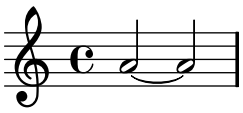
Liaisons de prolongation

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.
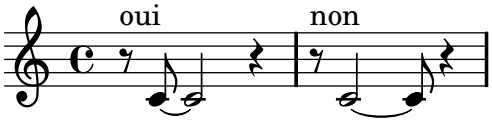
Une liaison de tenue s’indique au moyen d’un tilde `~`.

```
a2 ~ a
```



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 :

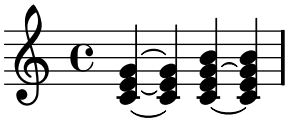
```
\relative {
  r8^"oui" c8 ~ c2 r4 |
  r8^"non" c2 ~ c8 r4
}
```



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 66. Ce procédé divise automatiquement les notes trop longues, et les lie par-delà les barres de mesure.
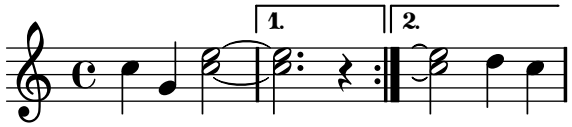
Quand une liaison de tenue se trouve entre deux accords, toutes les notes de même hauteur entre ces deux accords sont reliées. S’il n’y en a aucune, aucune liaison n’est créée. Il est également possible de lier partiellement deux accords, en mettant les liaisons à l’intérieur des accords.

```
<c e g> ~ <c e g>
<c~ e g~ b> <c e g b>
```



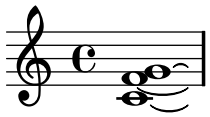
Lorsqu’une mesure « de seconde fois » après une reprise commence sur une note liée, la liaison doit être répétée, comme ici :

```
\repeat volta 2 { c g <c e>2 ~ }
\alternative {
  % First alternative: following note is tied normally
  { <c e>2. r4 }
  % Second alternative: following note has a repeated tie
  { <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 :
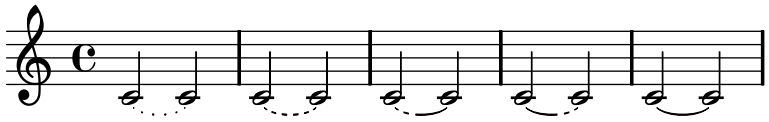
```
<c f g>1\laissezVibrer
```



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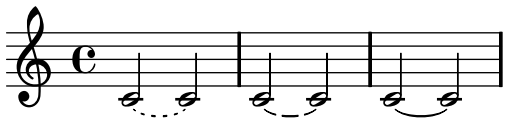
Les liaisons de prolongation peuvent s'imprimer sous la forme de ligne continue, discontinue ou pointillée.

```
\tieDotted
c2 ~ c
\tieDashed
c2 ~ c
\tieHalfDashed
c2 ~ c
\tieHalfSolid
c2 ~ c
\tieSolid
c2 ~ c
```



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

```
\tieDashPattern #0.3 #0.75
c2 ~ c
\tieDashPattern #0.7 #1.5
c2 ~ c
\tieSolid
c2 ~ c
```



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 108.

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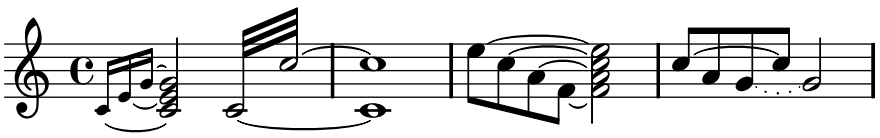
`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

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).

```
\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>
}
```



Voir aussi

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

Manuel de notation : [Liaisons d’articulation], page 108, [Découpage automatique des notes], page 66.

Morceaux choisis : Section “Rythme” 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 clé ou d’octave pendant une liaison de tenue produit un résultat indéfini. Dans ces cas-là, il est préférable d’utiliser un *legato*.

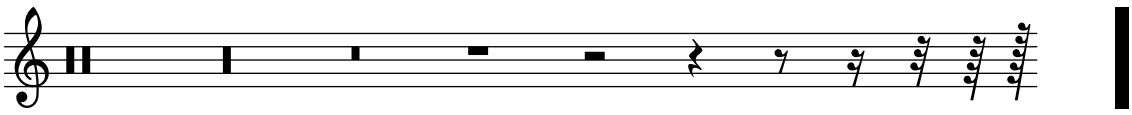
1.2.2 Écriture des silences

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`. Les durées supérieures à la pause s’indiquent à l’aide de commandes prédéfinies :
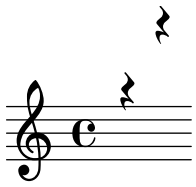
```
\new Staff {
  % These two lines are just to prettify this example
  \time 16/1
  \override Staff.TimeSignature #'stencil = ##f
  % Print a maxima rest, equal to four breves
  r\maxima
  % Print a longa rest, equal to two breves
  r\longa
  % Print a breve rest
  r\breve
  r1 r2 r4 r8 r16 r32 r64 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 dans la section [Silences valant une mesure], page 50.

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.

```
a'4\rest d'4\rest
```



Morceaux choisis

Styles de silences

Les silences peuvent être gravés selon différents styles.

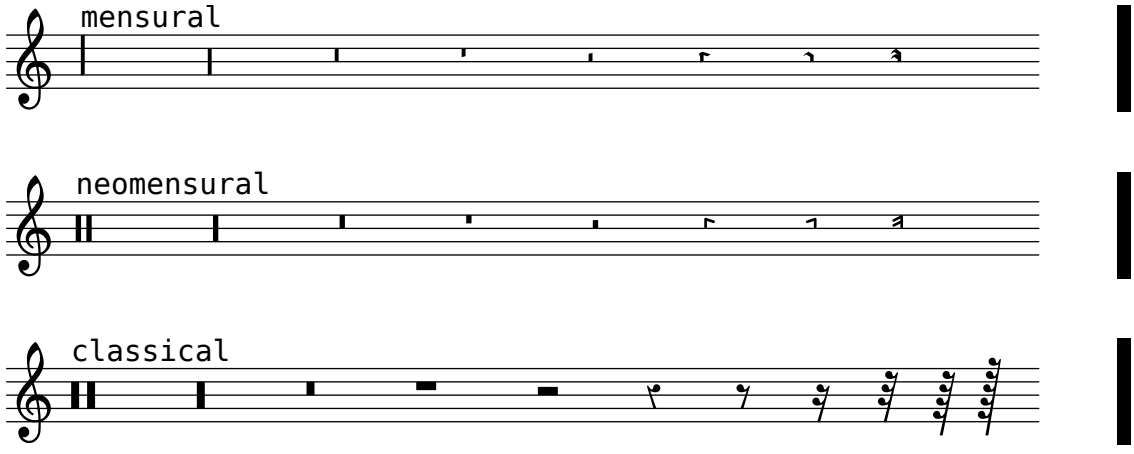
```
\layout {
  indent = 0
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}

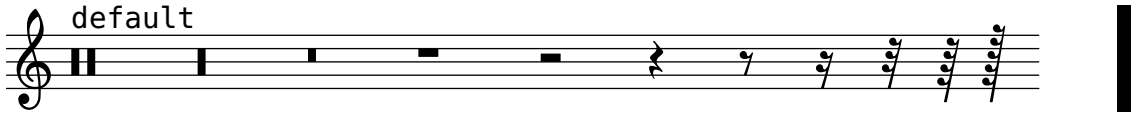
\new Staff \relative c {
  \cadenzaOn
  \override Staff.Rest #'style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""

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

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

  \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 50.

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

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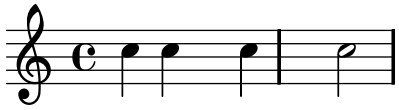
Problèmes connus et avertissements

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 trente-deuxième de soupir à la maxime (équivalant à huit pauses).

Silences invisibles
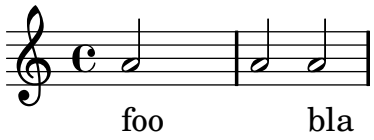
Un silence invisible – que l’on pourrait appeler un « saut » – peut être entré comme une note avec le nom de note `s` ou avec `\skip` *`durée`* :

```
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`.

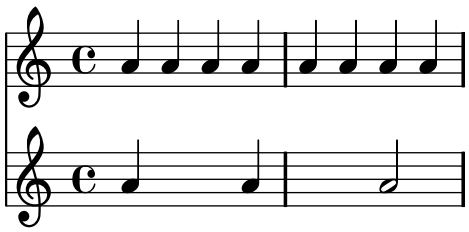
```
<<
  {
    a2 \skip2 a2 a2
  }
  \new Lyrics {
    \lyricmode {
      foo2 \skip 1 bla2
    }
  }
>>
```



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`.

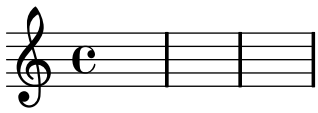
```
<<
  {
    \repeat unfold 8 {a4}
```

```
  }
  {
    a4 \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 181, ⟨undefined⟩ [Visibilité des objets], page ⟨undefined⟩.

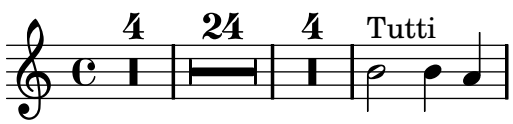
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

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

Silences valant une mesure

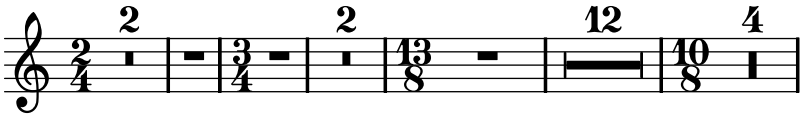
Un silence valant une ou plusieurs mesures entières s'entre avec un `R` majuscule.

```
% Rest measures contracted to single measure
\compressFullBarRests
R1*4
R1*24
R1*4
b2^"Tutti" b4 a4
```
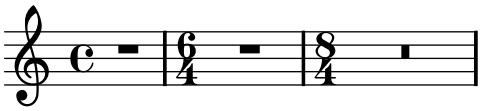


Ceci ne peut être utile que pour une mesure complètement vide. Sa durée doit donc correspondre à la longueur de la mesure telle que définie par la métrique. C'est la raison pour laquelle on utilisera aussi des points d'augmentation ou des fractions :

```
\compressFullBarRests
\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 |
```



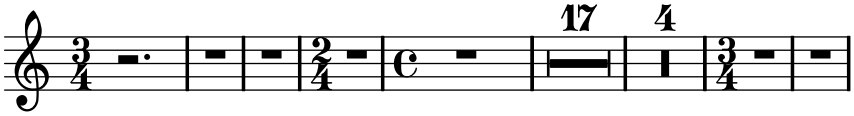
Un `R` qui s'étend sur une seule mesure s'imprime tantôt comme une pause, tantôt comme une brève – ou « bâton de pause » – et sera centré sur la mesure quelle qu'en soit la métrique :

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



Par défaut, un silence multimesures sera répété sur autant de mesures que nécessaire. Il peut aussi n'être imprimé qu'une seule fois, surplombé du nombre de mesures vides – ou « à compter » :

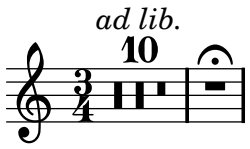
```
% Default behavior
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Rest measures contracted to single measure
\compressFullBarRests
r1 | R1*17 | R1*4 |
% Rest measures expanded
\expandFullBarRests
\time 3/4
R2.*2 |
```



Vous pouvez aussi ajouter du texte à un silence multimesures en utilisant la syntaxe *note*-`markup` (cf. Section 1.8.2 [Mise en forme du texte], page 198). La variable `\fermataMarkup` quant à elle permet d'ajouter un point d'orgue :
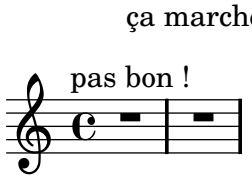
```
\compressFullBarRests
\time 3/4
```

```
R2.*10^\markup { \italic "ad lib." }
R2.^\fermataMarkup
```



> **Note :** C'est `MultiMeasureRestText` qui créera le texte, et non `TextScript`. 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
R1^"pas bon !"
% Formulation correcte, qui fonctionnera
\override MultiMeasureRestText #'padding = #5
R1^"ça marche !"
```



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

Commandes prédéfinies

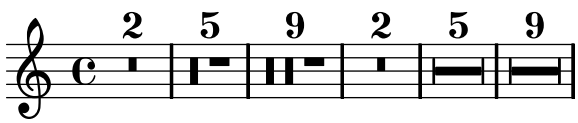
`\textLengthOn`, `\textLengthOff`, `\fermataMarkup`, `\compressFullBarRests`, `\expandFullBarRests`.

Morceaux choisis

Modifier l'apparence d'un silence multi-mesures

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 :

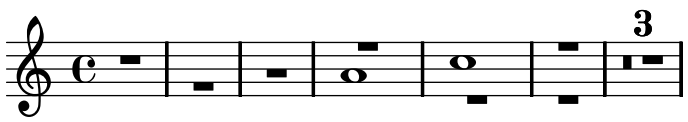
```
\relative c'' {
  \compressFullBarRests
  R1*2 | R1*5 | R1*9
  \override MultiMeasureRest #'expand-limit = #3
  R1*2 | R1*5 | R1*9
}
```



Positionnement des silences multi-mesures

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 multi-mesures. Néanmoins, et uniquement dans le cadre de musique polyphonique, les silences multi-mesures sont positionnés différemment selon qu'ils appartiennent à une voix au numéro pair ou impair. Le positionnement des silences multi-mesures peut se contrôler ainsi :

```
\relative c'' {
  % MMR - Multi-Measure Rest
  % MMRs by default are set under the fourth line
  R1
  % They can be moved with an override
  \override MultiMeasureRest #'staff-position = #-2
  R1
  % A value of 0 is the default position;
  % the following trick moves the rest to the center line
  \override MultiMeasureRest #'staff-position = #-0.01
  R1
  % MMRs in odd-numbered voices are under the top line
  << { R1 } \\ { a1 } >>
  % MMRs in even-numbered voices are under the bottom line
  << { c1 } \\ { R1 } >>
  % They remain separated even in empty measures
  << { R1 } \\ { R1 } >>
  % This brings them together even though there are two voices
  \compressFullBarRests
  <<
    \revert MultiMeasureRest #'staff-position
    { R1*3 }
    \\
    \revert MultiMeasureRest #'staff-position
    { R1*3 }
  >>
}
```



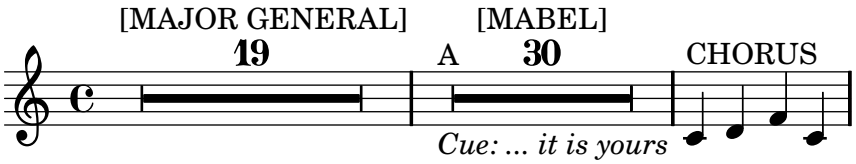
Ajout de texte à un silence multi-mesures

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

Rappelez-vous qu'un silence invisible génère une barre de mesure. Le texte attaché à ce silence invisible sera alors aligné sur la gauche de là où serait positionnée la note. Cependant, si la taille de la mesure est déterminée par la longueur du texte, il apparaîtra comme centré.

```
\relative c' {
  \compressFullBarRests
  \textLengthOn
  s1*0^\markup { [MAJOR GENERAL] }
  R1*19
  s1*0_\markup { \italic { Cue: ... it is yours } }
```

```
  s1*0^\markup { A }
  R1*30^\markup { [MABEL] }
  \textLengthOff
  c4^\markup { CHORUS } d f c
}
```



Voir aussi

Glossaire musicologique : Section “silence multi-mesures” dans *Glossaire*.

Manuel de notation : [Durées], page 37, Section 1.8 [Texte], page 189, Section 1.8.2 [Mise en forme du texte], page 198, [Commentaires textuels], page 189.

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 “MultiMeasureRestText” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Vous ne pouvez pas utiliser de doigtés (p.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.
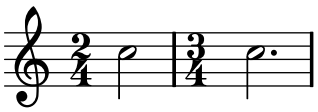
Les silences multimesure peuvent générer des collisions avec d’autres silences.

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 :

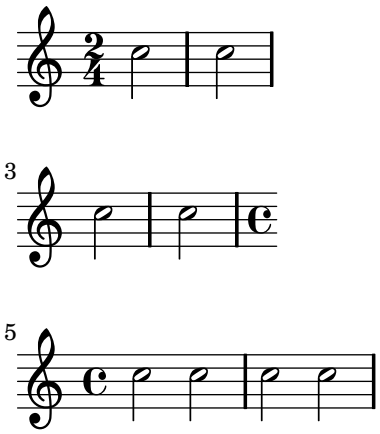
```
\time 2/4 c2
\time 3/4 c2.
```



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 ⟨undefined⟩ [Visibilité des objets], page ⟨undefined⟩.
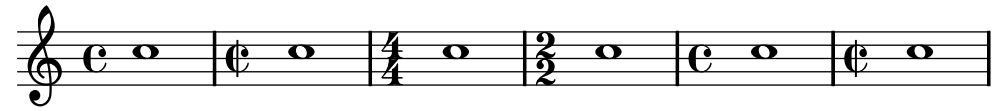
```
\time 2/4
c2 c
\break
c c
\break
\time 4/4
```

```
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 :
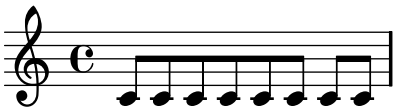
```
% Default style
\time 4/4 c1
\time 2/2 c1
% Change to numeric style
\numericTimeSignature
\time 4/4 c1
\time 2/2 c1
% Revert to default style
\defaultTimeSignature
\time 4/4 c1
\time 2/2 c1
```



Les métriques anciennes sont abordées au chapitre ⟨undefined⟩ [Métriques anciennes], page ⟨undefined⟩.

En plus de déterminer la métrique qui sera imprimée, la commande `\time` règlera aussi les valeurs par défaut des propriétés `baseMoment`, `beatStructure` et `beamExceptions`. Les valeurs prédéterminées par défaut de ces différentes propriétés sont inscrites dans le fichier ‘`scm/time-signature-settings.scm`’. Vous pouvez les modifier de la manière suivante :
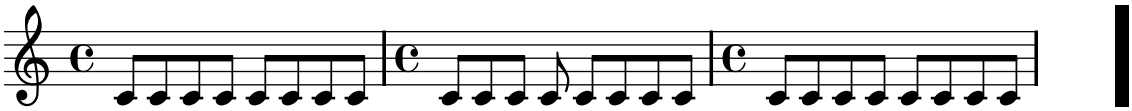
```
\score {
  \relative c' {
    \overrideTimeSignatureSettings
      #'Score    % context
      #'(4 . 4)  % timeSignatureFraction
      #'(1 . 4)  % baseMomentFraction
      #'(3 1)    % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```



`\overrideTimeSignatureSettings` prend cinq arguments : un contexte, une fraction indiquant la métrique, une fraction indiquant la base de la pulsation, la structure de cette pulsation, et enfin les règles d'exception en matière de ligature. Le contexte, exprimé sous la forme d'une variable Scheme, stipule à quel contexte ces réglages par défaut s'appliqueront – partition, portée ou voix. La métrique est indiquée par un doublet Scheme. La base de la pulsation est un doublet Scheme contenant le numérateur et le dénominateur correspondants à l'unité de temps pour cette métrique. La structure de la pulsation, exprimée sous la forme d'une liste d'unités de base en Scheme, détermine coment les temps sont regroupés dans la mesure. Les règles d'exception en matière de ligature s'énoncent sous la forme d'une liste associative pour les cas où il faudrait dépasser la pulsation de base – voir [Définition des règles de ligature automatique], page 71 à ce propos.

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

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



Commandes prédéfinies

`\numericTimeSignature`, `\defaultTimeSignature`.

Morceaux choisis

Changement de métrique sans affecter les règles de ligature

La commande `\time` gère les propriétés `timeSignatureFraction`, `beatLength`, `beatGrouping` et `measureLength` dans le contexte `Timing`, normalement rattaché à `Score`. Le fait de modifier la valeur de `timeSignatureFraction` aura pour effet de changer l'apparence du symbole affiché sans pour autant affecter les autres propriétés de la métrique :

```
\markup {
  This snippet is deprecated as of 2.13.5 and will be removed in 2.14
}
```
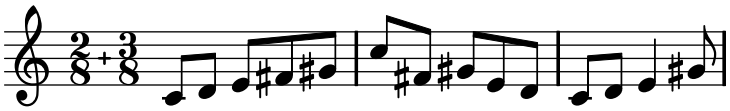
This snippet is deprecated as of 2.13.5 and will be removed in 2.14

Métrique décomposée

Des métriques telles que « 5/8 » peuvent s'interpréter sous une forme décomposée — « 3/8 + 2/8 » par exemple — qui combine plusieurs métriques. LilyPond est capable de rendre ce type de notation, plus facile à lire et à interpréter, en imprimant cette métrique composite et en adaptant les règles de ligature automatique en conséquence.

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))

\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```
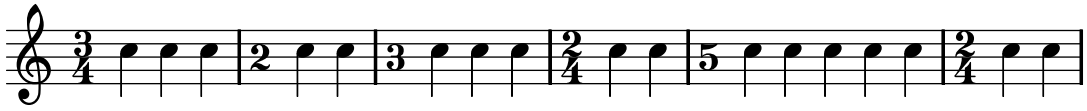


*Affichage seulement du numérateur d'une métrique (au*lieu d'une fraction)

La métrique est parfois indiquée non pas par une fraction (p.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` d'annuler ces modifications. Lorsque cette métrique sous le forme d'une 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 : ⟨undefined⟩ [Métriques anciennes], page ⟨undefined⟩, [Gestion du temps], page 97.
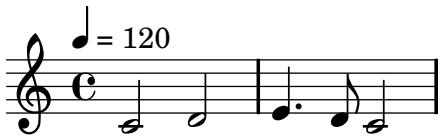
Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

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

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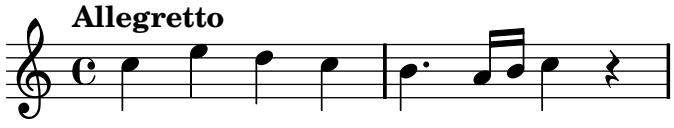
Une indication métronomique s’insère tout simplement comme ceci :

```
\tempo 4 = 120
c2 d
e4. d8 c2
```
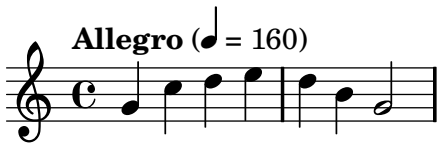


Vous pouvez préférer une indication textuelle :

```
\tempo "Allegretto"
c4 e d c
b4. a16 b c4 r4
```
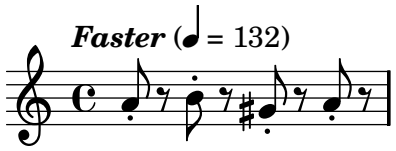


Lorsque vous combinez des indications métronomiques sous forme textuelle et numérique, l’indication numérique sera placée entre parenthèses :

```
\tempo "Allegro" 4 = 160
g4 c d e
d4 b g2
```
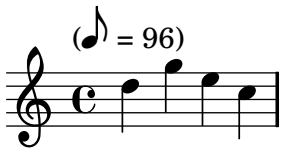


En matière d’indication textuelle, vous pouvez utiliser n’importe quel objet de type *markup*, comme ici :

```
\tempo \markup { \italic Faster } 4 = 132
a8-. r8 b-. r gis-. r a-. r
```



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

```
\tempo "" 8 = 96
d4 g e c
```



Morceaux choisis

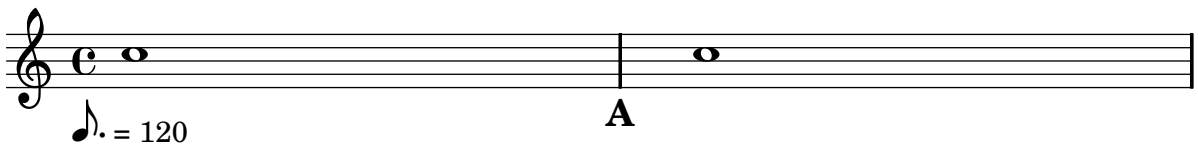
Impression du métronome et des repères sous la portée

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` ou `RehearsalMark` les placera au-dessous de la portée.

```
\layout { 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
}
```
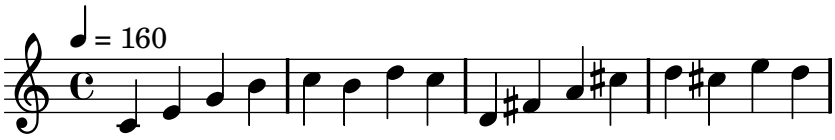


Changement de tempo sans indication sur la partition

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

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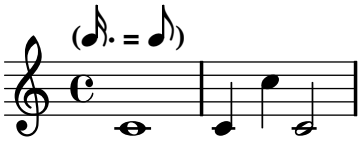
```
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```



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 198.

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 1.8.2 [Mise en forme du texte], page 198, ⟨undefined⟩ [Sortie MIDI], page ⟨undefined⟩.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "MetronomeMark" dans *Référence des propriétés internes*.
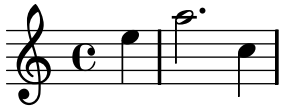
Levées

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

```
\partial durée
```
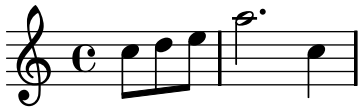
où *durée* correspond à la valeur rythmique devant être ajoutée avant la première mesure complète :

```
\partial 4 e4 |
a2. c,4 |
```



Une mesure incomplète peut être de n'importe quelle longueur inférieure à la mesure complète :

```
\partial 8*3 c8 d e |
a2. c,4 |
```



Le programme traduit cette commande `\partial durée` en

```
\set Timing.measurePosition = -durée
```

Par exemple, `\partial 8*3` est traduit en :

```
\set Timing.measurePosition = #(ly:make-moment -3 8)
```

La propriété `measurePosition` contient un nombre rationnel qui indique, à ce point précis, où l'on en est de la mesure. Notez qu'il peut s'agir d'un nombre négatif ; `\partial 4` signifie, pour le programme : « Dans cette mesure, il reste juste une noire ».

Voir aussi

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

Manuel de notation : [Notes d'ornement], page 92.

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

Références des propriétés internes : Section "Timing_translator" dans *Référence des propriétés internes*.
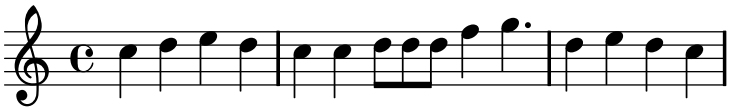
Problèmes connus et avertissements

`\partial` n'est destiné à être utilisé qu'en début de pièce. Si on l'utilise ailleurs qu'au début, des messages d'erreurs peuvent s'afficher. Utilisez plutôt `\set Timing.measurePosition` en pareil cas.

Musique sans métrique
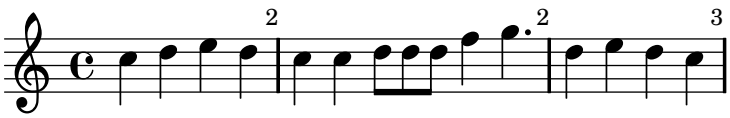
Les barres de mesure et les numéros de mesure sont calculés automatiquement, ce qui n'est pas souhaitable dans le cas d'une musique non mesurée – les cadences, par exemple. Les commandes `\cadenzaOn` et `\cadenzaOff` permettent de désactiver et de rétablir ce comportement automatique.

```
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```
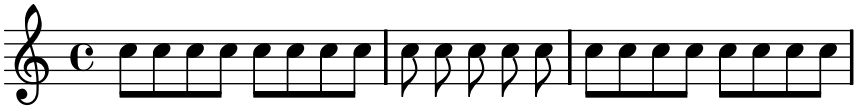


La numérotation reprend dès la fin d'une cadence, comme si celle-ci n'avait pas existé :

```
% Show all bar numbers
\override Score.BarNumber #'break-visibility = #all-visible
c4 d e d
\cadenzaOn
c4 c d8[ d d] f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```



L'instruction `\cadenzaOn` désative les ligatures automatique ; 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 77).

```
\repeat unfold 8 { c8 }
\cadenzaOn
\repeat unfold 5 { c8 }
\bar"|"
\cadenzaOff
\repeat unfold 8 { c8 }
```



Notez bien que 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 63.

Commandes prédéfinies

`\cadenzaOn`, `\cadenzaOff`.

Voir aussi

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

Manuel de notation : ⟨undefined⟩ [Visibilité des objets], page ⟨undefined⟩, [Notation polymétrique], page 63.

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

Problèmes connus et avertissements

LilyPond ne change de ligne ou de page qu'au niveau des barres de mesure. Si votre musique non mesurée s'étend sur plus d'une ligne, il vous faudra insérer des barres de mesure invisibles,

```
\bar ""
```

pour indiquer où des sauts de ligne peuvent prendre place.

Les règles de ligature automatique peuvent être fortement perturbées lorsqu'une cadence dure plus de deux temps. Aussi nous vous recommandons de recourir aux ligatures manuelles à l'aide de `[` et `]`.

Au risque d'obtenir des messages d'erreur inhabituels, il est fortement recommandé de créer explicitement un contexte `Voice` lorsqu'un morceau débute par un `\cadenzaOn`.

```
\new Voice {
  \relative c' {
    \cadenzaOn
    c16[^"Solo Free Time" d e f] g2.
    \bar "||"
    \cadenzaOff
  }
}
```

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.

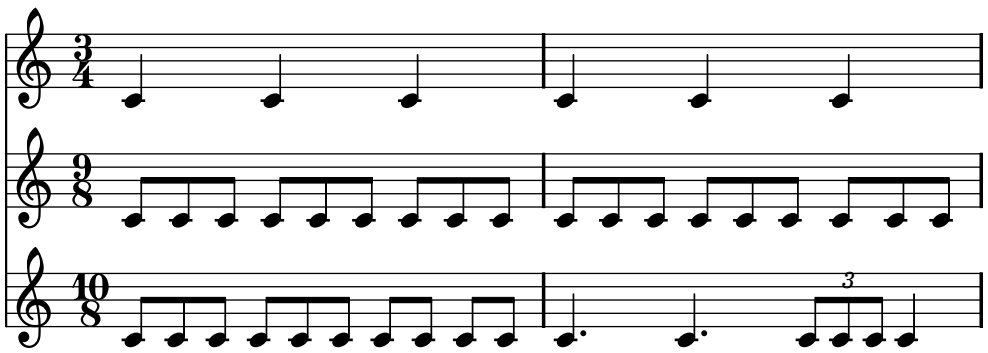
Portées aux métriques différentes, mesures d'égale longueur

Il suffit, pour obtenir cette forme de notation, de tout d'abord affecter une même métrique aux différentes portées. L'indicateur de métrique sera ensuite remplacé dans chacune des portées par un quotient donné en argument à la propriété `timeSignatureFraction`. La durée des notes sera enfin proratisée selon la métrique commune (voir [Métrique], page 54) grâce à la fonction `\scaleDurations`, qui s'utilise tout comme la commande `\times` – sans toutefois créer de crochet de nolet (voir [Changement d'échelle des durées], page 43).

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 c' <<
  \new Staff {
    \time 3/4
    c4 c c |
    c 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. c4. \times 2/3 { c8[ c c] } c4
    }
  }
>>
```



Portées aux métriques différentes, mesures de longueur inégale

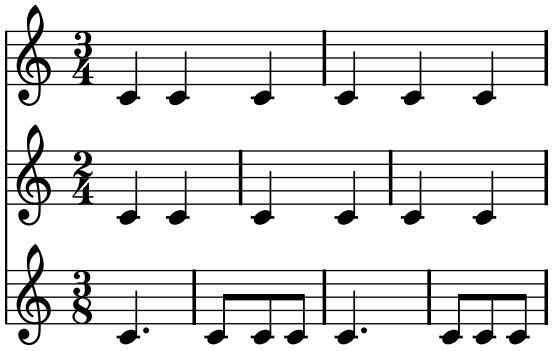
Il arrive aussi que chaque portée ait sa propre métrique. Vous y parviendrez en déplaçant le `Timing_translator` et le `Default_bar_line_engraver` dans le contexte `Staff`.

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

```
    c4. |
    c8 c c |
  }
>>
```
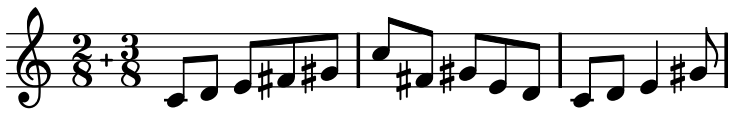


Morceaux choisis

Métrique décomposée

Des métriques telles que « 5/8 » peuvent s'interpréter sous une forme décomposée — « 3/8 + 2/8 » par exemple — qui combine plusieurs métriques. LilyPond est capable de rendre ce type de notation, plus facile à lire et à interpréter, en imprimant cette métrique composite et en adaptant les règles de ligature automatique en conséquence.

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))

\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```



Voir aussi

Glossaire musicologique : Section “polymétrie” dans *Glossaire*, Section “métrique composite” dans *Glossaire*, Section “métrique” dans *Glossaire*.

Manuel de notation : [Métrique], page 54, [Changement d'échelle des durées], page 43.

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
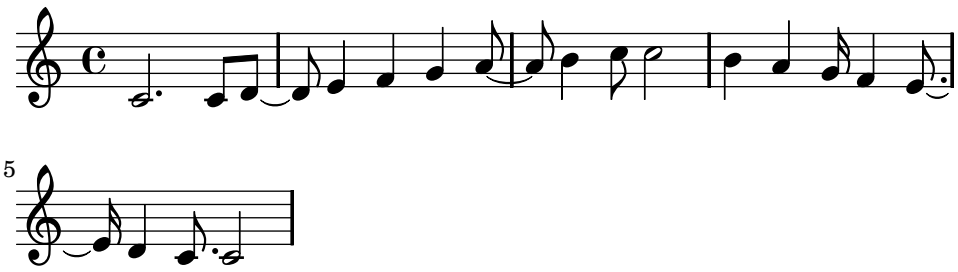
L'utilisation de métriques différentes en parallèle entraine un alignement vertical. De ce fait, les barres de mesure ont tendance à fausser l'espacement régulier entre les portées.

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`. Dans l'exemple suivant, les notes qui dépassent de la mesure sont divisées et liées.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
}

{ c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 }
```



Ce graveur divise toutes les notes qui sortent de la mesure, et insère 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.

Voir aussi

Glossaire musicologique : Section "liaison de tenue" dans *Glossaire*.

Manuel d'initiation : Section "Tout savoir sur les graveurs" dans *Manuel d'initiation*, Section "Ajout et suppression de 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 "Forbid_line_break_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Bien que toutes les durées – particulièrement celles contenant des nolets – ne puissent être exactement représentées avec des notes normales et des points, le graveur `Completion_heads_engraver` n'insèrera pas de nolet.

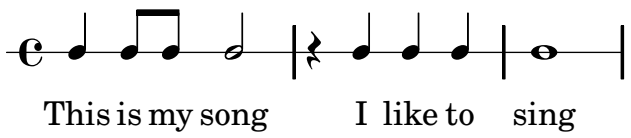
`Completion_heads_engraver` affecte seulement les notes, il ne divise pas les silences.

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.

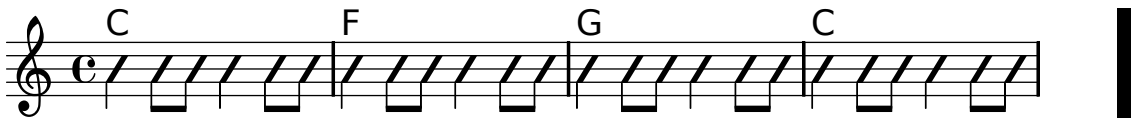
```
<<
  \new RhythmicStaff {
```

```
    \new Voice = "myRhythm" {
      \time 4/4
      c4 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
  }
>>
```



Commandes prédéfinies

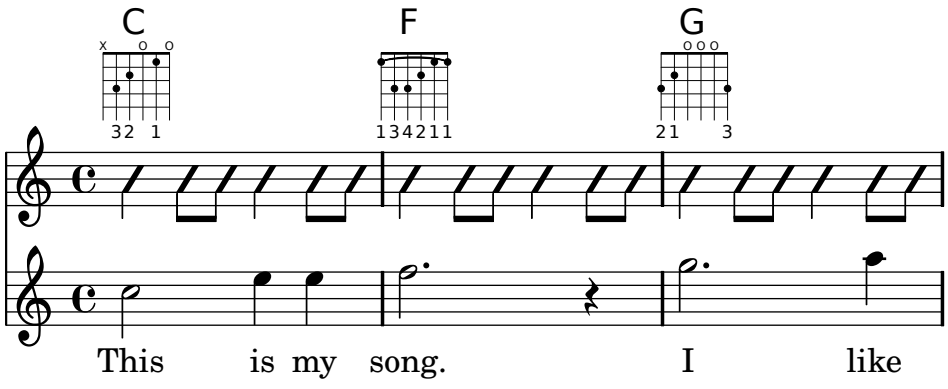
`\improvisationOn`, `\improvisationOff`.

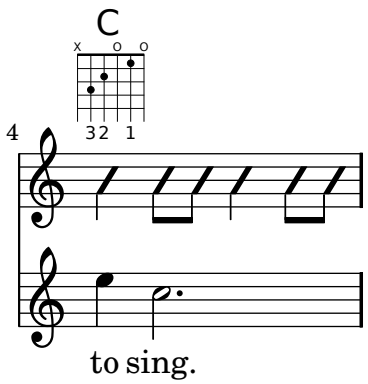
Morceaux choisis

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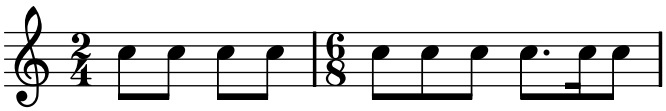
Référence des propriétés internes : Section “RhythmicStaff” dans *Référence des propriétés internes*. Section “Pitch_squash_engraver” dans *Référence des propriétés internes*.

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.
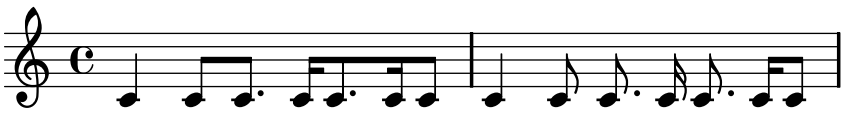
```
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
```



Lorsque ce comportement automatisé n’est pas satisfaisant, on peut définir des groupements manuellement – voir [Barres de ligature manuelles], page 77. 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` :

```
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.

Note : 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 71.

Commandes prédéfinies

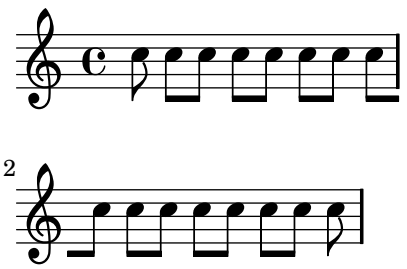
`\autoBeamOff`, `\autoBeamOn`.

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.

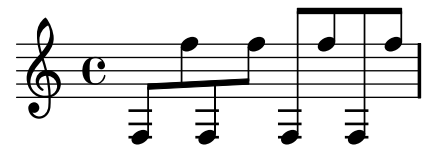
```
\relative c'' {
  \override Beam #'breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}
```



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`' espace, 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

Le 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
```
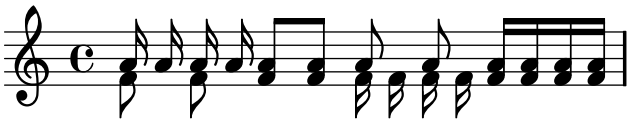
pour désactiver les ligatures automatiques pour l'ensemble de la portée concernée.

L'instruction `\partcombine` fonctionne apparament 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 77, [Définition des règles de ligature automatique], page 71.

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*.
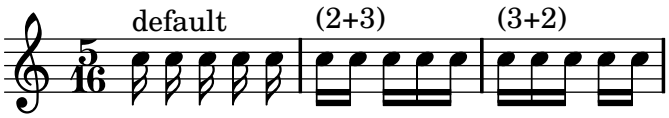
Problèmes connus et avertissements

Les ligatures peuvent générer des collisions avec des têtes de notes ou altérations appartenant à d'autres voix.

Définition des règles de ligature automatique

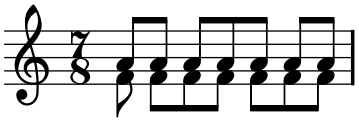
Dans la plupart des cas, une ligature automatique se termine à la fin d'une pulsation. Ce sont les propriétés de contexte `baseMoment` et `beatStructure` qui détermineront là où prend fin la pulsation. La propriété `beatStructure` est constituée d'une liste d'éléments Scheme qui définit la longueur de chaque pulsation, prenant `baseMoment` comme unité. L'unité de base (le `baseMoment`) correspond par défaut au dénominateur de la métrique. D'autre part, chaque unité de `baseMoment` constitue une seule pulsation.

```
\time 5/16
c16^"default" c c c c |
\set Timing.beatStructure = #'(2 3)
c16^"(2+3)" c c c c |
\set Timing.beatStructure = #'(3 2)
c16^"(3+2)" c c c c |
```



Les effets de règles de ligature peuvent être restreints à un contexte particulier. En l'absence de régles particulières déterminées dans un contexte de niveau inférieur, les règles définies au niveau directement supérieur s'appliqueront.
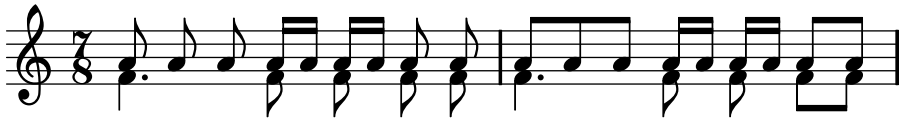
```
\new Staff {
  \time 7/8
  \set Staff.beatStructure = #'(2 3 2)
  <<
    \new Voice = one {
      \relative c'' {
        a8 a a a a a a
      }
    }
    \new Voice = two {
      \relative c' {
        \voiceTwo
        \set Voice.beatStructure = #'(1 3 3)
        f8 f f f f f f
      }
    }
  >>
}
```



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` :
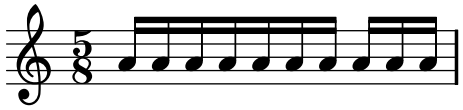
```
\time 7/8
% rhythm 3-1-1-2
% Context Voice specified --  does not work correctly
% Because of autogenerated voices, all beating will
% be at baseMoment (1 . 8)
\set beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. f8 f f f} >>

% Works correctly with context Staff specified
\set Staff.beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \\ {f4. 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
\set Timing.baseMoment = #(ly:make-moment 1 16)
\set Timing.beatStructure = #'(7 3)
\repeat unfold 10 { a16 }
```
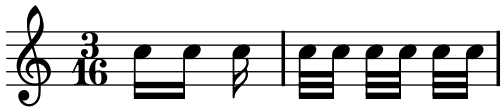


`baseMoment` constitue un **moment**, autrement dit une unité de durée musicale. La fonction Scheme `ly:make-moment` est tout particulièrement chargée de créer cette quantité de type *moment* – voir [Gestion du temps], page 97.

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 54.

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`'.

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

```
\time 3/16
\set Timing.beatStructure = #'(2 1)
\set Timing.beamExceptions =
  #'(                          ;ouvre de la liste associative
     (end .                    ;ceci concerne la terminaison des ligatures
      (                        ;ouvre la liste des points de terminaison
       ((1 . 32) . (2 2 2))    ;règle pour les triples -- groupées à la double
      )))                      %referme la liste
c16 c c |
\repeat unfold 6 { c32 } |
```



`beamExceptions` contient la liste des règles de ligature selon leur type.

Le seul type de règle pris en charge à ce jour est `#'end`, qui concerne les terminaisons.

Chaque règle est constituée d'une liste de doublets associatifs en langage Scheme (un *alist* pour les puristes), qui indique la durée de base et sa règle de regroupement.

```
#'((durée-type1 . groupement-1)
   (durée-type2 . groupement-2)
   (durée-type3 . groupement-3))
```

durée-type est constitué d'une paire indiquant la durée de base – par exemple `(1 . 16)` pour une double-croche.

groupement est constitué d'une liste Scheme qui indique le regroupement à effectuer, en unité de type de ligature.

> **Note :** La propriété `beamExceptions` doit contenir absolument **toutes** les exceptions. Il n'est en effet pas possible d'en ajouter, modifier ou suprimer *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, de spécifier à nouveau la métrique.

```
\time 4/4
\repeat unfold 8 { a8 }
% eliminate beam exception that groups beats 1, 2 and 3, 4
\set Timing.beamExceptions = #'()
\repeat unfold 8 { a8 }
% go back to default behavior
\time 4/4
\repeat unfold 8 { 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 sosnt abordées au chapitre [Métrique], page 54.

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
- utiliser les 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.

Pour information, les règles de ligature par défaut sont répertoriées dans le fichier '`scm/time-signature-settings.scm`'.

Morceaux choisis

Subdivision des ligatures
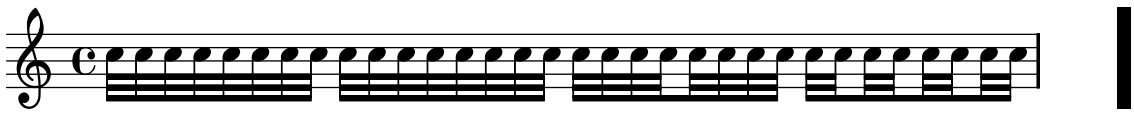
Les ligatures d'une succession de notes de durée inférieure à la croche ne sont pas subdivisées par défaut. Autrement dit, tous les traits de ligature (deux ou plus) seront continus. Ce comportement peut être modifié afin de diviser la ligature en sous-groupes grâce à la propriété `subdivideBeams`. Lorsqu'elle est activée, les ligatures seront subdivisées selon un intervalle défini

par `baseMoment` ; il n'y aura alors plus qu'un seul trait de ligature entre chaque sous-groupe. 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]
}
```

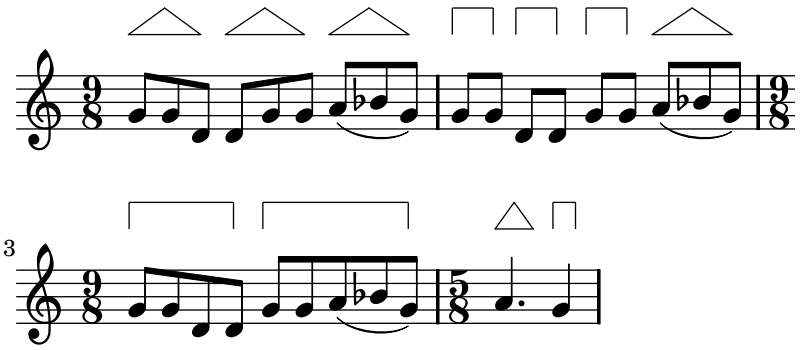


Signes de direction signes de sous-groupe

Les règles de ligature par mesure sont gérées par la propriété `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` lorsque qu'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/beam-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) |
    #(set-time-signature 9 8 '(4 5))
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
```

```
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```
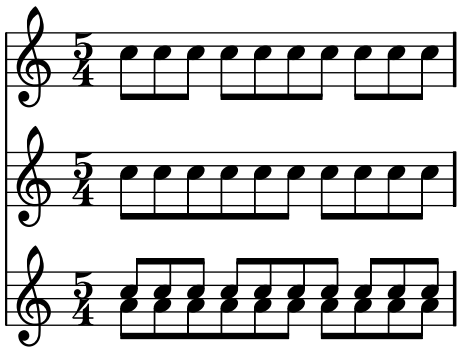


Définition de règles de ligature pour la partition

Les règles de ligatures 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

Fichiers d'initialisation : '`scm/beam-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*.
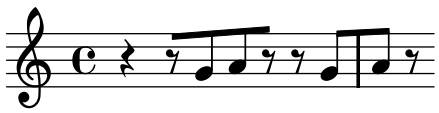
Problèmes connus et avertissements

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 polyphonique, 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 anuellement les dernières ligatures.

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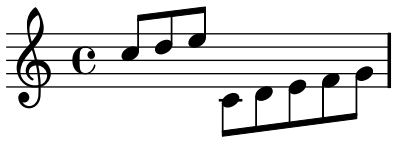
Dans certaines situations, il peut s'avérer nécessaire de supplanter l'algorithme de groupement automatique des notes, par exemple pour prolonger une ligature par-dessus un silence ou une barre de mesure, ou bien pour suivre le rythme des paroles plutôt que celui des notes. Le début et la fin de la ligature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

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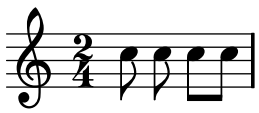


Le positionnement des ligature manuelles se détermine comme pour toute indication attachée à une note :

```
c8^[ d e] c,_[ d e f g]
```
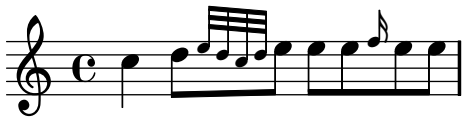


Le fait d'affubler une note particulière d'un `\noBeam` aura pour effet de l'empêcher d'être ligaturée :

```
\time 2/4
c8 c\noBeam c c
```
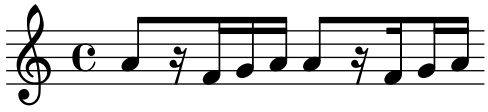


Notes d'ornement et normales font l'objet d'un traitement distinct. Il est dont possible de ligaturer ou non des notes d'ornement sans géner ce qui est en place au niveau de la notation normale.

```
c4 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.

```
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.

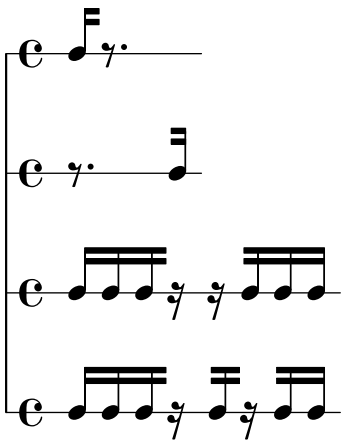
Pour des crochets rectiligne à gauche, c'est `stemRightBeamCount` qu'il faudra déterminer (exemple 2).

Pour que les barres de ligature débordent sur la droite, `stemRightBeamCount` doit avoir une valeur positive ; pour un débrodement 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 `c'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
      c16[]
    }
    % Example 3
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r r
      \set stemLeftBeamCount = #2
      c16 c c
    }
    % Example 4
    \new RhythmicStaff {
      c16 c
      \set stemRightBeamCount = #2
      c16 r
      c16[]
      r16
      \set stemLeftBeamCount = #2
      c16 c
    }
  >>
}
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩, [Notes d'ornement], page 92.

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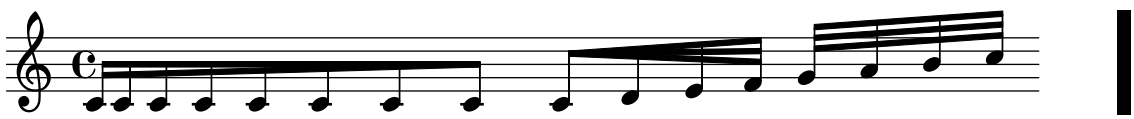
Les ligatures en soufflet permettent d'indiquer qu'un petit groupe de notes se joue en accélérant ou en ralentissant sans pour autant modifier le tempo du morceau. L'étendue du soufflet s'indique par `[` et `]`, et son orientation est déterminée par la proriété `grow-direction` de l'objet `Beam`.

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 première et dernière 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.

```
\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.
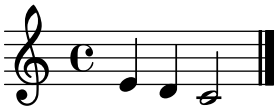
1.2.5 Mesures

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 :

```
e4 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 ""
```

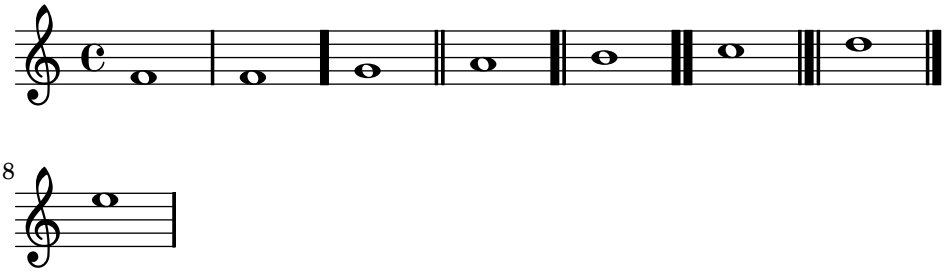
Ceci insérera une barre de mesure invisible, et permettra – sans pour autant le forcer – de sauter de ligne à cet endroit, sans incrémenter le numéro de mesure. Pour forcer le saut de ligne, référez-vous à ⟨undefined⟩ [Sauts de ligne], page ⟨undefined⟩.

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.

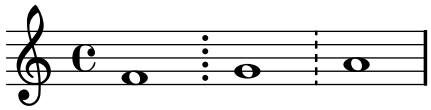
Vous disposez de deux types de barres simples et de cinq différentes doubles barres :

```
f1 \bar "|"
f1 \bar "."
g1 \bar "||"
a1 \bar ".|"
b1 \bar ".|."
c1 \bar "|.|"
d1 \bar "|."
e1
```



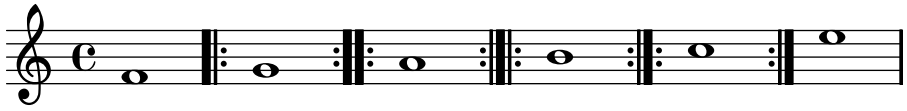
ainsi que d'une barre en pointillé et d'une discontinue :

```
f1 \bar ":"
g1 \bar "dashed"
a1
```



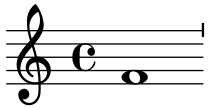
et de cinq types de barre de reprise :

```
f1 \bar "|:"
g1 \bar ":|:"
a1 \bar ":|.|:"
b1 \bar ":|.:"
c1 \bar ":|"
e1
```



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

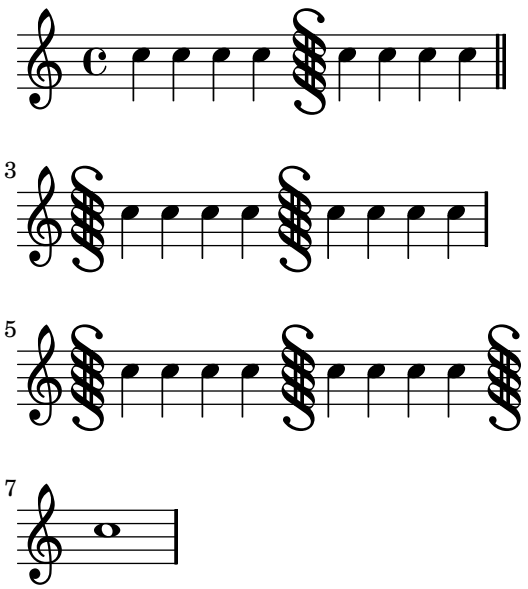
```
f1 \bar "'"
```



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

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 :
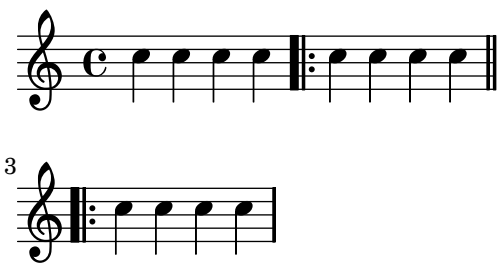
```
c4 c c c
\bar "S"
c4 c c c \break
\bar "S"
c4 c c c
\bar "|S"
c4 c c c \break
\bar "|S"
c4 c c c
\bar "S|"
c4 c c c \break
\bar "S|"
c1
```



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 120) qui se chargeront d'imprimer le type de barre approprié.

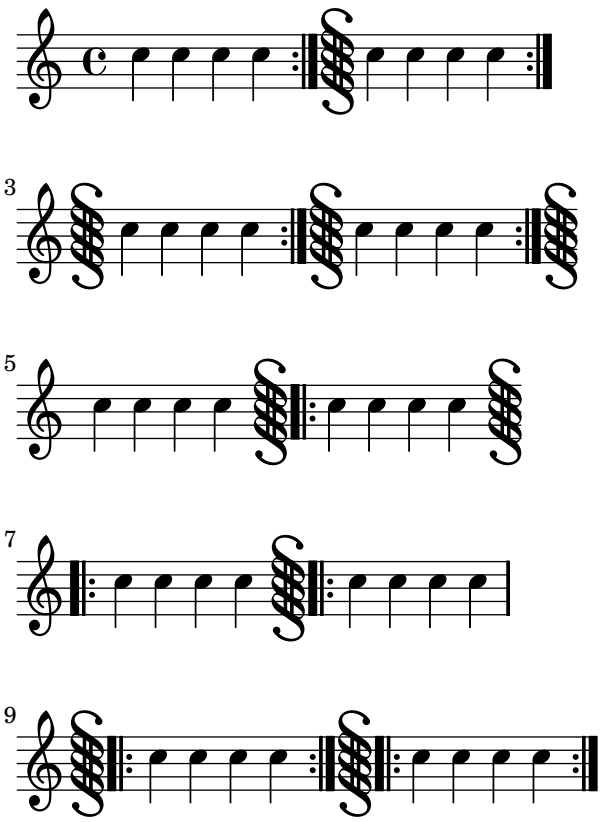
Dans les faits, un `"||:"` équivaut à un `"|:"` sauf s'il intervient à un saut de ligne : une double barre sera alors imprimée en fin de portée, et la barre de reprise au début de la nouvelle.

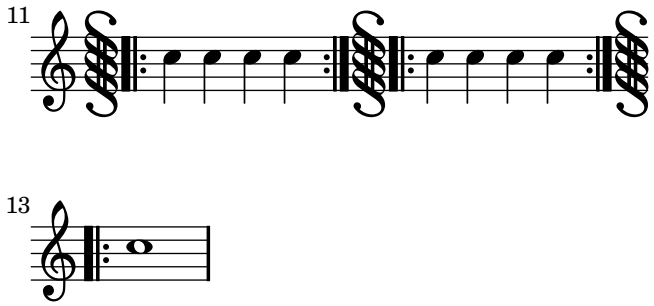
```
\override Score.RehearsalMark #'padding = #3
c4 c c c
\bar "||:"
c4 c c c \break
\bar "||:"
c4 c c c
```



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

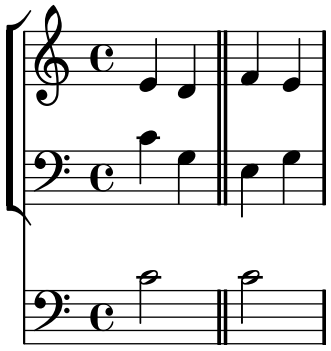
```
c4 c c c
\bar ":|S"
c4 c c c \break
\bar ":|S"
c4 c c c
\bar ":|S."
c4 c c c \break
\bar ":|S."
c4 c c c
\bar "S|:"
c4 c c c \break
\bar "S|:"
c4 c c c
\bar ".S|:"
c4 c c c \break
\bar ".S|:"
c4 c c c
\bar ":|S|:"
c4 c c c \break
\bar ":|S|:"
c4 c c c
\bar ":|S.|:"
c4 c c c \break
\bar ":|S.|:"
c1
```





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 {
      e4 d
      \bar "||"
      f4 e
    }
    \new Staff { \clef bass c4 g e g }
  >>
  \new Staff { \clef bass c2 c2 }
>>
```



Morceaux choisis

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 : ⟨undefined⟩ [Sauts de ligne], page ⟨undefined⟩, Section 1.4 [Répétitions et reprises], page 120, [Regroupement de portées], page 151.

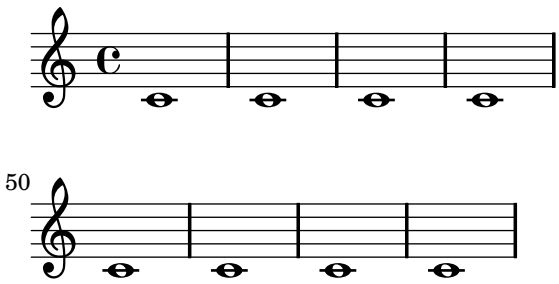
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

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

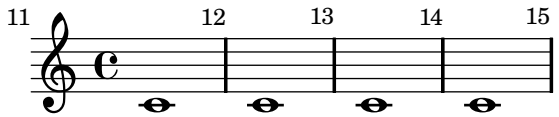
Les numéros de mesure sont imprimés par défaut à chaque début de ligne, sauf la première. Ce nombre est stocké par la propriété `currentBarNumber` qui sera mise à jour à chaque mesure. Vous pouvez aussi le définir de manière arbitraire :

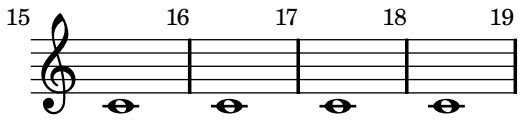
```
c1 c c c
\break
\set Score.currentBarNumber = #50
c1 c c c
```



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`, `visible en début de ligne`. Voici comment imprimer partout les numéros de mesure :

```
\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
```



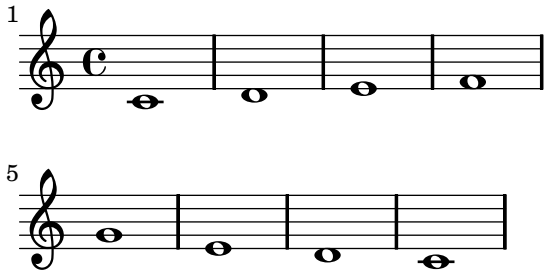


Morceaux choisis

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. À noter 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.

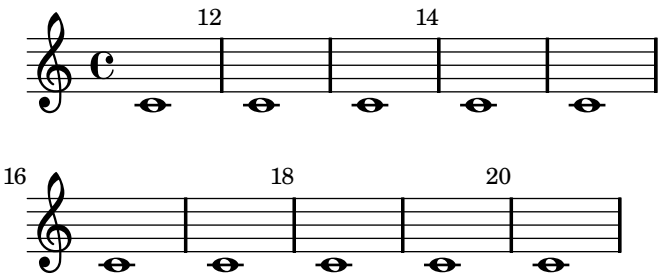
```
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



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 recourrant 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
}
```



Inscrire le numéro de mesure dans un cadre ou un cercle

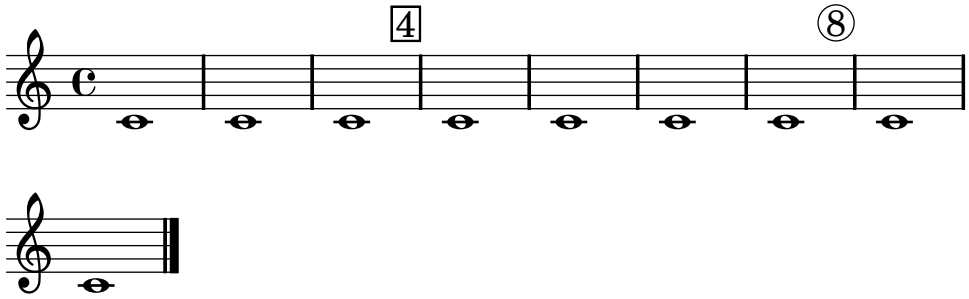
Les numéros de mesures peuvent être encadrés ou entourés d'un cercle.

```
\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 }

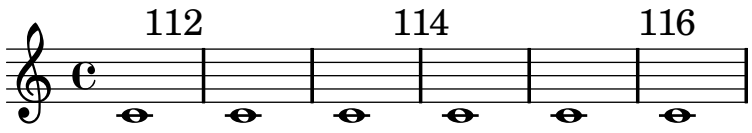
  % Draw a circle round the following bar number(s)
  \override Score.BarNumber #'stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|."
}
```



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
}
```
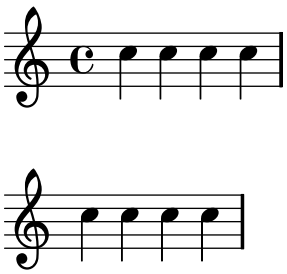


Supprimer les numéros de mesure d'une partition

Désactiver le graveur concerné — `Bar_number_engraver` — donnera une partition — contexte `Score` — sans numéros de mesure.

```
\layout {
  \context {
    \Score
    \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 pripriétés 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 |
```

Le test de mesure peut être aussi utilisé dans les paroles, par exemple :
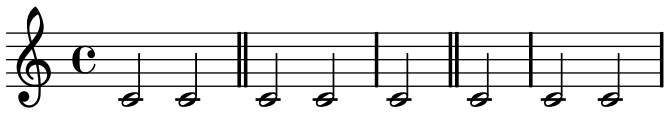
```
\lyricmode {
 \time 2/4
 Twin -- kle | Twin -- kle
}
```

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 occurence est affiché.

Il est aussi possible d'attribuer une autre valeur au symbole `|`, en assignant une expression musicale à `pipeSymbol`. 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.

```
pipeSymbol = \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 123.

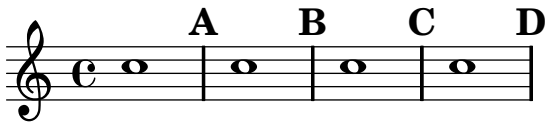
Voir aussi

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

Indications de repère

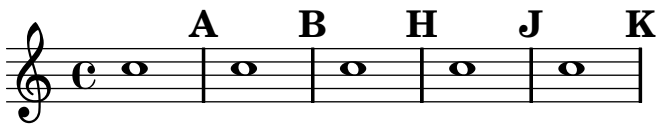
Indiquer un repère s'obtient grâce à la commande `\mark`.

```
c1 \mark \default
c1 \mark \default
c1 \mark \default
c1 \mark \default
```



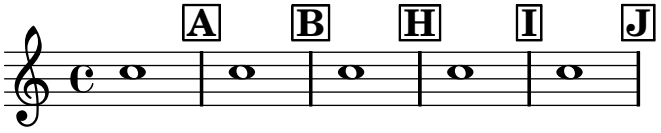
Lorsque vous utilisez `\mark \default`, le repère s'incrémente automatiquement ; toutefois donner un nombre en argument permet de spécifier manuellement le repère en question. La valeur à utiliser est enregistrée dans la propriété `rehearsalMark`.

```
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



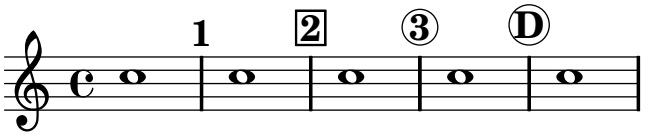
La lettre **I** n'est pas utilisée, conformément aux usages de la gravure. Cependant, vous pourrez intégrer la lettre **I** en utilisant l'une des commandes suivantes selon que ce repère doive être simple, inclus dans un rectangle ou dans un cercle :

```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet
\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 étiquette. 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.

```
\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-numbers` (comportement par défaut), `format-mark-box-numbers`, `format-mark-letters` et `format-mark-box-letters`. 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 :

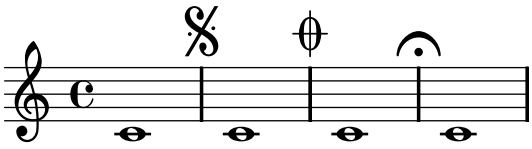
```
\mark "A1"
```

`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 glype musical tel que le signe *segno*.

```
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 ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩.

Pour affiner le positionnement des repères, veuillez vous référer à Section 1.8.2 [Mise en forme du texte], page 198, et tout particulièrement ce qui concerne la `break-alignable-interface` au chapitre ⟨undefined⟩ [Alignement des objets], page ⟨undefined⟩.

Voir aussi

Manuel de notation : ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩, Section 1.8.2 [Mise en forme du texte], page 198, ⟨undefined⟩ [Alignement des objets], page ⟨undefined⟩.

Fichiers d'initialisation : '`scm/translation-functions.scm`' contient les définitions de `format-mark-numbers` et `format-mark-letters`. Elles seront source d'inspiration pour d'autres fonctions de formatage.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.
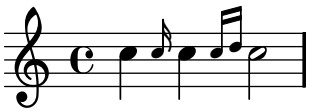
Référence des propriétés internes : Section "MarkEvent" dans *Référence des propriétés internes*, Section "Mark_engraver" dans *Référence des propriétés internes*, Section "RehearsalMark" dans *Référence des propriétés internes*.

1.2.6 Fonctionnalités rythmiques particulières

Notes d'ornement
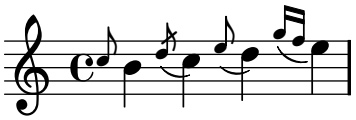
Les petites notes sont des ornements entièrement écrits. Leur taille est un peu plus petite que celle des notes normales et elles n'occupent pas de temps dans la mesure.

```
c4 \grace c16 c4
\grace { c16[ d16] } c2
```
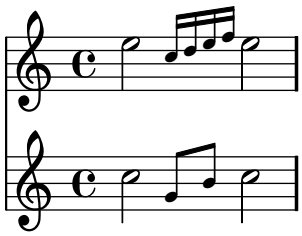


Les plus courantes sont les acciaccatures, qui doivent se jouer très vite, et qui s'écrivent sous forme d'une petite note barrée (sur la hampe) et liée. L'appoggiature est une petite note non barrée, qui vole une fraction à la durée de la note réelle qui la suit.

```
\grace c8 b4
\acciaccatura d8 c4
\appoggiatura e8 d4
\acciaccatura { g16[ f] } e4
```
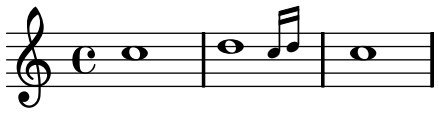


Les petites notes se placent de façon synchrone entre les différentes portées. Dans l'exemple suivant, il y a deux petites double-croches pour chaque petite croche.

```
<<
  \new Staff { e2 \grace { c16[ d e f] } e2 }
  \new Staff { c2 \grace { g8[ b] } c2 }
>>
```



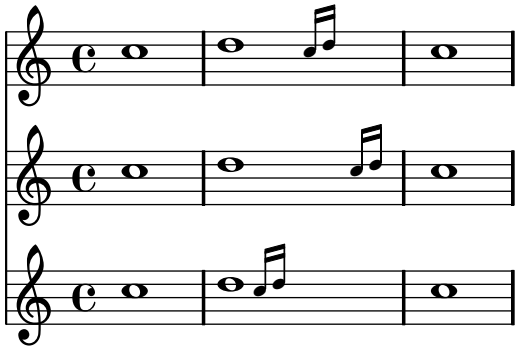
La commande `\afterGrace` sert à placer une petite note après une note réelle – et non *avant* comme d'ordinaire. Cette commande requiert deux arguments : la note réelle, et la ou les petites notes qui suivent.

```
c1 \afterGrace d1 { c16[ d] } c1
```
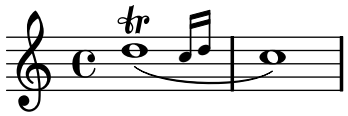


Les petites notes se placent alors aux 3/4 de la durée de la note réelle. Cette fraction peut être changée en définissant `afterGraceFraction`. Dans l'exemple suivant, vous pouvez observer la différence entre le comportement par défaut, à 15/16 et enfin à la moitié de la durée de base.

```
<<
  \new Staff {
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 15 16))
    c1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff {
    #(define afterGraceFraction (cons 1 2))
    c1 \afterGrace d1 { c16[ d] } c1
  }
>>
```



Des silences invisibles permettent d'ajuster l'espace entre les petites notes et la note réelle. Nous pourrions positionner ces petites notes à 7/8 de la durée de la note de base :

```
\new Voice {
  <<
    { d1^\trill_( }
    { s2 s4. \grace { c16[ d] } }
  >>
  c1)
}
```



Les expressions `\grace` obéissent à des règles typographiques particulières, notamment pour régler l'orientation et la 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 ajoutés doivent également être désactivés dans cette même expression.

```
\new Voice {
  \acciaccatura {
    \stemDown
    f16->
    \stemNeutral
  }
  g4 e c2
}
```
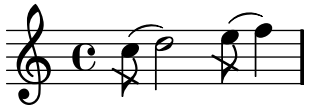


Morceaux choisis

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.
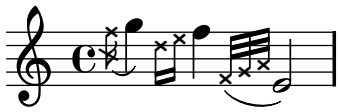
```
\relative c'' {
  \override Stem #'stroke-style = #"grace"
  c8( d2) e8( f4)
}
```



Mise en forme des notes d'ornement

Il est possible de changer globalement la mise en forme des petites notes dans un morceau, au moyen de la fonction `add-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
    }
  }
}
```
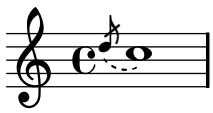


Redéfinition des réglages de mise en forme par défaut des notes d'ornement

Vous pouvez modifier les valeurs des variables `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic`, et `stopAppoggiaturaMusic` afin d'en personnaliser les effets. Pour plus de détails, voir le fichier `ly/grace-init.ly`.

```
startAcciaccaturaMusic = {
  s1*0(
  \override Stem #'stroke-style = #"grace"
  \slurDashed
}

stopAcciaccaturaMusic = {
  \revert Stem #'stroke-style
  \slurSolid
  s1*0)
}

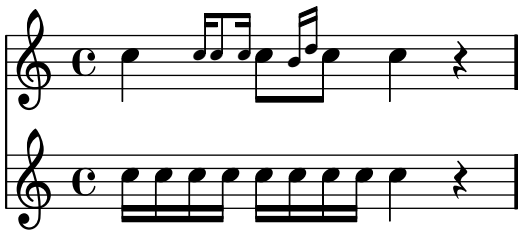
\relative c'' {
  \acciaccatura d8 c1
}
```



Positionnement des notes d'ornement avec espace flottant

Lorsque vous activez 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 c4 r
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “ornements” dans *Glossaire*, Section “acciacature” dans *Glossaire*, Section “appogiature” dans *Glossaire*.

Fichiers d’initialisation : ‘`ly/grace-init.ly`’.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.
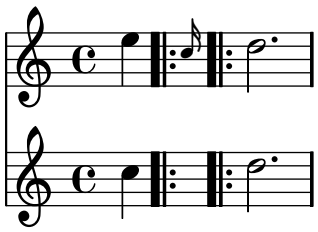
Référence des propriétés internes : Section “GraceMusic” dans *Référence des propriétés internes*, Section “Grace_beam_engraver” dans *Référence des propriétés internes*, Section “Grace_engraver” dans *Référence des propriétés internes*, Section “Grace_spacing_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un groupe de notes ligaturées consituant une *acciaccatura* apparaîtra comme une *appoggiatura*, c’est-à-dire sans trait.
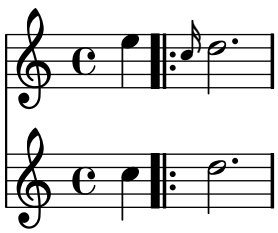
La synchronisation des petites notes se fait de façon parfois surprenante, car les autres objets de la portée – barre de mesure, armure, etc. – sont eux aussi synchrones. Pensez-y lorsque vous mélez des portées comprenant des petites notes et d’autres sans :

```
<<
  \new Staff { e4 \bar "|:" \grace c16 d2. }
  \new Staff { c4 \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 { e4 \bar "|:" \grace c16 d2. }
  \new Staff { c4 \bar "|:" \grace s16 d2. }
>>
```



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.

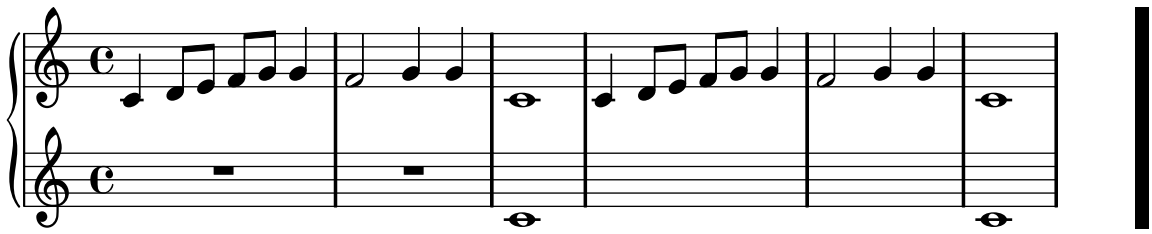
Alignement et cadences

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 résoudre ce problème. Ces fonctions Scheme prennent en argument un fragment de musique, et génèrent un `\skip` ou un silence multi-mesures d'une durée correspondant à ce fragment.

```
MyCadenza = \relative c' {
  c4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(ly:export (mmrest-of-length MyCadenza))
    c'1
    #(ly:export (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'existance 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 86.
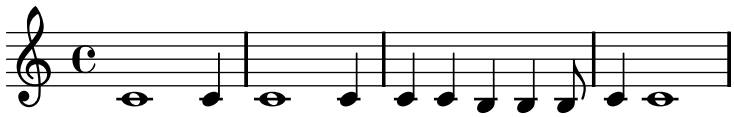
`measureLength`
: La longueur de la mesure, dans la métrique en cours. Pour une mesure à 4/4, elle est de 1, et de 3/4 pour une mesure à 6/8. Sa valeur détermine où peut s'insérer une barre et comment seront générées les ligatures automatiques.

`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 c' {
  \set Timing.measureLength = #(ly:make-moment 5 4)
  c1 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 m` construit une durée de n/m 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.
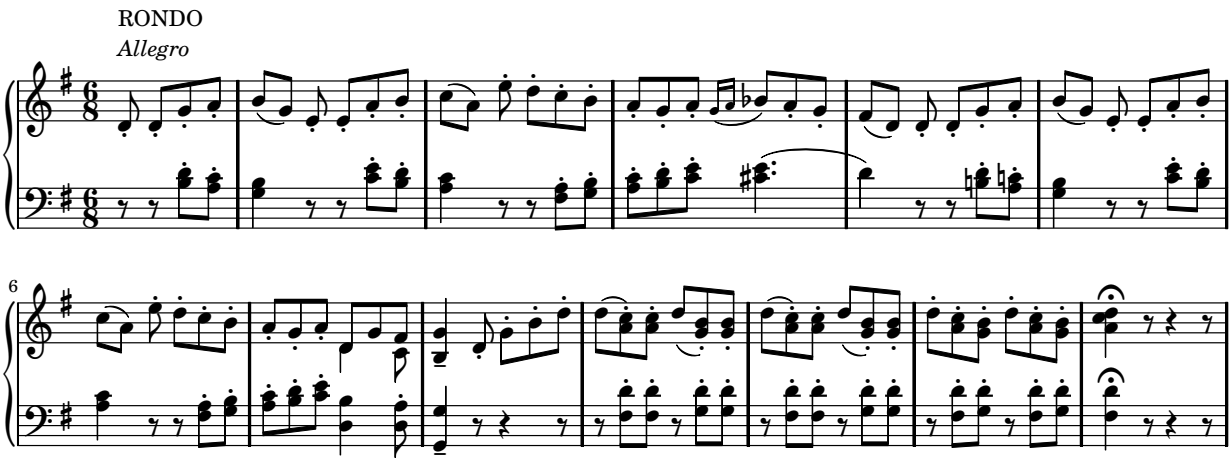
Voir aussi

Manuel de notation : [Numéros de mesure], page 86, [Musique sans métrique], page 61

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.
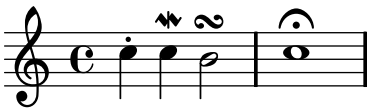
Articulations et ornements

Les différents symboles qui indiquent des ponctuations ou des modes de jeu différents s'ajoutent aux notes de la manière suivante :

```
note\nom
```
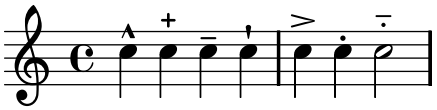
Les valeurs de *nom* sont répertoriées dans l'annexe ⟨undefined⟩ [Liste des signes d'articulation], page ⟨undefined⟩. En voici un exemple :

```
c4\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.

```
c4-^  c-+  c--  c-|
c4->  c-.  c2-_
```
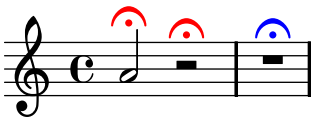


Même si LilyPond place automatiquement ces symboles, selon les règles contenues dans le fichier `scm/script.scm`, il est possible de l'obliger à les positionner au-dessus ou en-dessous de la note, comme indiqué au chapitre ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

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*.

Une articulation peut se rattacher aussi bien à un silence qu'à une note, mais **pas à un silence multi-mesures**. Il existe cependant un cas particulier : le point d'orgue – ou point d'arrêt – attaché à un silence valant l'intégralité de la mesure. LilyPond dispose à cet effet de la commande `\fermataMarkup`, qui créera un objet `MultiMeasureRestText` rattaché à ce « silence multi-mesures ».

```
\override Script #'color = #red
\override MultiMeasureRestText #'color = #blue
a2\fermata r\fermata
R1\fermataMarkup
```



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 189.

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
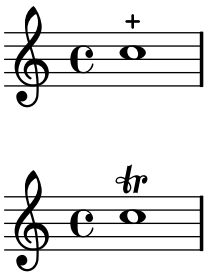
Modification de la signification des raccourcis pour les signes d'articulation

Les raccourcis sont répertoriés dans le fichier `ly/script-init.ly`, dans lequel on retrouve les variables `dashHat`, `dashPlus`, `dashDash`, `dashBar`, `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-+ }
```
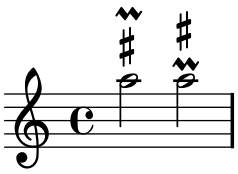


Contrôle de l'ordre vertical des articulations et ornements

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 il est donc placé plus près de la note ; ensuite c'est l'objet `Script` – le mordant – qui a la propriété la plus basse, et il 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 }
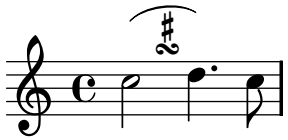
  \once \override Script #'script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



Création d'un grupetto retardé

Obtenir un *grupetto* retardé et dans lequel la note la plus basse est altérée requiert quelques surcharges. La propriété `outside-staff-priority` doît être désactivée (`#f`) pour éviter qu'elle prenne le pas sur la propriété `avoid-slur`. La valeur assignée à `halign` permet de gérer horizontalement le « retard ».

```
\relative c'' {
  \once \override TextScript #'avoid-slur = #'inside
  \once \override TextScript #'outside-staff-priority = ##f
  c2(^\markup \tiny \override #'(baseline-skip . 1) {
    \halign #-4
    \center-column {
      \sharp
      \musicglyph #"scripts.turn"
    }
  }
  d4.) c8
}
```



Voir aussi

Glossaire musicologique : Section “tenuto” dans *Glossaire*, Section “accent” dans *Glossaire*, Section “staccato” dans *Glossaire*, Section “portato” dans *Glossaire*.

Manuel d'initiation : Section “Positionnement des objets” dans *Manuel d'initiation*.

Manuel de notation : [Commentaires textuels], page 189, ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩, ⟨undefined⟩ [Liste des signes d'articulation], page ⟨undefined⟩, [Trilles], page 118.

Fichiers installés : `scm/script.scm`.

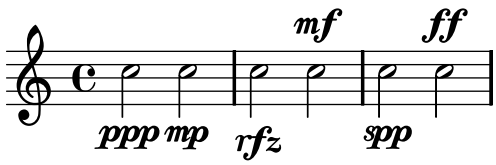
Morceaux choisis : Section “Signes d'interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Script” dans *Référence des propriétés internes*, Section “TextScript” dans *Référence des propriétés internes*.

Nuances

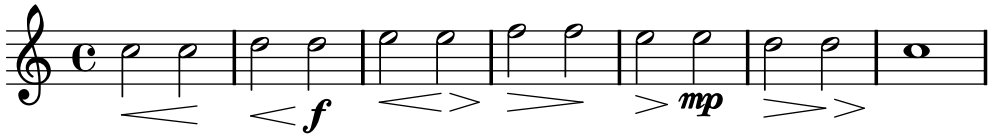
À chaque nuance absolue correspond une commande qui peut être indiquée après une note : `c4\ff` par exemple. Les commandes de nuance disponibles sont `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, et `\rfz`. Les nuances se placent aussi bien en dessous qu'au-dessus de la portée ; pour plus d'information, consultez ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

```
c2\ppp c\mp
c2\rfz c^\mf
c2_\spp c^\ff
```
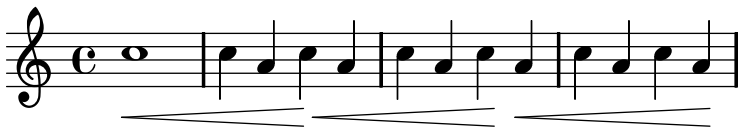


Un crescendo est délimité par `\<` et `\!`, ou peut se terminer par une commande de nuance explicite, ou bien un decrescendo ou un nouveau crescendo. Il en va de même pour un diminuendo. Au lieu de `\<` et `\>`, vous pouvez utiliser `\cr` et `\decr`, auquel cas LilyPond n'imprimera pas de soufflet (*hairpin* en anglais).

```
c2\< c\!
d2\< d\f
e2\< e\>
f2\> f\!
e2\> e\mp
d2\> d\>
c1\!
```
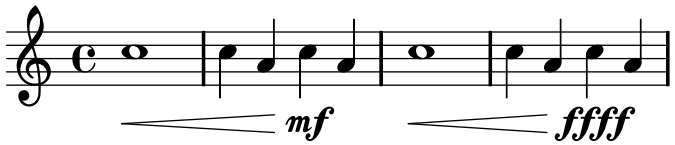


Un soufflet terminé par un simple `\!` prendra fin sur la droite de la tête de note à laquelle il est attaché. Dans le cas où il se termine par l'intervention d'un autre soufflet (contraire ou non), il prendra fin au milieu de la tête de note affublée d'un `\<` ou d'un `\>`, et le nouveau soufflet débutera à l'extrémité droite de cette même tête de note.

```
c1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
```



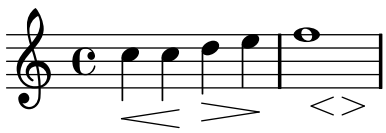
Il en va de même lorsqu'un soufflet est interrompu par une nuance explicite. Notez bien que la largeur occupée par cette nuance explicite influe sur la terminaison du soufflet :

```
c1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
```
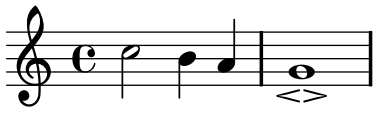


Les indications de nuance sont attachées aux notes ; aussi, lorsque l'on veut faire se succéder plusieurs nuances pendant une note tenue, il faudra avoir recours à des silences invisibles :

```
c4\< c\! d\> e\!
<< f1 { s4 s4\< s4\> s4\! } >>
```
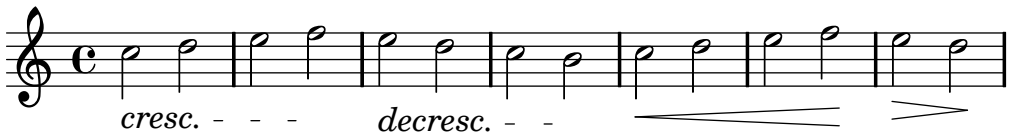


On peut avoir recours à l'indication `\espressivo` pour indiquer un crescendo suivi d'un decrescendo sur une même note. Gardez à l'esprit qu'il s'agit d'une articulation, et en aucun cas d'une nuance.

```
c2 b4 a
g1\espressivo
```



Au lieu d'une notation graphique des (de)crescendos, on peut utiliser une notation textuelle. Une ligne pointillée marque alors son étendue. Les commandes prédéfinies `\crescTextCresc`, `\dimTextDecresc`, `\dimTextDecr`, et `\dimTextDim` remplaceront les commandes `\<` et `\>` par le texte correspondant. `\crescHairpin` et `\dimHairpin` permettent de revenir à des nuances en soufflet.

```
\crescTextCresc
c2\< d | e f\!
\dimTextDecresc
e2\> d | c b\!
\crescHairpin
c2\< d | e f\!
\dimHairpin
e2\> d\!
```



Pour créer des indications de nuance qui restent alignées avec les nuances habituelles, reportez-vous au chapitre [Personnalisation des indications de nuance], page 106.

Le positionnement vertical des nuances est géré par le Section "DynamicLineSpanner" dans *Référence des propriétés internes*.

Commandes prédéfinies

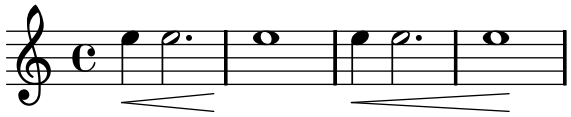
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

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` :
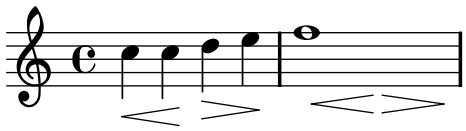
```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin #'to-barline = ##f
  e4\< e2.
  e1\!
}
```



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.
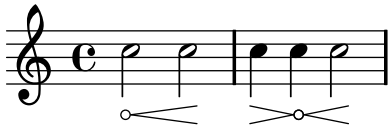
```
\relative c'' {
  c4\< c\! d\> e\!
  \override Hairpin #'minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```



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\!
}
```



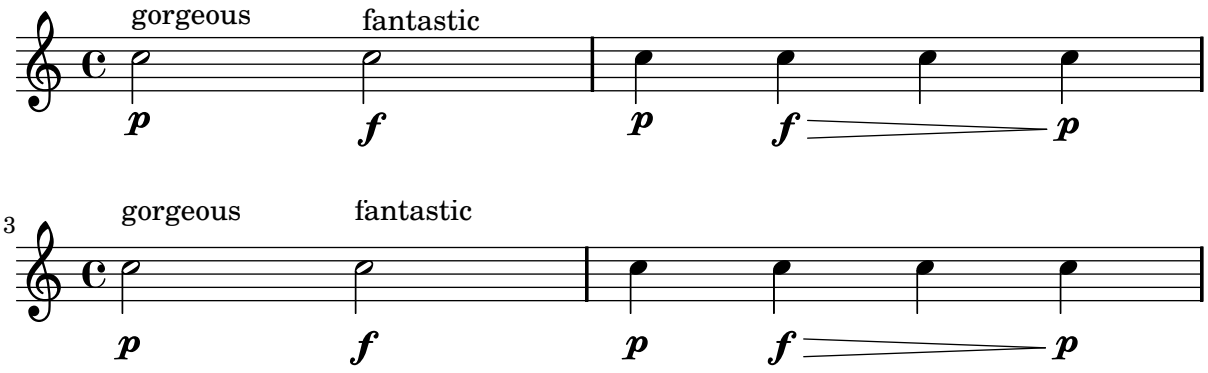
Alignement vertical des nuances indications textuelles

Vous pourrez, en jouant sur la propriété `'Y-extent`, aligner les différents objets `DynamicLineSpanner` (souflets ou textuels) quelle que soit leur étendue, par rapport à un même point de référence. Tous les éléments seront alors centrés sur une même ligne, ce qui sera visuellement plus agréable.

C'est le même principe qui sert à aligner les indications textuelles sur une ligne de référence.

```
music = \relative c'' {
  c2\p^\markup { gorgeous } c\f^\markup { fantastic }
  c4\p c\f\> c c\!\p
}

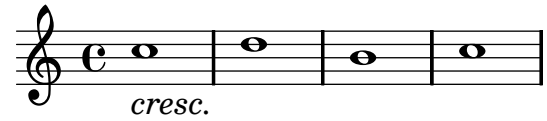
{
  \music \break
  \override DynamicLineSpanner #'staff-padding = #2.0
  \override DynamicLineSpanner #'Y-extent = #'(-1.5 . 1.5)
  \override TextScript #'Y-extent = #'(-1.5 . 1.5)
  \music
}
```



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 avec :
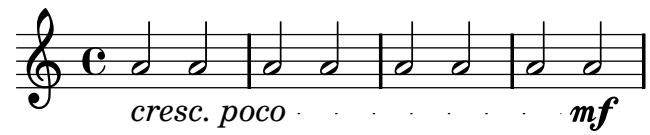
```
\relative c'' {
  \override DynamicTextSpanner #'style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



Modifidation 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 : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩, [Personnalisation des indications de nuance], page 106, ⟨undefined⟩ [Contenu de la sortie MIDI], page ⟨undefined⟩, ⟨undefined⟩ [Gestion des nuances en MIDI], page ⟨undefined⟩.

Morceaux choisis : Section “Signes d’interprétation” dans *Morceaux choisis*.
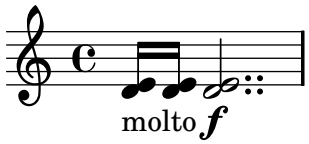
Référence des propriétés internes : Section “DynamicText” dans *Référence des propriétés internes*, Section “Hairpin” dans *Référence des propriétés internes*, Section “DynamicLineSpanner” dans *Référence des propriétés internes*.

Personnalisation des indications de nuance

La manière la plus simple de personnaliser une indication de nuance consiste à utiliser un objet `\markup`.

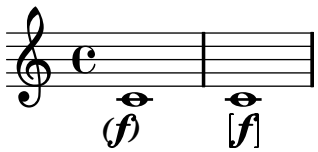
```
moltoF = \markup { molto \dynamic f }

\relative c' {
  <d e>16_\moltoF <d e>
  <d e>2..
}
```
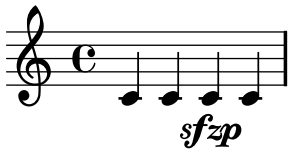


Vous pouvez créer des indications de nuance éditoriales (entre parenthèses ou crochets) grâce aux étiquettes (*mode markup*) ; la syntaxe en est abordée au chapitre Section 1.8.2 [Mise en forme du texte], page 198.

```
roundF = \markup {
    \center-align \concat { \bold { \italic ( }
           \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative c' {
  c1_\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.

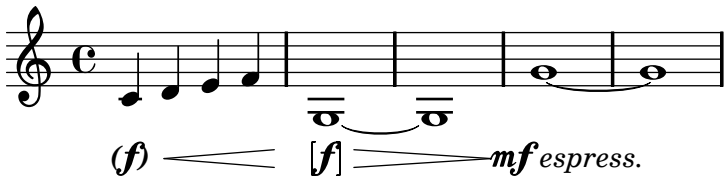
```
sfzp = #(make-dynamic-script "sfzp")
\relative c' {
  c4 c c\sfzp c
}
```



`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

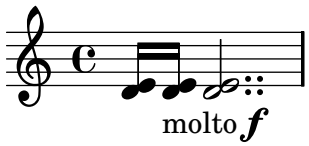
possèdent des propriétés particulières et prédéfinies quant à leur police. Aussi, lorsque vous créez du texte en pareille situation, nous vous recommandons d'utiliser `\normal-text` pour annuler ces propriétés. L'intérêt majeur de recourrir à la fonction `make-dynamic-script` plutôt qu'à un simple *markup* 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' {
  c4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g
  g'~\mfEspressDynamic
  g
}
```



La constrution d'une indication de nuance personnalisée peut aussi se faire en langage Scheme ; voir Section "Construction Scheme d'un marqueur" dans *Extension de LilyPond* pour en connaître les modalités.

```
moltoF = #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative c' {
  <d e>16 <d e>
  <d e>2..\moltoF
}
```



L'utilisation des fontes en mode *markup* est abordée au chapitre [Sélection de la fonte et de la taille], page 200.

Voir aussi

Manuel de notation : Section 1.8.2 [Mise en forme du texte], page 198, [Sélection de la fonte et de la taille], page 200, ⟨undefined⟩ [Contenu de la sortie MIDI], page ⟨undefined⟩, ⟨undefined⟩ [Gestion des nuances en MIDI], page ⟨undefined⟩.

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

Manuel d’extension : Section “Construction Scheme d’un marqueur” dans *Extension de LilyPond*.

1.3.2 Signes d’interprétation sous forme de courbe

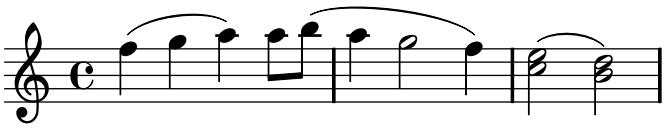
Ce chapitre traite des signes d’interprétation imprimés sous forme de courbe : liaisons d’articulation ou de phrasé, respirations, chutes et sauts.

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 doît se terminer dans la voix où elle a été entamée.

```
f4( g a) a8 b(
a4 g2 f4)
<c e>2( <b d>2)
```
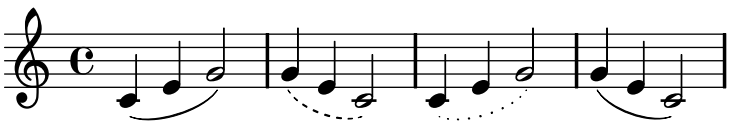


Vous pouvez décider de l’orientation des liaisons par rapport à la portée, comme indiqué au chapitre ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

Une seule liaison d’articulation peut être imprimée à la fois. S’il est nécessaire d’imprimer une liaison plus longue, englobant des liaisons plus courtes, utilisez des [Liaisons de phrasé], page 110.

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 :
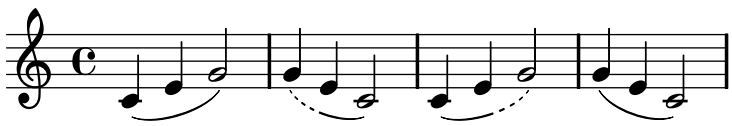
```
c4( e g2)
\slurDashed
g4( e c2)
\slurDotted
c4( e g2)
\slurSolid
g4( e c2)
```



En utilisant `\slurHalfDashed`, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L’inverse s’obtient avec `\slurHalfSolid`.
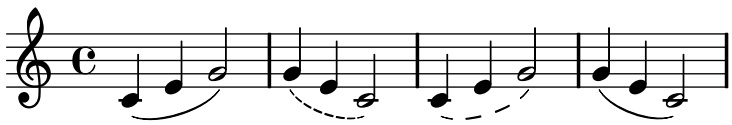
```
c4( e g2)
\slurHalfDashed
g4( e c2)
\slurHalfSolid
c4( e g2)
\slurSolid
```

```
g4( e c2)
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
c4( e g2)
\slurDashPattern #0.7 #0.75
g4( e c2)
\slurDashPattern #0.5 #2.0
c4( e g2)
\slurSolid
g4( e c2)
```



Commandes prédéfinies

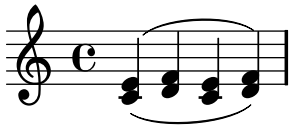
`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

Morceaux choisis

Accords et double liaison d'articulation

Certains auteurs utilisent deux liaisons lorsqu'ils veulent lier des accords. Dans LilyPond, il faut pour cela activer la propriété `doubleSlurs` :
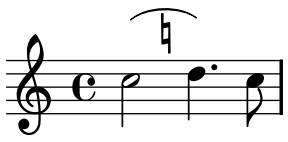
```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```



Positionnement d'une annotation à l'intérieur d'une liaison

Lorsqu'il vous faut inscrire une annotation à l'intérieur d'une liaison, la propriété `outside-staff-priority` doît être désactivée.

```
\relative c'' {
  \override TextScript #'avoid-slur = #'inside
  \override TextScript #'outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```
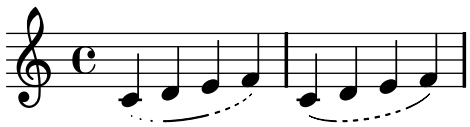


Dessin d'une liaison d'articulation au trait discontinu

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 : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩, [Liaisons de phrasé], page 110.
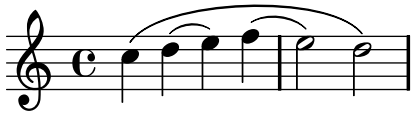
Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

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.

```
c4\( d( e) f(
e2) d\)
```
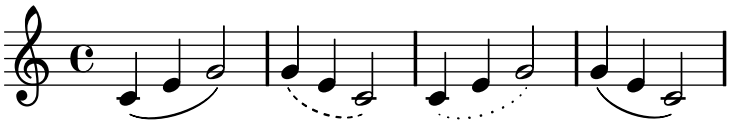


D'un point de vue typographique, rien ne distingue une liaison de phrasé d'une liaison d'articulation. Cependant, LilyPond les considère comme des objets différents. Une commande `\slurUp` n'affectera donc pas une liaison de phrasé. Vous pouvez décider de l'orientation des liaisons de phrasé par rapport à la portée, comme indiqué au chapitre ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

Il n'est pas possible d'avoir plusieurs liaisons de phrasé en même temps.
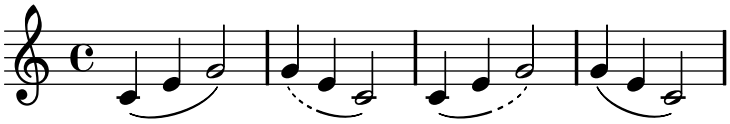
Une liaison est par défaut dessinée d'un trait plein. Il est aussi possible de l'imprimer sous la forme de tirets ou en pointillé :

```
c4\( e g2\)
\phrasingSlurDashed
g4\( e c2\)
\phrasingSlurDotted
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```
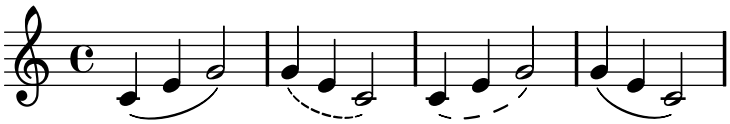


En utilisant `\phrasingslurHalfDashed`, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L'inverse s'obtient avec `\phrasingslurHalfSolid`.

```
c4\( e g2\)
\phrasingSlurHalfDashed
g4\( e c2\)
\phrasingSlurHalfSolid
c4\( e g2\)
\phrasingSlurSolid
g4\( e c2\)
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
c4\( e g2\)
\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 108.

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 : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩, [Liaisons d'articulation], page 108.
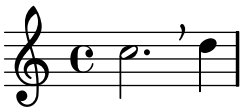
Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

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

Signes de respiration
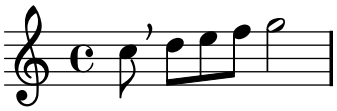
Les indications de respiration sont indiquées par la commande `\breathe`.

```
c2. \breathe d4
```



Un signe de respiration interrompt obligatoirement les ligatures, même automatiques. Pour passer outre ce fonctionnement, voir [Barres de ligature manuelles], page 77.

```
c8 \breathe d e f g2
```



LilyPond gère les *divisiones*, signes utilisés en notation ancienne pour indiquer les respirations. Pour de plus amples détails, voir ⟨undefined⟩ [Divisions], page ⟨undefined⟩.

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
}
```
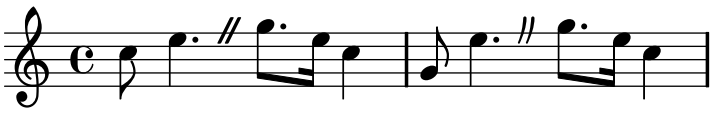


Insertion d'une césure

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 : ⟨undefined⟩ [Divisions], page ⟨undefined⟩.

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.

```
c2-\bendAfter #+4
c2-\bendAfter #-4
c2-\bendAfter #+6.5
c2-\bendAfter #-6.5
c2-\bendAfter #+8
c2-\bendAfter #-8
```



Lorsque vous saisissez des chutes ou des sauts, **n’oubliez pas** le tiret (`-`) qui précède la commande `\bendAfter`.

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*.
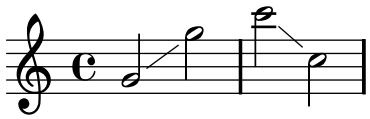
1.3.3 Signes d’interprétation sous forme de ligne

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.

```
g2\glissando g'
c2\glissando c,
```
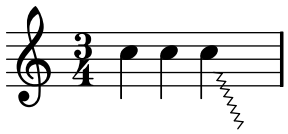


Un glissando est indiqué graphiquement, par une ligne ou des vaguelettes – voir ⟨undefined⟩ [Styles de ligne], page ⟨undefined⟩.

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 "|"
}
```



Voir aussi

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

Manuel de notation : ⟨undefined⟩ [Styles de ligne], page ⟨undefined⟩.

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

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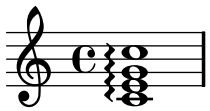
Problèmes connus et avertissements

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` :

```
<c e g c>1\arpeggio
```



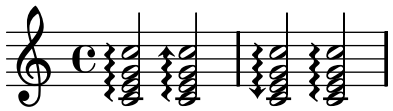
LilyPond dispose de différents graphismes pour indiquer un arpège ; `\arpeggioNormal` reviendra au style par défaut.

```
<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
```



Des crochets indiquent que l’accord devra être plaqué et non arpégé :

```
<c e g c>2
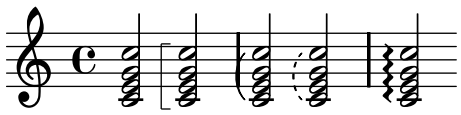
\arpeggioBracket
<c e g c>2\arpeggio

\arpeggioParenthesis
<c e g c>2\arpeggio

\arpeggioParenthesisDashed
<c e g c>2\arpeggio

\arpeggioNormal
```

```
<c e g c>2\arpeggio
```



Les indications d'arpeggio peuvent se présenter sous la forme de ligne discontinue à l'aide de la propriété `'dash-details`. Pour plus de détails à ce propos, consultez [Liaisons d'articulation], page 108.

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 44.

Commandes prédéfinies

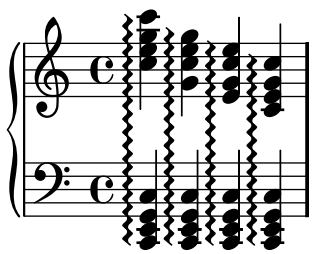
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`.

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
    }
  }
>>
```

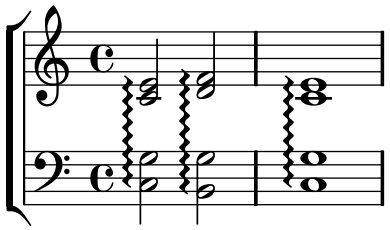


Arpège distribué pour un autre contexte que le piano

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
```
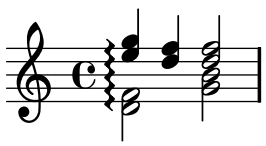
```
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
      \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 108, [Liaisons de prolongation], page 44.

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

Référence des propriétés internes : Section “Arpeggio” dans *Référence des propriétés internes*, Section “Slur” dans *Référence des propriétés internes*, Section “PianoStaff” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Il est impossible de mêler au même instant, dans un contexte `PianoStaff`, des lignes d’arpèges connectées et d’autres non connectées.

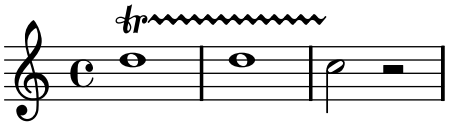
Il n’est pas possible d’imprimer des lignes d’arpège connectées sous forme de parenthèse.

Trilles

Les trilles brefs s’indiquent comme n’importe quelle ponctuation, avec un simple `\trill` ; voir [Articulations et ornements], page 99.

Les trilles plus longs sont délimités par `\startTrillSpan` et `\stopTrillSpan` :

```
d1\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 rappellées sur la première note de la nouvelle ligne :

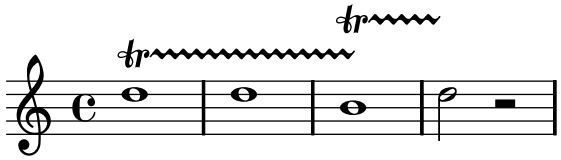
```
d1\startTrillSpan
\break
d1
c2\stopTrillSpan r2
```
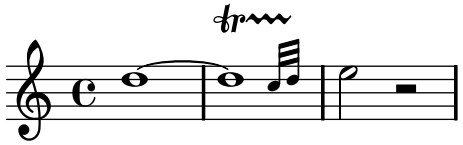




Lorsque des trilles interviennent sur une succession de hauteurs différentes, point n’est besoin d’expliciter la commande `\stopTrillSpan` puisque l’apparition d’un nouveau trille interrompt de fait celui qui le précédait :

```
d1\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 92.

```
d1~\afterGrace
d1\startTrillSpan { c32[ d]\stopTrillSpan }
e2 r2
```
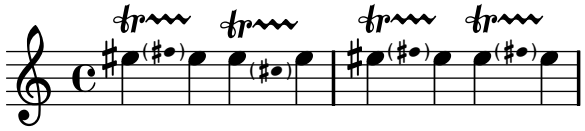


Les trilles qui font intervenir une hauteur précise peuvent être indiqués par la commande `pitchedTrill`. Le premier argument est la note réelle ; le second est une hauteur qui sera imprimée comme une tête de note noire entre parenthèses.

```
\pitchedTrill
e2\startTrillSpan fis
d2 c2\stopTrillSpan
```



Dans l'exemple suivant, le second trille de la deuxième mesure est ambigu – le fa qui est forcé n'est pas diésé. Pour lever toute ambiguïté, il suffit de forcer l'impression de l'altération en ajoutant un `!` à la note considérée.

```
\pitchedTrill
eis4\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
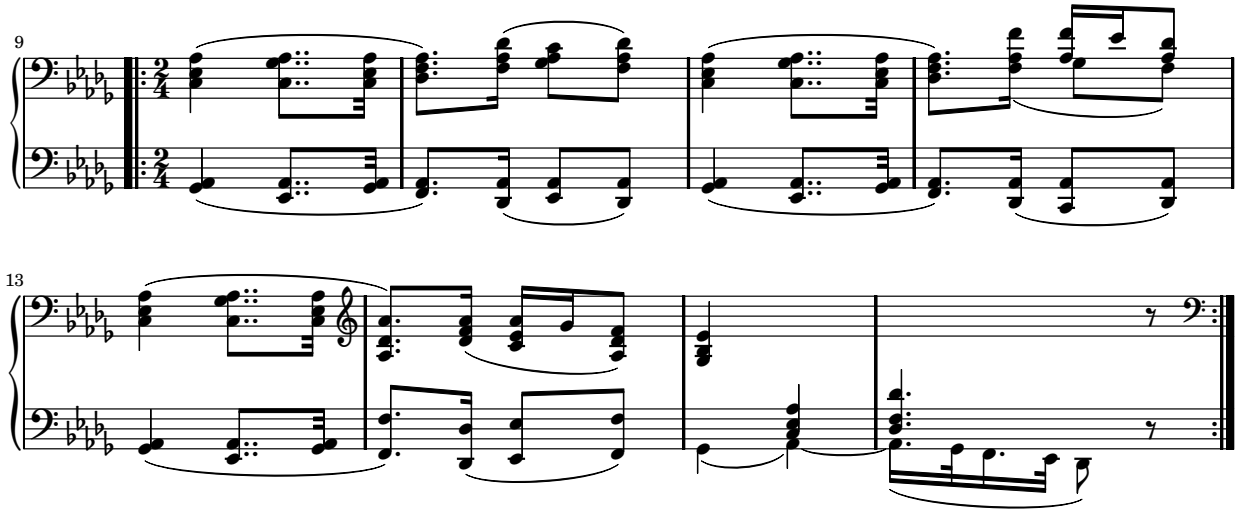
Glossaire musicologique : Section “trille” dans *Glossaire*.

Manuel de notation : [Articulations et ornements], page 99, [Notes d’ornement], page 92.

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` Le passage répété n’est pas développé, mais il est encadré par des barres de reprise et peut se terminer par plusieurs fins alternatives – ou *volte* – imprimées de gauche à droite sous des crochets. Lorsque la répétition commence au début de la pièce, aucune barre de reprise n’est gravée au début de la partition. Il s’agit de la notation courante des reprises avec fins alternatives.

`unfold` La musique répétée est développée dans la partition autant de fois qu’indiqué. Ceci est particulièrement utile pour de la musique répétitive.

`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.

1.4.1 Répétition d’un long passage

Cette section présente la syntaxe des répétitions longues – c’est-à-dire plusieurs mesures. Ces répétitions peuvent prendre deux formes : encadrées par des barres de reprises, ou bien développées dans la partition. Les barres et autres signes de reprise peuvent être contrôlés manuellement.

Répétitions courantes
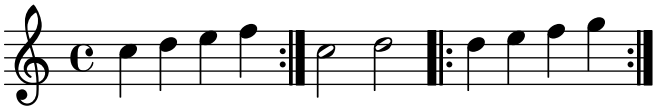
On peut indiquer une répétition de la façon suivante :

```
\repeat volta nombre_de_fois expression_musicale
```

où *`expression_musicale`* représente ce qui doît être répété.

Les reprises courantes, sans alternative, s’indiquent comme ceci :

```
\repeat volta 2 { c4 d e f }
c2 d
\repeat volta 2 { d4 e f g }
```
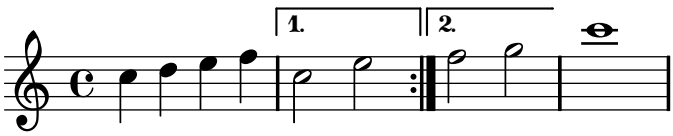


On peut ajouter une fin alternative à l'aide de la commande `\alternative`. Chaque *`alternative`* est une expression musicale en elle-même ; il faudra donc les regrouper par des accolades.

```
\repeat volta nombre_de_fois expression_musicale
\alternative {
  { expression_musicale }
}
```

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.

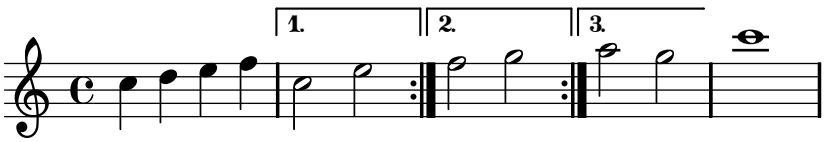
Voici une simple reprise avec une fin alternative :

```
\repeat volta 2 { c4 d e f }
\alternative {
  { c2 e }
  { f2 g }
}
c1
```



Et une répétition avec plusieurs alternatives :

```
\repeat volta 3 { c4 d e f }
\alternative {
  { c2 e }
  { f2 g }
  { a2 g }
}
c1
```
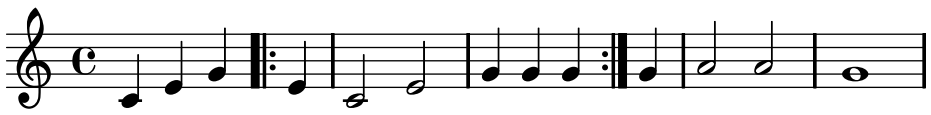


Note : Une clause `\relative` ne doît jamais se trouver à l'intérieur d'une section `\repeat` : vous aurez immanquablement des portées parasites. Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Lorsqu'une reprise sans fins alternatives 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
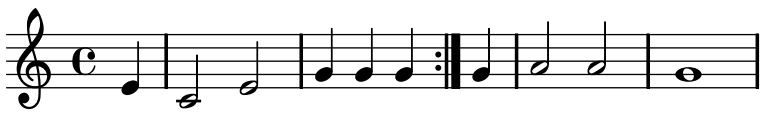
cas, les indications de reprise ne constituent pas des barres de mesure à proprement parler ; il n'est donc pas nécessaire de faire appel à la commande `\partial` ou à des contrôles d'intégrité de mesure.

```
% no \partial here
c4 e g  % no bar check here
% no \partial here
\repeat volta 4 {
  e4 |
  c2 e |
  % no \partial here
  g4 g g  % no bar check here
}
% no \partial here
g4 |
a2 a |
g1 |
```



Il est possible de créer des reprises en début de morceau avec une levée. Le cas est similaire à ce que nous venons de voir. Toutefois, l'utilisation d'un `\partial` est ici nécessaire pour respecter cette entâme.

```
\partial 4  % required
\repeat volta 4 {
  e4 |
  c2 e |
  % no \partial here
  g4 g g  % no bar check here
}
% no \partial here
g4 |
a2 a |
g1 |
```
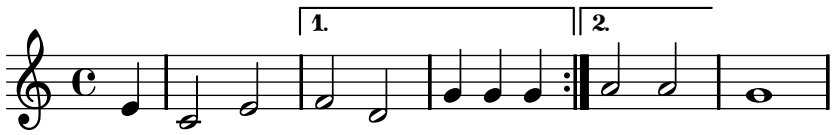


Lorsqu'une reprise débute par une mesure incomplète et a des fins alternatives, il est indispensable d'ajuster manuellement la propriété `Timing.measureLength` selon les préceptes suivants :

- au début de chacune des mesures incomplètes du bloc `\alternative`, ce qui est le cas en principe pour chaque fin d'alternative sauf – la plupart du temps – pour la dernière.
- au début de chaque alternative à l'exception de la première.

```
\partial 4
\repeat volta 2 { e4 | c2 e | }
\alternative {
  {
    f2 d |
```

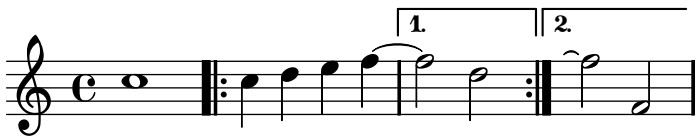
```
    \set Timing.measureLength = #(ly:make-moment 3 4)
    g4 g g  % optional bar check is allowed here
  }
  {
    \set Timing.measureLength = #(ly:make-moment 4 4)
    a2 a |
  }
}
g1 |
```



La propriété `measureLength` est abordée plus en détail au chapitre [Gestion du temps], page 97.

Des liaisons de tenue peuvent être ajoutées à toute alternative :

```
c1
\repeat volta 2 { c4 d e f ~ }
\alternative {
  { f2 d }
  { f2\repeatTie f, }
}
```
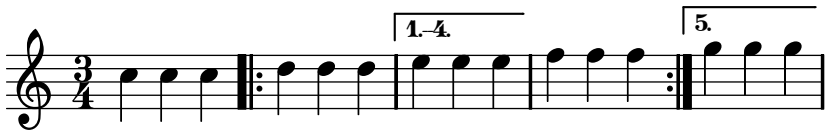


Morceaux choisis

Diminution de la taille du crochet d'alternative

Les crochets indiquant les fins alternatives s'étalent tout au long ce celle-ci. On peut les raccourcir en définissant la propriété `voltaSpannerDuration`. Dans l'exemple suivant, le crochet ne se prolonge que sur une mesure à 3/4.

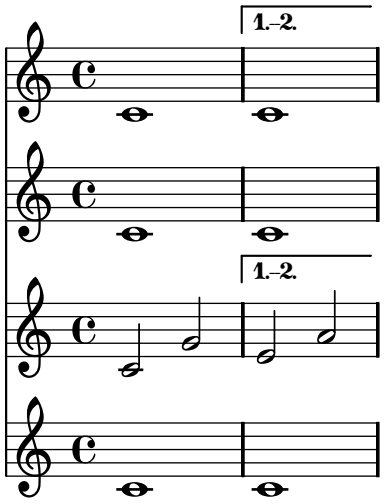
```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3 4)
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```



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.

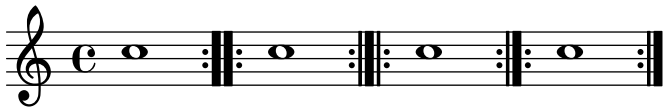
```
<<
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



Succession de reprise et style de barre par défaut

LilyPond dispose de trois styles de barre différents 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 }
}
```



Voir aussi

Glossaire musicologique : Section "répétition" dans *Glossaire*, Section "volta" dans *Glossaire*.

Manuel de notation : [Barres de mesure], page 81, ⟨undefined⟩ [Modification des greffons de contexte], page ⟨undefined⟩, [Gestion du temps], page 97.

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 à partir d’un bloc `\repeat` sur un bloc `\alternative` 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.

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.

Indications de reprise manuelles
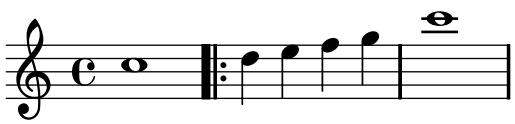
> **Note :** Les méthodes présentées dans les lignes qui suivent ne devraient servir à indiquer que des constructions de répétition inhabituelles. En règle générale, il vaut mieux recourrir à la fonction `\repeat` pour créer une reprise ou bien insérer la barre de mesure adéquate. Pour plus d’information, voir le chapitre [Barres de mesure], page 81.

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 `|:`

```
c1
\set Score.repeatCommands = #'(start-repeat)
d4 e f g
c1
```
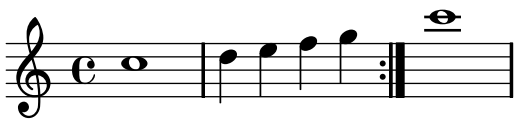


Traditionnellement, on n’imprime pas de signe de reprise en début de morceau.

`end-repeat`

Pour imprimer une barre de reprise `:|`

```
c1
d4 e f g
\set Score.repeatCommands = #'(end-repeat)
```
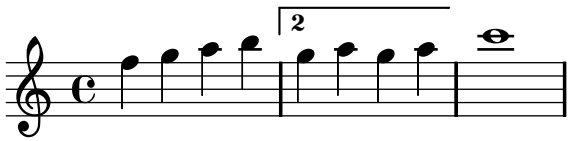
```
c1
```



`(volta` *nombre*`) ... (volta #f)`

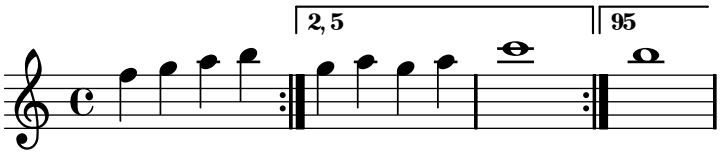
Pour obtenir un crochet indiquant le numéro de l'alternative. Pour que le crochet s'imprime effectivement, il faut spécifier explicitement l'endroit où il doît se terminer.

```
f4 g a b
\set Score.repeatCommands = #'((volta "2"))
g4 a g a
\set Score.repeatCommands = #'((volta #f))
c1
```
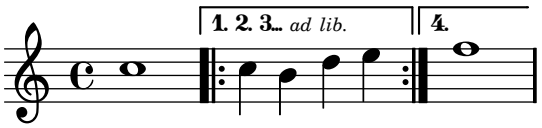


Plusieurs commande de reprise peuvent intervenir au même moment :

```
f4 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 198. 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'' {
  c1
  \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))
}
```

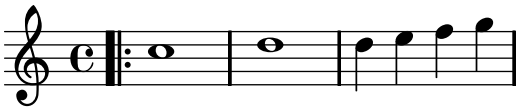


Morceaux choisis

Impression d'une barre de reprise en début de morceau

Bien qu'allant à l'encontre des usages en matière de gravure, vous imprimerez une barre de reprise (`|:`) en début de partition si vous surchargez la propriété adéquate :

```
\relative c'' {
  \once \override Score.BreakAlignment #'break-align-orders =
    #(make-vector 3 '(instrument-name
                      left-edge
                      ambitus
                      span-bar
                      breathing-sign
                      clef
                      key-signature
                      time-signature
                      staff-bar
                      custos
                      span-bar))
  \once \override Staff.TimeSignature #'space-alist =
    #'((first-note . (fixed-space . 2.0))
       (right-edge . (extra-space . 0.5))
       ;; free up some space between time signature
       ;; and repeat bar line
       (staff-bar . (extra-space . 1)))
  \bar "|:"
  c1
  d1
  d4 e f g
}
```



Voir aussi

Manuel de notation : [Barres de mesure], page 81, Section 1.8.2 [Mise en forme du texte], page 198.

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*.
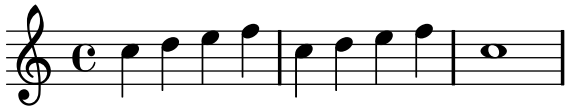
Répétitions explicites

Adjoindre à la commande `\repeat` l'opion `unfold` permet de s'affranchir de ressaisir de la musique répétitive. En voici la syntaxe :

```
\repeat unfold nombre_de_fois expression_musicale
```
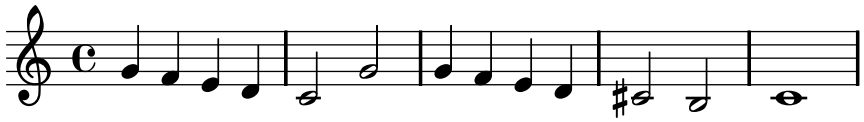
Le contenu de *expression_musicale* sera donc expansé autant de fois que stipulé par *nombre_de_fois*.

```
\repeat unfold 2 { c4 d e f }
c1
```
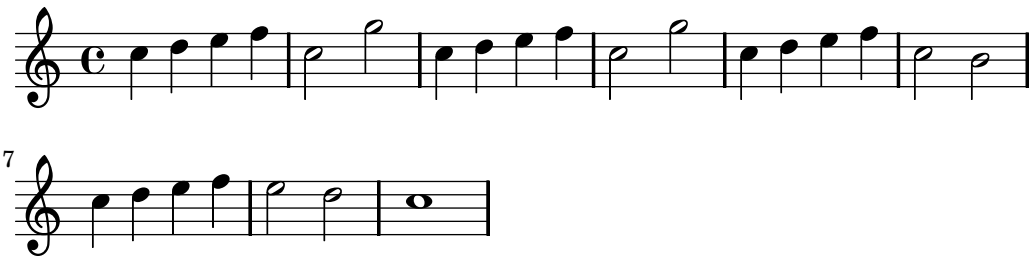


Une répétition expansée peut aussi avoir une fin alternative :

```
\repeat unfold 2 { g4 f e d }
\alternative {
  { c2 g' }
  { cis,2 b }
}
c1
```
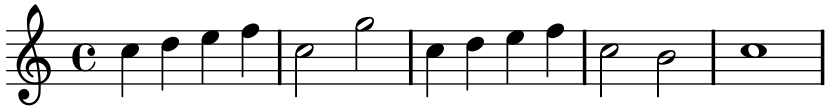


Si l'on donne trop peu d'alternatives en regard du nombre de fois où le passage doit être rejoué, la première alternative sera jouée plusieurs fois.

```
\repeat unfold 4 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
 }
c1
```



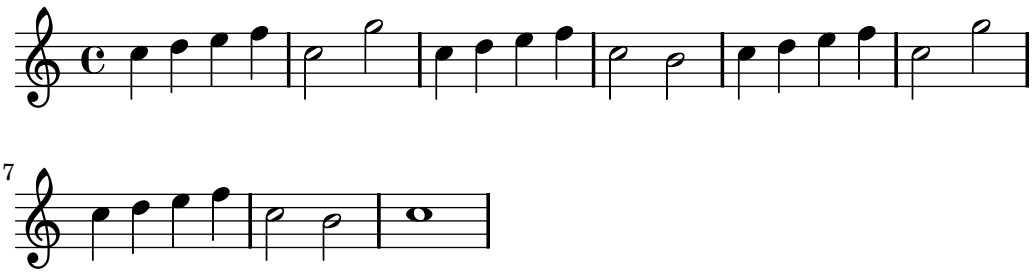
S'il y a par contre plus d'alternatives que de répétitions, les alternatives superflues seront tout simplement ignorées et ne seront pas imprimées.

```
\repeat unfold 2 { c4 d e f }
\alternative {
  { c2 g' }
  { c,2 b }
  { e2 d }
}
c1
```



Vous pouvez imbriquer plusieurs fonctions `unfold`, qu'elles comportent ou non des fins alternatives :

```
\repeat unfold 2 {
  \repeat unfold 2 { c4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
  }
}
c1
```



Note : L'insertion d'un `\relative` dans une section `\repeat` sans déclaration explicite du contexte `Voice` génèrera une portée supplémentaire – 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*.

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
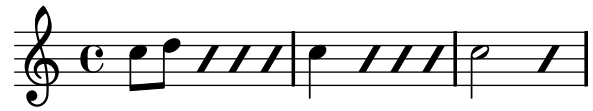
Le style de « reprise en pourcent » sert à répéter une séquence de notes. Elle sera imprimée une fois, puis remplacée par un symbole spécial.

En voici la syntaxe :

```
\repeat percent nombre expression_musicale
```
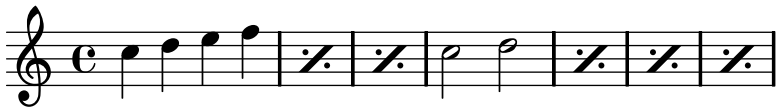
Les séquences inférieures à une mesure sont remplacées par une barre oblique.

```
\repeat percent 4 { c8 d }
\repeat percent 4 { c4 }
\repeat percent 2 { c2 }
```
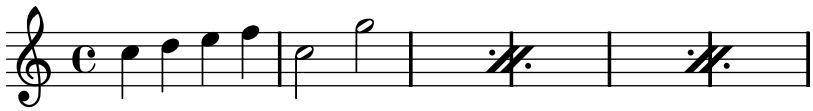


Les séquences d'une ou deux mesures sont remplacées par un symbole qui ressemble au symbole de pourcentage.

```
\repeat percent 3 { c4 d e f }
\repeat percent 4 { c2 d }
```



```
\repeat percent 3 { c4 d e f | c2 g' }
```
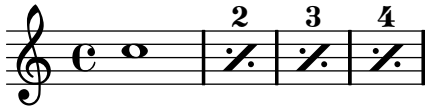


Morceaux choisis

Compteur de répétition en pourcent

Les répétitions de plus de 2 mesures sont surmontées d'un compteur, si l'on active la propriété `countPercentRepeats` comme le montre l'exemple suivant :

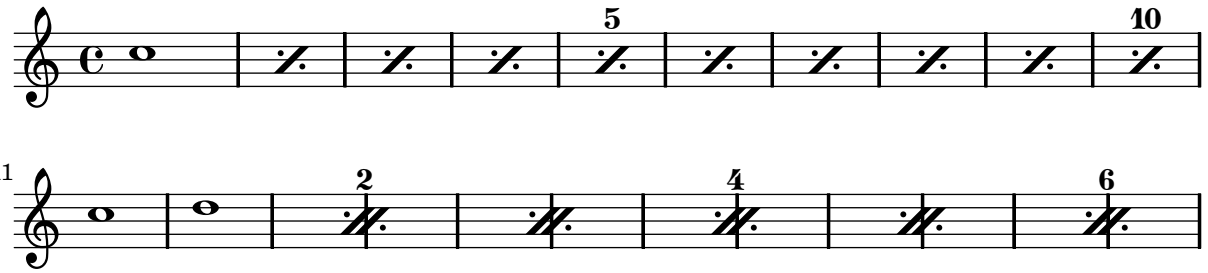
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



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 }
}
```
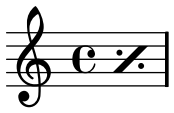


Répétition en pourcent isolée

Des symboles de pourcentage isolés peuvent aussi être obtenus, au moyen d'un silence multi-mesures dont on modifie l'aspect :

```
makePercent =
#(define-music-function (parser location 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 “PercentRepeat” 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*.

Problèmes connus et avertissements
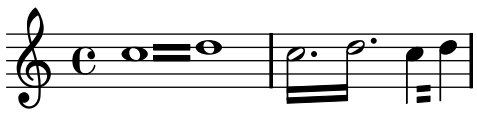
LilyPond ne dispose que de trois signes différents pour indiquer une répétition « en pourcent » : une simple barre oblique qui représente un seul temps quelle que soit la durée des notes répétées ; une barre oblique simple entourée de points qui représente une mesure entière ; une double barre oblique entourée de points et placée sur la barre de mesure pour indiquer une répétition de deux mesures.

Répétitions en trémolo

Il y a deux formes de trémolo : la répétition alernative 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 :

```
\repeat tremolo 8 { c16 d }
\repeat tremolo 6 { c16 d }
\repeat tremolo 2 { c16 d }
```



La syntaxe de `\repeat tremolo` requiert espressément 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.
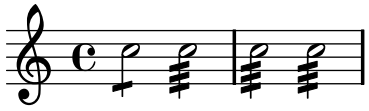
On peut indiquer de la même manière un trémolo sur une seule note, qu'il faudra alors laisser sans accolades :

```
\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 – telle que mémorisée dans `tremoloFlags` – sera utilisée.

```
c2:8 c:32
c: c:
```
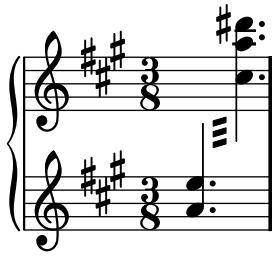


Morceaux choisis

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`.
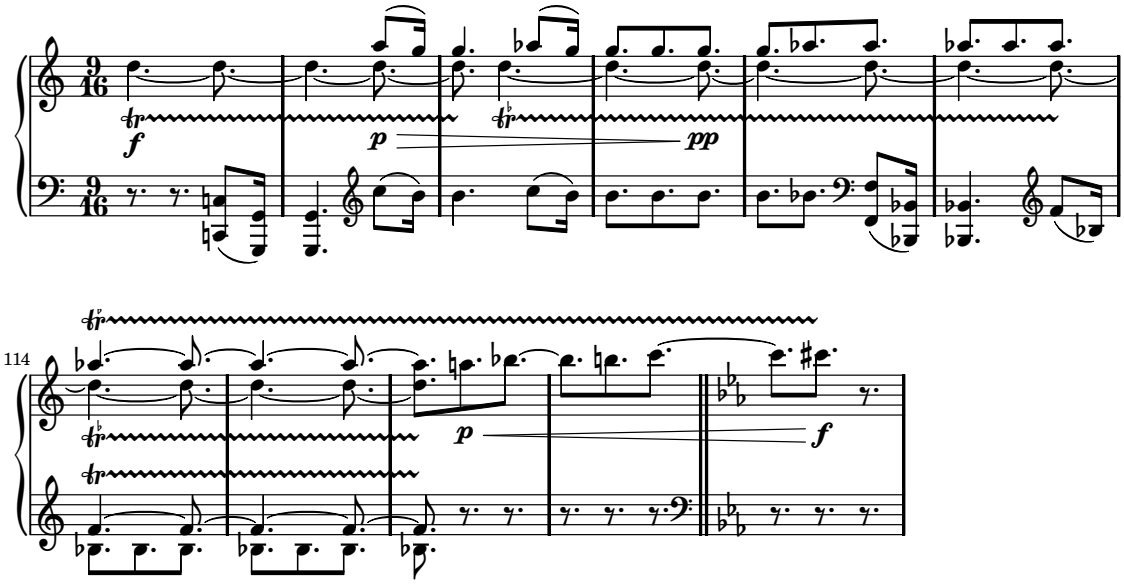
```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```



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.

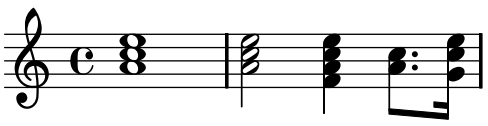
1.5.1 Monophonie

Nous allons voir ici comment gérer plusieurs notes simultanées dans un même voix.

Notes en accords

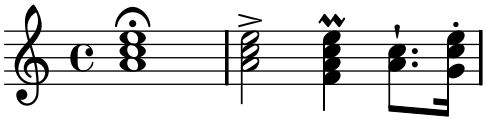
Un accord est formé en mettant une série de hauteurs entre `<` et `>`. Un accord peut être suivi d’une durée comme une simple note.

```
<a c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
```



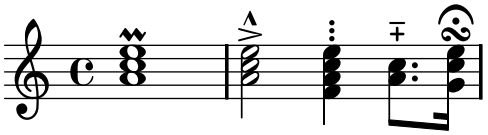
Un accord peut être suivi d’une indication d’articulation comme une simple note.

```
<a c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^| <g c e>16-.
```



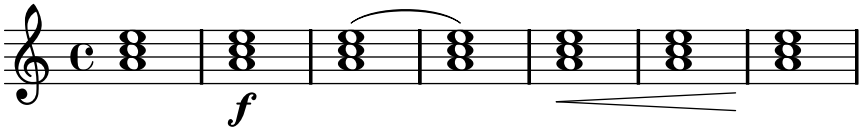
Certaines notes, au sein même d'un accord, peuvent être affectées d'une articulation ou d'un ornement :

```
<a c\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4 <a-+ c-->8. <g\fermata c e\turn>16
```



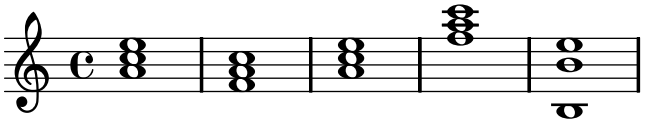
Certains éléments de notation, tels que nuances, soufflets et liaisons, ne peuvent se rattacher qu'à l'accord et non aux notes qui le composent, sous peine de ne les voir s'imprimer.

```
<a\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>) <a c e>\< <a c e> <a c e>\!
```



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.

```
<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 ⟨undefined⟩ [Notation des accords], page ⟨undefined⟩.

Voir aussi

Glossaire musical : Section "accord" dans *Glossaire*.

Manuel d'initiation : Section "Combinaison de notes en accords" dans *Manuel d'initiation*.

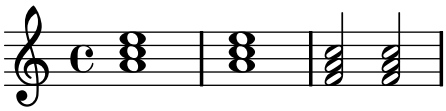
Manuel de notation : ⟨undefined⟩ [Notation des accords], page ⟨undefined⟩. [Articulations et ornements], page 99, [Octaves relatives], page 2.

Morceaux choisis : Section "Notes simultanées" dans *Morceaux choisis*.

Répétition d'accords

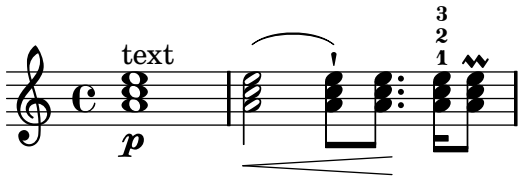
Dans le but de vous épargner de la saisie, LilyPond dispose d'un raccourci – symbolisé par la lettre `q` – qui a pour effet de répéter le dernier accord saisi :

```
<a c e>1 q <f a c>2 q
```
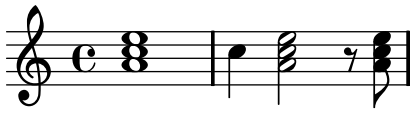


À l'instar de n'importe quel accord, le symbole de répétition peut être affublé d'une durée, de signes d'articulation, *markups*, liaisons, ligatures… En fait, c'est la structure du dernier accord qui est dupliquée.

```
<a c e>1\p^"text" q2\<( q8)[-| q8.]\! q16-1-2-3 q8\prall
```



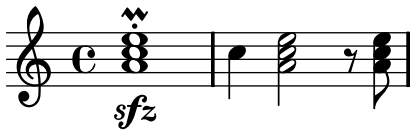
Dans la mesure où le symbole de répétition d'accord enregistre la structure du dernier accord construit, il est tout à fait possible de l'utiliser même après une succession de notes individuelles et de silences :

```
<a c e>1 c4 q2 r8 q8
```



ILe symbole de répétition d'accord ne prend en charge que les hauteurs, en aucun cas les nuances, articulation ou ornementations, qu'elles aient été attachées aux notes le composant ou à l'ensemble.

```
<a-. c\prall e>1\sfz c4 q2 r8 q8
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Notation des accords], page ⟨undefined⟩, [Articulations et ornements], page 99.
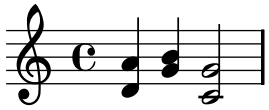
Fichiers d'initialisation : '`ly/chord-repetition-init.ly`'.

Expressions simultanées
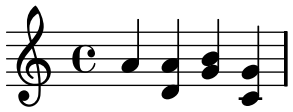
Lorsqu'une ou plusieurs expressions musicales sont encadrées par des doubles chevrons, elles sont considérées comme étant simultanées. Si la première expression débute par une note unique ou si l'intégralité de l'expression simultanée est explicitement rattachée à une voix en particulier, elle sera placée 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
  << { a4 b g2 } { d4 g c,2 } >>
}
```



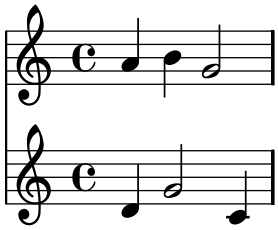
```
% single first note
a << { a4 b g } { d4 g c, } >>
```



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.

Dans l'exemple suivant, l'expression simultanée génère implicitement plusieurs portées :

```
% no single first note
<< { a4 b g2 } { d4 g2 c,4 } >>
```

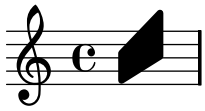


En pareil cas, des rythmes différents ne sont source d'aucun problème.

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

```
\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 extrèmement 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

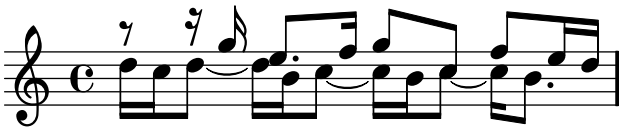
Nous allons nous intéresser, dans les paragraphes qui suivent, à la gestion de notes simultanées réparties sur plusieurs voix ou plusieurs portées.

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 est la suivante :

```
\new Staff <<
  \new Voice = "first"
    { \voiceOne r8 r16 g e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    { \voiceTwo d16 c d8~ d16 b c8~ c16 b c8~ c16 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.
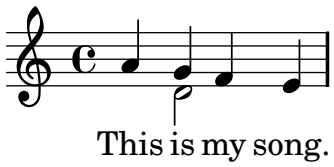
Polyphonie temporaire

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.

```
<<
  \new Voice = "melody" {
    a4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
  This is my song.
  }
>>
```

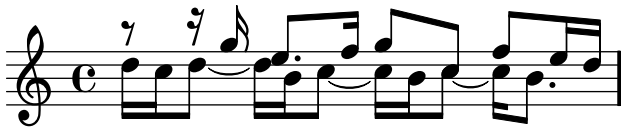


Vous remarquerez que les commandes `\voiceOne` et `\voiceTwo` permettent d'obtenir des réglages différents pour chacune des voix.

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 séparateurs : **tous** les membres de cette contruction 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 :
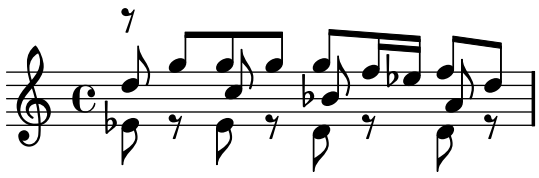
```
<<
  { r8 r16 g e8. f16 g8[ c,] f e16 d }
  \\
  { d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```



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 même 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 à 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.

```
<<
  { r8 g g  g g f16 ees f8 d }
  \\
  { ees,8 r ees r d r d r }
  \\
  { d'8 s c s bes s a s }
>>
```
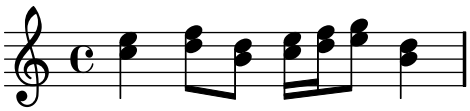


En dehors des cas les plus simples, nous vous invitons à toujours créer les contextes de voix de manière explicite. Voir à ce sujet Section "Contextes et graveurs" dans *Manuel d'initiation* et Section "Instanciation explicite des voix" dans *Manuel d'initiation*.

Identité rythmique

Lorsque l'on doît 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 <<
  { e4 f8 d e16 f g8 d4 }
  { c4 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 "Les voix contiennent la musique" dans *Manuel d'initiation*, Section "Instanciation explicite des voix" dans *Manuel d'initiation*.

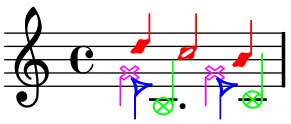
Manuel de notation : ⟨undefined⟩ [Portées de percussion], page ⟨undefined⟩, [Silences invisibles], page 49, [Hampes], page 184.

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 facliement :

```
<<
  { \voiceOneStyle d4 c2 b4 }
  \\
  { \voiceTwoStyle e,2 e }
  \\
  { \voiceThreeStyle b2. c4 }
  \\
  { \voiceFourStyle g'2 g }
>>
```



La commande `\voiceNeutralStyle` permet de revenir à la présentation normale.

Commandes prédéfinies

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

Voir aussi
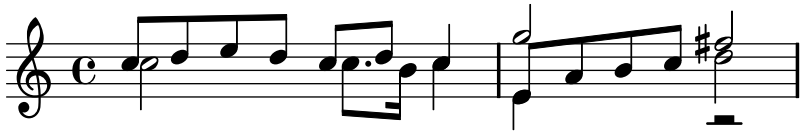
Manuel d'initiation : Section "J'entends des Voix" dans *Manuel d'initiation*, Section "Autres sources de documentation" dans *Manuel d'initiation*.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Résolution des collisions
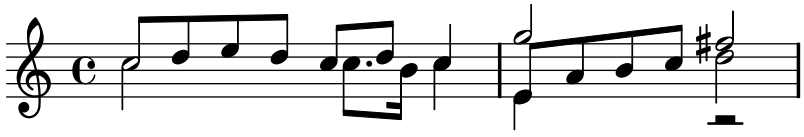
Les notes de hauteur identique appartenant à des voix différentes, même si leur hampe sont opposées, verront leur tête automatiquement fusionnées. Les notes dont la tête diffère ou bien qui ont la hampe dans la même direction ne seront pas automatiquement fusionnées. Les silences, lorsqu'ils sont dans 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 mesur, ainsi qu'au premier temps de la deuxième mesure.

```
<<
  {
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 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 sont bien fusionnées :
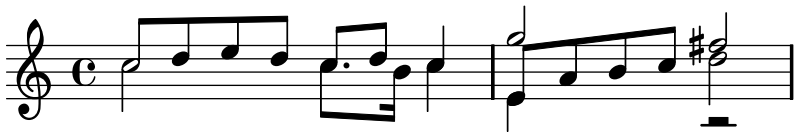
```
<<
  {
    \mergeDifferentlyHeadedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



De même, vous pouvez fusionner les têtes de notes pointées et non-pointées comme au troisième temps de la première mesure :
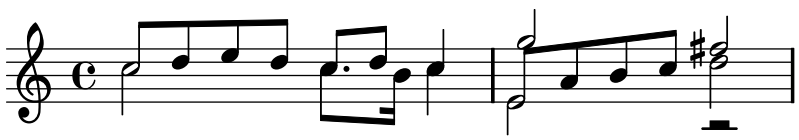
```
<<
  {
```

```
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 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.

```
<<
  {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\ {
    c2 c8. b16 c4
    e,2 r
  } \\ {
    \oneVoice
    s1
    e8 a b c d2
  }
>>
```



Les commandes `\shiftOn`, `\shiftOnn`, et `\shiftOnnn` déterminent le niveau de décalage des accords de la voix en cours. 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`. `\shiftOnn` et `\shiftOnnn` sont des niveaux supplémentaires de décalage.

Les têtes de notes ne seront fusionnées 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`.
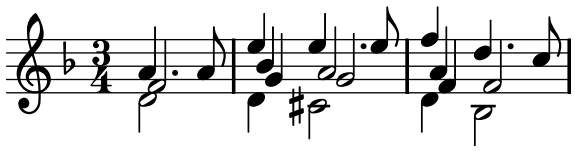
Morceaux choisis

Ajout de voix pour éviter les collisions

Dans certains cas de musique polyphonie 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 des d'autres voix.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

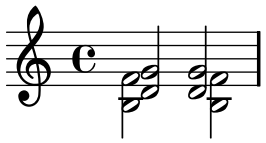
\relative c'' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \\
    {
      \voiceThree
      f,2
      bes4 a2
      a4 s2
    }
    \\
    {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
    \\
    \bar "||"{
      \voiceTwo
      d2
      d4 cis2
      d4 bes2
    }
  >>
}
```



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 "Notes simultanées" dans *Manuel d'initiation*, Section "Les voix contiennent la musique" dans *Manuel d'initiation*, Section "Exemple concret" 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

Il n'y a aucune prise en charge des agrégats dans lesquels une même note apparaît avec différentes altérations. Il est conseillé d'avoir recours aux enharmoniques, ou d'utiliser la notation spécifique de cluster – voir [Clusters], page 136.

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 :
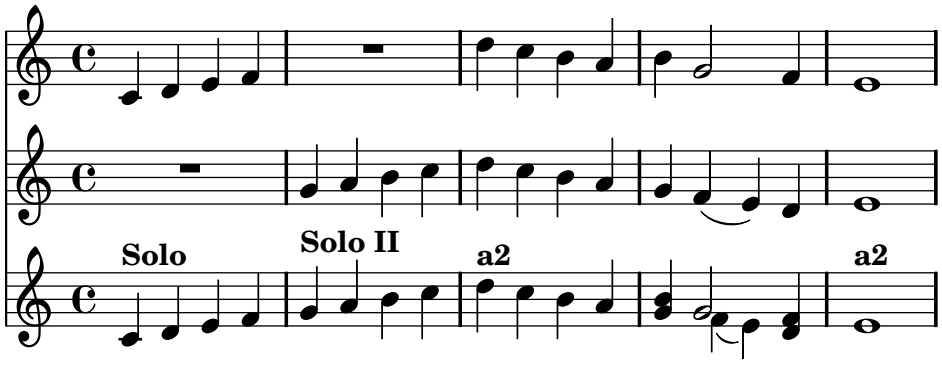
`\partcombine` *`expression_musicale_1 expression_musicale_2`*

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 identificateurs sont les mêmes pour la combinaison que pour les parties séparées.

```
instrumentOne = \relative c' {
  c4 d e f
  R1
  d'4 c b a
  b4 g2 f4
  e1
}

instrumentTwo = \relative g' {
  R1
  g4 a b c
  d c b a
  g f( e) d
  e1
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partcombine \instrumentOne \instrumentTwo
>>
```



Les notes de la troisième mesure n'apparaîssent qu'une seule fois, alors qu'elles ont été spécifiés 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 estampillées d'un « a2 » par défaut.

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`*.

Morceaux choisis

Combinaison de deux parties sur une même portée
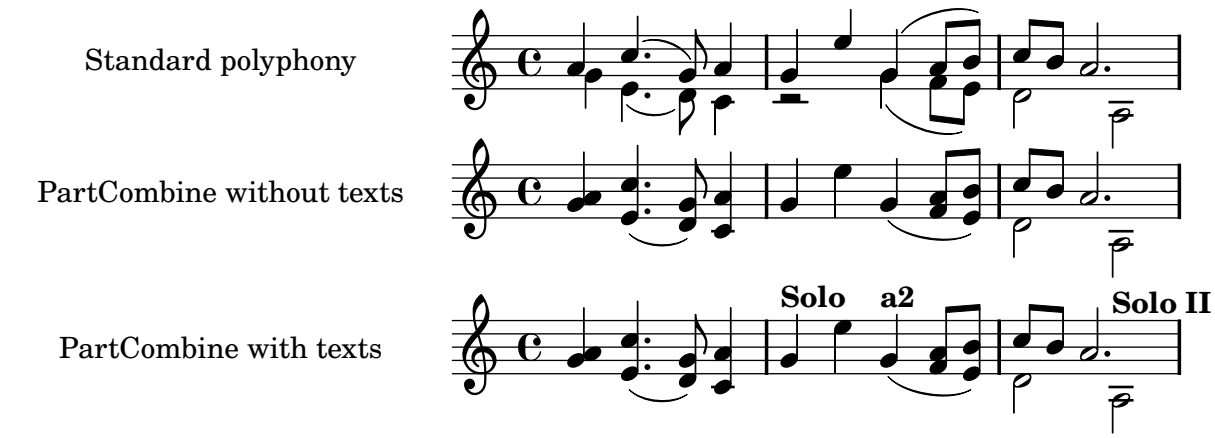
L'outil de combinaison de parties (la commande `\partcombine`) permet d'avoir deux parties différentes sur une même portée. LilyPond ajoute automatiquement des indications textuelles, telles que « solo » ou « a2 ». Si votre intention n'est que de fusionner les parties, sans ajouter de texte, assignez 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 un même portée : en polyphonie normale, avec `\partcombine` sans indication supplémentaire, et avec `\partcombine` commentée.

```
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
}
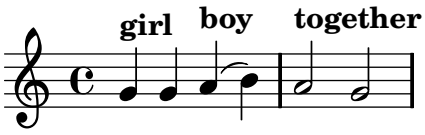
\score {
  <<
    <<
    \new Staff {
      \set Staff.instrumentName = #"Standard polyphony"
      << \musicUp \\ \musicDown >>
    }
    \new Staff \with { printPartCombineTexts = ##f } {
      \set Staff.instrumentName = #"PartCombine without texts"
      \partcombine \musicUp \musicDown
    }
    \new Staff {
      \set Staff.instrumentName = #"PartCombine with texts"
      \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 167.

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

`\partcombine` ne prend en charge que deux voix.

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` ne peut s'inscrire dans un bloc `\times`.

`\partcombine` ne peut s'inscrire dans un bloc `\relative`.

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 Section "Voice" dans *Référence des propriétés internes*, les événements qu'ils contiendraient seront ignorés. 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` 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.

Saisie de musique en parallèle
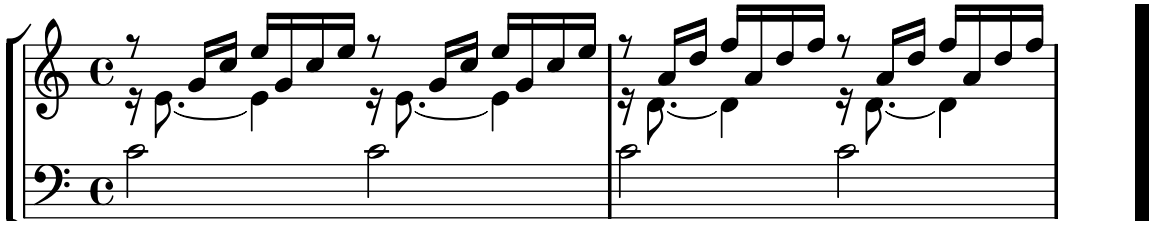
On peut écrire plusieurs voix de façon entremêlée. La fonction `\parallelMusic` prend en charge une liste des variables à créer, ainsi qu'une expression musicale. Le contenu des différentes mesures de l'expression musicale deviennent les valeurs des variables respectives que vous pourrez ensuite utiliser pour imprimer la partition.

> **Note :** Les contrôles de barre de mesure `|` sont obligatoires et le 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.~   e'4           r16 e'8.~   e'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.~   d'4           r16 d'8.~   d'4           |
  c'2                       c'2                       |

}
\new StaffGroup <<
  \new Staff << \voiceA \\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```

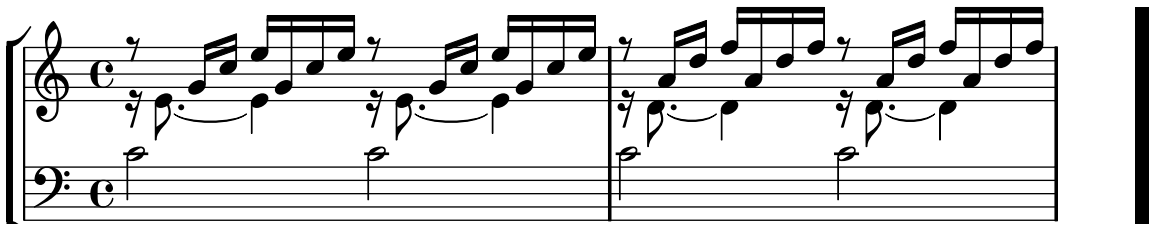


Vous pouvez travailler en mode relatif. Notez cependant que la commande `\relative` n'apparaît pas au sein du bloc `\parallelMusic`. Le calcul des hauteurs relatives s'effectue voix par voix, et non au fil des lignes saisies ; en d'autres termes, les notes de la `voiceA` ignorent tout de celles de la `voiceB`.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ e4       r16 e8.~  e4        |
  c2                c                   |
```

```
  % Bar 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  d4       r16 d8.~  d4       |
  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
}
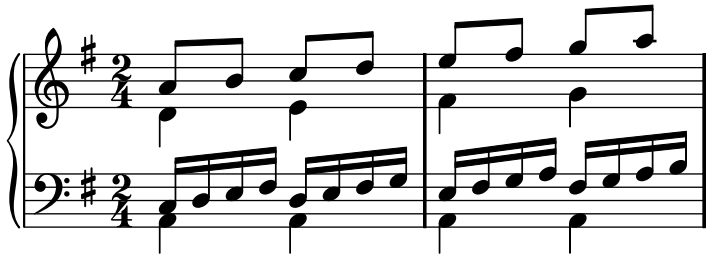
\parallelMusic #'(voiceA voiceB voiceC voiceD) {
  % Bar 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g |
  a4          a         |

  % 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
        >>
      }
  >>
}
```
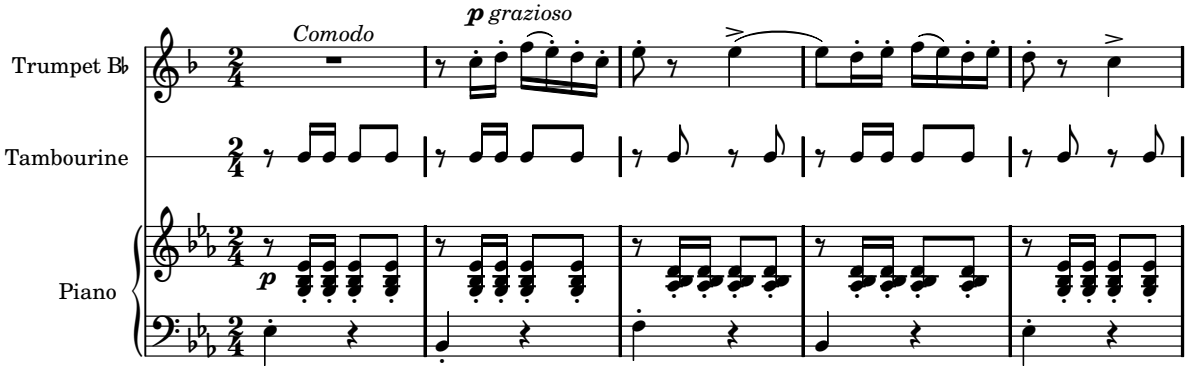


Voir aussi

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
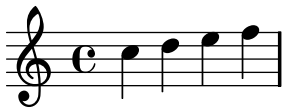
Nous allons voir ici comment créer des portées et comment les regrouper.

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 ⟨undefined⟩ [Création d'un contexte], page ⟨undefined⟩.

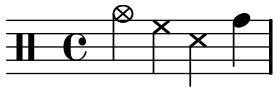
Le contexte de portée standard s'appelle `Staff` :

```
\new Staff { c4 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 ⟨undefined⟩ [Portées de percussion], page ⟨undefined⟩.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



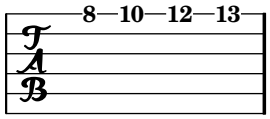
Un `RhythmicStaff` est composé d'une portée à ligne unique chargée de présenter les valeurs rythmiques saisies. Seules sont imprimées les durées. Pour de plus amples détails, consultez [Gravure de lignes rythmiques], page 66.

```
\new RhythmicStaff { c4 d e f }
```



Un `TabStaff` crée une portée de tablature correpondant aux six cordes d'une guitare standard. Pour de plus amples détails, consultez ⟨undefined⟩ [Tablatures par défaut], page ⟨undefined⟩.
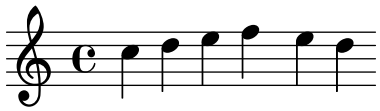
```
\new TabStaff { c4 d e f }
```



Lilypond dispose aussi de deux contextes dédiés à la musique ancienne : `MensuralStaff` et `VaticanaStaff`. Ils sont abordés plus en détails au chapitre ⟨undefined⟩ [Contextes prédéfinis], page ⟨undefined⟩.

Le contexte `GregorianTranscriptionStaff` permet d'obtenir des éditions modernes du grégorien. Bien entendu, il est dépourvu de barres de mesure.

```
\new GregorianTranscriptionStaff { c4 d e f e d }
```



Vous pourrez toujours définir d'autres contextes de portée selon vos besoins, en suivant les indications fournies au chapitre ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

Voir aussi

Glossaire musicologique : Section “staff” dans *Glossaire*, Section “portées” dans *Glossaire*.

Manuel de notation : ⟨undefined⟩ [Création d’un contexte], page ⟨undefined⟩, ⟨undefined⟩ [Portées de percussion], page ⟨undefined⟩, [Gravure de lignes rythmiques], page 66, ⟨undefined⟩ [Tablatures par défaut], page ⟨undefined⟩, ⟨undefined⟩ [Contextes prédéfinis], page ⟨undefined⟩, [Symbole de la portée], page 157, ⟨undefined⟩ [Contextes du chant grégorien], page ⟨undefined⟩, ⟨undefined⟩ [Les contextes de la musique mensurale], page ⟨undefined⟩, ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

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

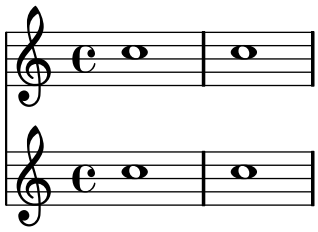
Référence des propriétés internes : Section “Staff” dans *Référence des propriétés internes*, Section “DrumStaff” dans *Référence des propriétés internes*, Section “GregorianTranscriptionStaff” dans *Référence des propriétés internes*, Section “RhythmicStaff” dans *Référence des propriétés internes*, 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 regoupement, avec son signe particulier en début de portée et ses règles de gestion des barres de mesure.

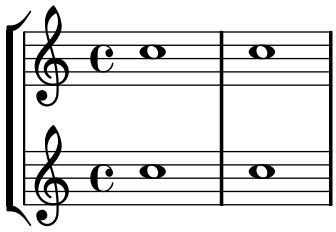
Lorsqu’aucun contexte particulier n’est spécifié, les propriétés suivantes s’appliqueront par défaut : les portées du groupe ne sont pas reliées, hormis par une simple ligne verticale en début de ligne, et les barres de mesure sont indépendantes.

```
<<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



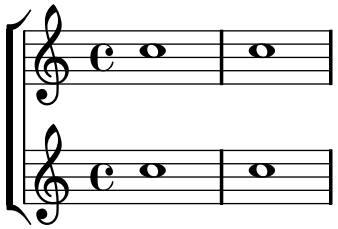
Dans un `StaffGroup`, le groupe de portées est signifié par un crochet, et les barres de mesure sont d’un seul tenant.

```
\new StaffGroup <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



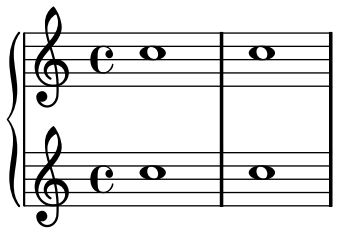
Dans un `ChoirStaff`, le groupe de portées est signifié par un crochet sur la gauche, et les barres de mesure sont individuelles.

```
\new ChoirStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



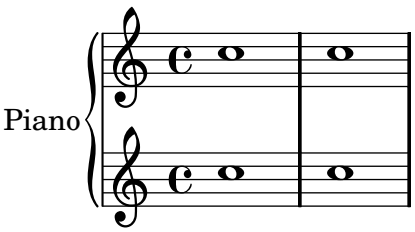
Dans un `GrandStaff`, le groupe de portées est signifié par une accolade sur la gauche, et les barres de mesure sont d'un seul tenant.

```
\new GrandStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```



Le contexte `PianoStaff` est indentique au `GrandStaff`, sauf qu'il gère automatiquement l'affichage du nom d'instrument – voir [Noms d'instrument], page 167 pour plus de détails.

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff { c1 c }
  \new Staff { 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.

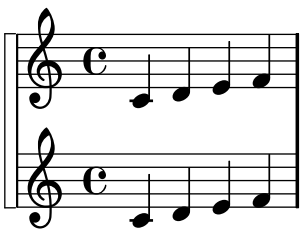
Vous pouvez aussi créer vos propres contextes de regroupement, comme l'explique ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

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 `ChoirStaffGroup`.

```
\score {
  \new StaffGroup { <<
  \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```



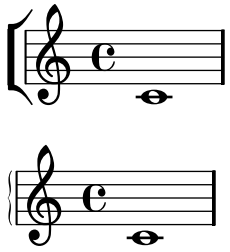
Indicateur de regroupement et portée unique

Lorsque, dans des regoupements de type `ChoirStaff` ou `StaffGroup`, une seule portée est active, aucune indication n'est donnée en début de ligne. Surcharger la propriété adéquate 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.

```
\markup \left-column {
  \score {
    \new StaffGroup <<
      % Must be lower than the actual number of staff lines
      \override StaffGroup.SystemStartBracket #'collapse-height = #1
      \override Score.SystemStartBar #'collapse-height = #1
      \new Staff {
        c'1
      }
    >>
    \layout { }
  }
  \null
  \score {
    \new PianoStaff <<
      \override PianoStaff.SystemStartBrace #'collapse-height = #1
      \override Score.SystemStartBar #'collapse-height = #1
      \new Staff {
        c'1
      }
    >>
    \layout { }
```

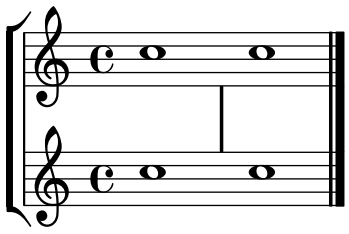
```
  }
}
```



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 rendre « transparentes » les portions de barre qui recouvrent les portées.

```
global = {
  \override Staff.BarLine #'transparent = ##t
  s1 s
  % the final bar line is not interrupted
  \revert Staff.BarLine #'transparent
  \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 : [Noms d'instrument], page 167, ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

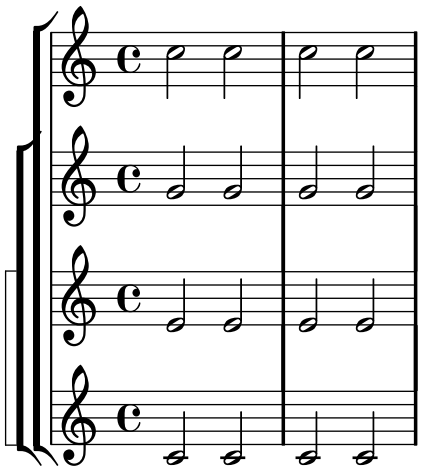
Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

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 { c2 c | c2 c }
  \new StaffGroup <<
    \new Staff { g2 g | g2 g }
    \new StaffGroup \with {
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff { e2 e | e2 e }
      \new Staff { c2 c | c2 c }
    >>
  >>
>>
```



Vous pouvez aussi créer vos propres contextes d'imbrication, comme l'explique ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

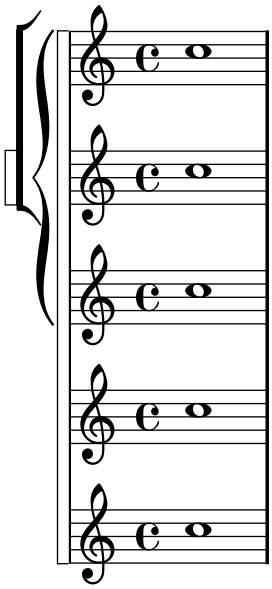
Morceaux choisis

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'' <<
  \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 : [Regroupement de portées], page 151, [Noms d'instrument], page 167, ⟨undefined⟩ [Définition de nouveaux contextes], page ⟨undefined⟩.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

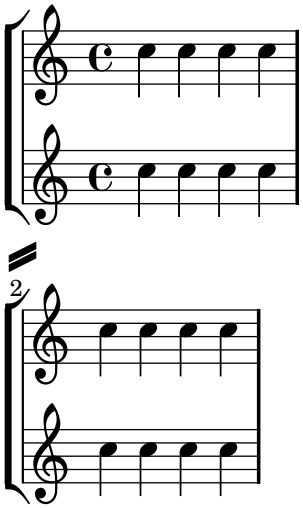
Référence des propriétés internes : Section "StaffGroup" dans *Référence des propriétés internes*, Section "ChoirStaff" dans *Référence des propriétés internes*, Section "SystemStartBar" dans *Référence des propriétés internes*, Section "SystemStartBrace" dans *Référence des propriétés internes*, Section "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 povez, 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'' {
          c4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative c'' {
          c4 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 : ⟨undefined⟩ [Mise en forme de la page], page ⟨undefined⟩.

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*.
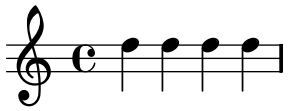
Symbole de la portée

Les 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. Néanmoins, 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 12.
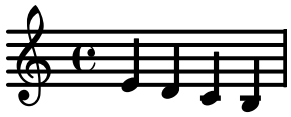
```
\new Staff \with {
  \override StaffSymbol #'line-count = #3
```

```
}
{ d4 d d d }
```
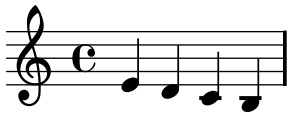


Lorsque vous modifierez l'épaisseur des lignes, gardez à l'esprit que les lignes supplémentaires et les hampes seront aussi modifiées. En effet, elles dépendent directement de l'épaisseur des lignes de la portée.

```
\new Staff \with {
  \override StaffSymbol #'thickness = #3
}
{ e4 d c b }
```



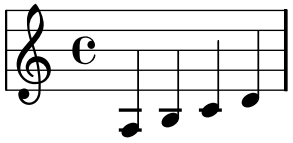
L'épaisseur des lignes supplémentaires (*ledger lines*) peut être déterminée indépendamment des lignes de la portée. Dans l'exemple suivant, les deux nombres constituent des coefficients multiplicateurs de l'épaisseur de la ligne. Ils sont fournis en argument pour déterminer l'épaisseur des lignes supplémentaires.

```
\new Staff \with {
  \override StaffSymbol #'ledger-line-thickness = #'(1 . 0.2)
}
{ e4 d c b }
```



Modifier l'équidistance des lignes de la portée affectera aussi les lignes supplémentaires.

```
\new Staff \with {
  \override StaffSymbol #'staff-space = #1.5
}
{ a4 b c d }
```



Vous trouverez de plus amples détails sur les propriétés du `StaffSymbol` au chapitre Section "staff-symbol-interface" dans *Référence des propriétés internes*.

Les propriétés d'une portée peuvent être modifiées en cours de partition à l'aide des commandes `\stopStaff` et `\startStaff` :

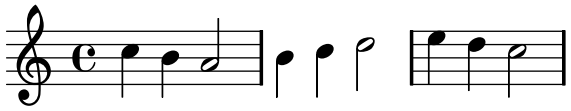
```
c2 c
\stopStaff
\override Staff.StaffSymbol #'line-count = #2
\startStaff
```

```
b2 b
\stopStaff
\revert Staff.StaffSymbol #'line-count
\startStaff
a2 a
```



En règle générale, `\startStaff` et `\stopStaff` permettent d'entamer puis clôturer une portée n'importe où dans une partition.

```
c4 b a2
\stopStaff
b4 c d2
\startStaff
e4 d c2
```



Commandes prédéfinies
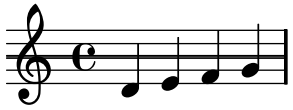
`\startStaff`, `\stopStaff`.

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 clé de sol. Il suffit pour cela d'ajouter une ligne qui sera accolée à celle qui doît 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 suplémentaire” dans *Glossaire*, Section “portée” dans *Glossaire*.

Manuel de notation : [Clefs], page 12.
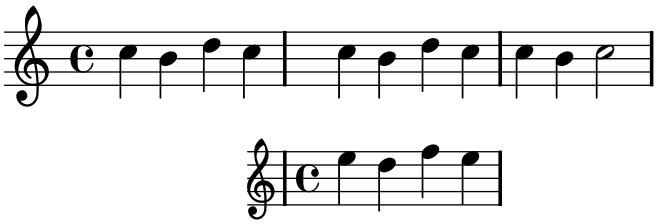
Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

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'' {
  c4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```
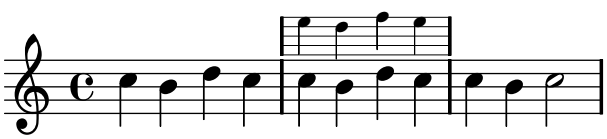


L'exemple ci-dessus n'est probablement pas ce qui vous conviendra le plus. Afin que cette ossia se place au-dessus de la portée à laquelle elle se réfère, étant par ailleurs dépourvue de métrique et de clef, et d'une taille légèrement inférieure, vous devrez avoir recours à quelques retouches. Le manuel 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'' {
  c4 b d c
  <<
    { c4 b d c }

    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = #"main"
      fontSize = #-3
      \override StaffSymbol #'staff-space = #(magstep -3)
      \override StaffSymbol #'thickness = #(magstep -3)
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```
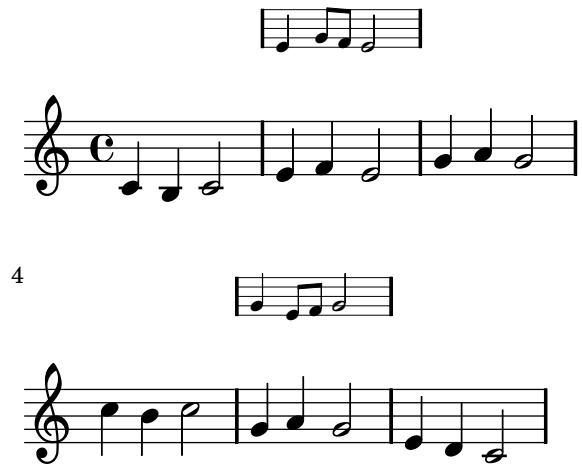


Dans le cas où de nombreux et courts fragments d'ossia affecteraient une même portée, il est judicieux de créer un contexte `Staff` vide auquel sera attribué un *identificateur*. Il suffira

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"
    \override Clef #'transparent = ##t
    fontSize = #-3
    \override StaffSymbol #'staff-space = #(magstep -3)
    \override StaffSymbol #'thickness = #(magstep -3)
  }
  { \stopStaff s1*6 }
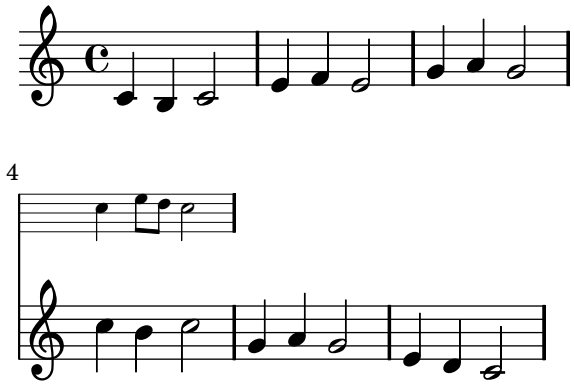
  \new Staff \relative c' {
    c4 b c2
    <<
      { e4 f e2 }
      \context Staff = ossia {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = ossia {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```



Vous pourriez aussi recourir à la commande `\Staff \RemoveEmptyStaves` 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 `\Staff \RemoveEmptyStaves`, reportez-vous au chapitre [Masquage de portées], page 163.

```
<<
  \new Staff = ossia \with {
    \remove "Time_signature_engraver"
    \override Clef #'transparent = ##t
    fontSize = #-3
    \override StaffSymbol #'staff-space = #(magstep -3)
    \override StaffSymbol #'thickness = #(magstep -3)
  } \relative c'' {
    R1*3
    c4 e8 d c2
  }
  \new Staff \relative c' {
    c4 b c2
    e4 f e2
    g4 a g2 \break
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    \override VerticalAxisGroup #'remove-first = ##t
  }
}
```



Morceaux choisis
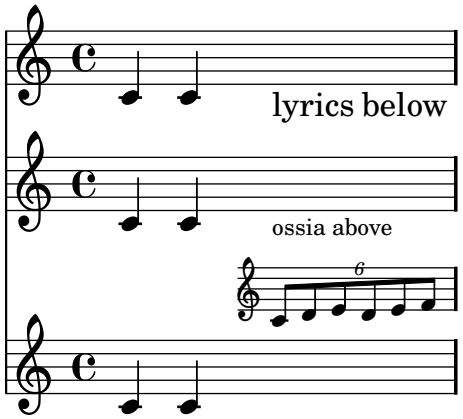
Positionnement d'une ossia et des paroles

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"
    } {
      \times 4/6 {
        \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 “Taille des objets” dans *Manuel d’initiation*, Section “Longueur et épaisseur des objets” dans *Manuel d’initiation*.

Manuel de notation : [Masquage de portées], page 163.

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 c''' { a8 f e16 d c b a2 }
```

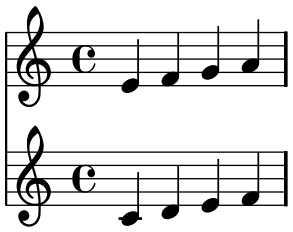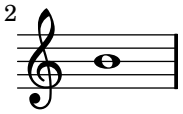


L'instruction `\Staff \RemoveEmptyStaves` placée dans un bloc `\layout` aura pour effet de masquer toute portée qui ne contient rien. Dans les partitions d'orchestre, les portées qui n'ont que des silences sont habituellement masquées afin de gagner de la place. Ce style d'édition s'appelle en anglais « French Score ». Cette fonctionnalité masque et supprime toutes les portées vides d'une partition, hormis celles du premier système.
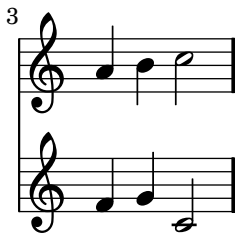
> **Note :** Une portée est considérée comme vide dès lors qu'elle ne contient que des silences multimesures, des silences visibles ou invisibles (ou d'espacement – les *skip*) ou bien une combinaison de ces éléments.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}

\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
>>
```







`\Staff \RemoveEmptyStaves` permet aussi de gérer des fragments d'ossia attachés à une portée. Pour plus de détails, voir [Portées d'ossia], page 160.

La commande `\VaticanaStaff \RemoveEmptyStaves` permet de masquer des portées vides dans les contextes de musique ancienne. Pour des contextes `RhythmicStaff`, il faudra utiliser `\RhythmicStaff \RemoveEmptyStaves`.

Commandes prédéfinies

`\Staff \RemoveEmptyStaves`, `\VaticanaStaff \RemoveEmptyStaves`, `\RhythmicStaff \RemoveEmptyStaves`.

Morceaux choisis

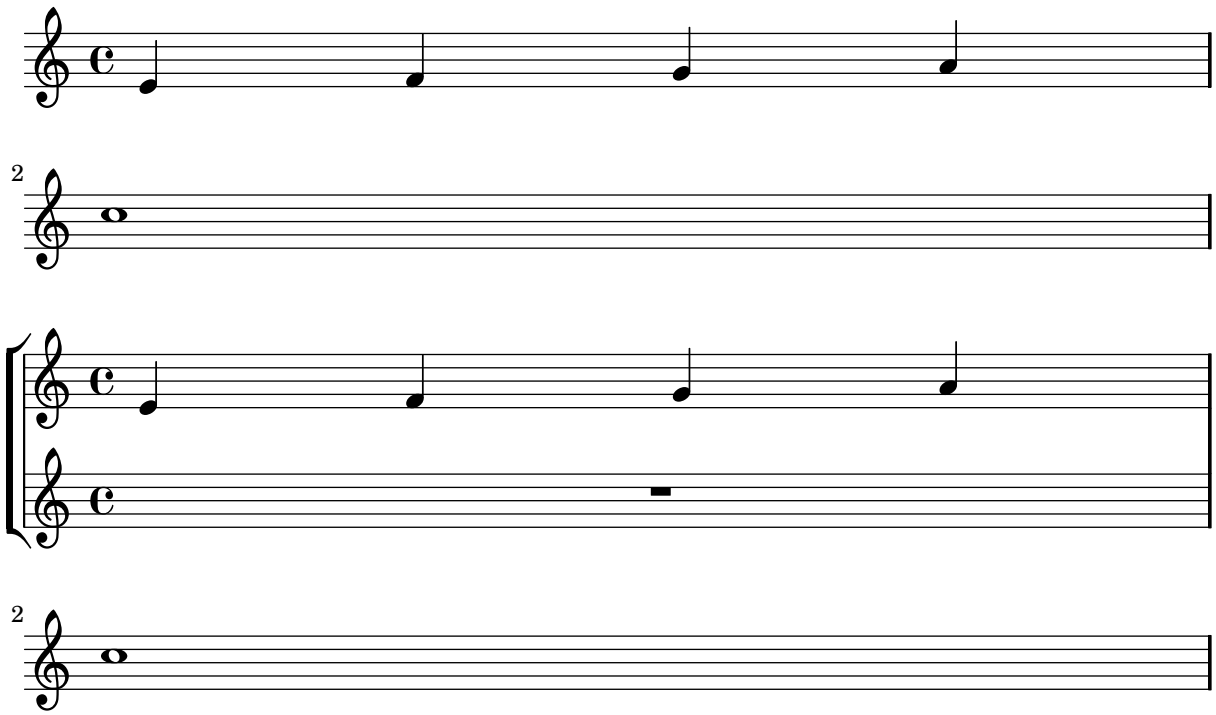
Masquage de la première ligne si elle est vide

Par défaut, le premier système comportera absolument toutes les portées. Si vous voulez masquer les portées vides y compris pour le premier système, vous devrez activer la propriété `remove-first` du `VerticalAxisGroup`. Mentionnée dans un bloc `\layout`, cette commande agira de manière globale. Pour qu'elle ne soit effective que pour une portée individuelle, vous devrez également spécifier le contexte (`Staff` pour qu'il ne concerne que la portée en cours) en préfixe de la propriété.

La première ligne inférieure du deuxième `StaffGroup` est bien présente, pour la simple raison que le réglage en question ne s'applique qu'à la portée dans laquelle il a été inscrit.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
    % \override VerticalAxisGroup #'remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup #'remove-first = ##t
    R1 \break
    R
  }
>>
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
```

```
    R1 \break
    R
  }
>>
```



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 : [Symbole de la portée], page 157, [Portées d’ossia], page 160, [Dictée à trous], page 181, [Silences invisibles], page 49, ⟨undefined⟩ [Visibilité des objets], page ⟨undefined⟩.

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

Référence des propriétés internes : Section “ChordNames” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*, Section “Lyrics” dans *Référence des propriétés internes*, Section “Staff” dans *Référence des propriétés internes*, Section “VerticalAxisGroup” dans *Référence des propriétés internes*, Section “Staff_symbol_engraver” 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 formattage. En pareil cas il vaut mieux, au lieu de supprimer le graveur, recourrir aux dérogations suivantes :

```
\override StaffSymbol #'stencil = ##f
\override NoteHead #'no-ledgers = ##t
```

En ce qui concerne les désagréments et messages liés à l’utilisation de l’instruction `\Staff \RemoveEmptyStaves`, consultez ⟨undefined⟩ [Modification des réglages par défaut d’un contexte], page ⟨undefined⟩.
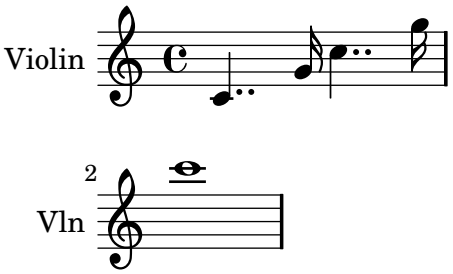
1.6.3 Écriture de parties séparées

Nous allons voir au fil des lignes qui suivent, comment insérer des indications de tempo ou des noms d'instrument dans une partition. Nous aborderons aussi la citation d'autres voix, et comment la mettre en forme.

Noms d'instrument
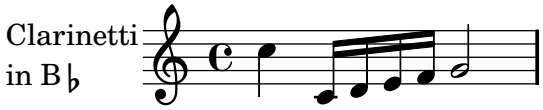
Dans un conducteur, les noms d'instrument sont portés en regard de chacune des portées, qu'il s'agisse d'un contexte `Staff` ou d'un `PianoStaff`. La première ligne affichera la valeur de `instrumentName`, et les suivantes celle de `shortInstrumentName`.

```
\set Staff.instrumentName = #"Violin "
\set Staff.shortInstrumentName = #"Vln "
c4.. g'16 c4.. g'16
\break
c1
```
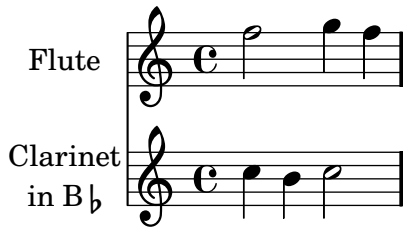


Le recours à la commande `\markup` permet de construire des noms d'instruments particuliers, tels que

```
\set Staff.instrumentName = \markup {
  \column { "Clarinetti"
            \line { "in B" \smaller \flat } } }
c4 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 recourrir à l'instruction `\center-column` :
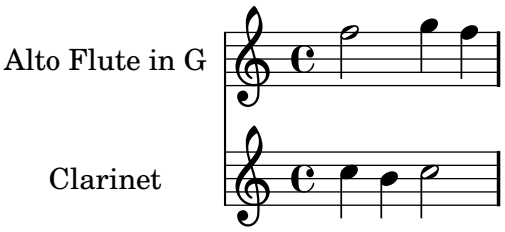
```
<<
  \new Staff {
    \set Staff.instrumentName = #"Flute"
    f2 g4 f
  }
  \new Staff {
    \set Staff.instrumentName = \markup \center-column {
      Clarinet
      \line { "in B" \smaller \flat }
    }
    c4 b c2
  }
>>
```
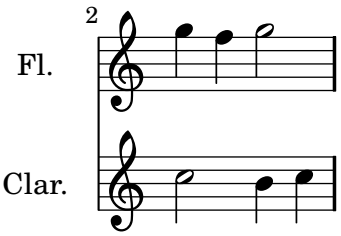


Lorsque le nom d'un instrument est relativement long, il est judicieux d'augmenter les retraits – *indent* – au sein du bloc `\layout` à l'aide des commandes `indent` et `short-indent`. Pour plus de plus amples détails sur ces réglages, reportez-vous au chapitre ⟨undefined⟩ [Dimensionnement horizontal], page ⟨undefined⟩.

```
\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm
}

\relative c'' <<
  \new Staff {
    \set Staff.instrumentName = #"Alto Flute in G"
    \set Staff.shortInstrumentName = #"Fl."
    f2 g4 f \break
    g4 f g2
  }
  \new Staff {
    \set Staff.instrumentName = #"Clarinet"
    \set Staff.shortInstrumentName = #"Clar."
    c,4 b c2 \break
    c2 b4 c
  }
>>
```



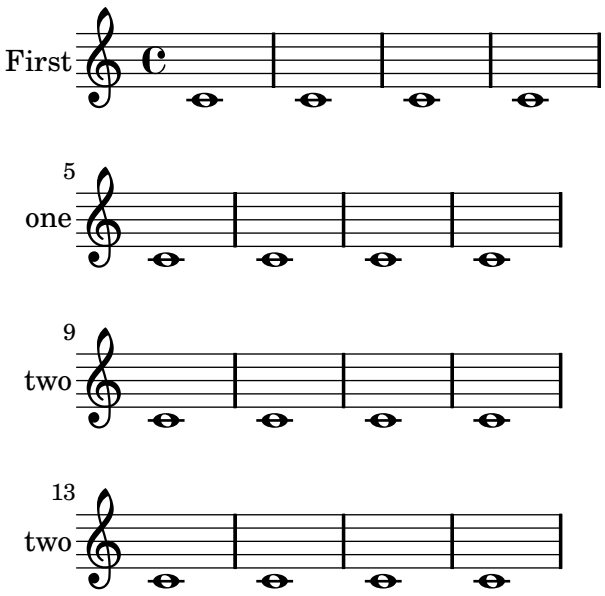


Des noms d'instruments peuvent s'utiliser dans d'autres contextes, tels que `GrandStaff`, `ChoirStaff`, ou `StaffGroup`, dès lors qu'on leur adjoint le graveur `Instrument_name_engraver`. Pour de plus amples informations sur la manière d'activer ou désactiver un graveur, voir ⟨undefined⟩ [Modification des greffons de contexte], page ⟨undefined⟩.

Vous pouvez changer le nom d'un instrument en cours de morceau. Notez cependant que la valeur de `instrumentName` ne s'affichera que sur la première portée :

```
\set Staff.instrumentName = #"First"
```

```
\set Staff.shortInstrumentName = #"one"
c1 c c c \break
c1 c c c \break
\set Staff.instrumentName = #"Second"
\set Staff.shortInstrumentName = #"two"
c1 c c c \break
c1 c c c \break
```

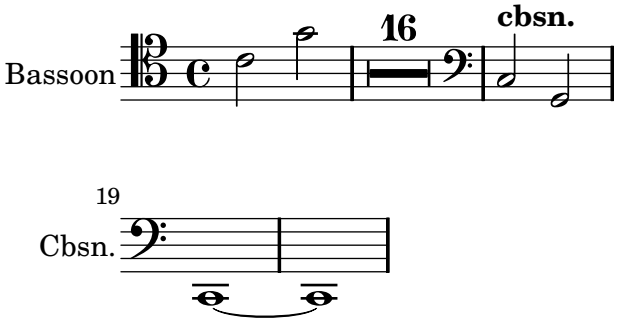


Lorsqu'un musicien doît changer d'instrument – piccolo et flûte, basson et contrebasson – , la commande `\addInstrumentDefinition`, couplée à l'instruction `\instrumentSwitch` permet de spécifier en détail les modifications intervenant lors du changement. La commande `\addInstrumentDefinition` prend deux arguments : une chaîne de caractères qui servira d'identificateur, et une liste d'associations de valeurs aux propriétés de ce nouvel instrument. Ces définitions devront être déclarées avant tout autre élément musical. L'instruction `\instrumentSwitch` se placera dans la musique au moment de la subtitution :

```
\addInstrumentDefinition #"contrabassoon"
  #`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
     (shortInstrumentName . "Cbsn.")
     (clefGlyph . "clefs.F")
     (middleCPosition . 6)
     (clefPosition . 2)
     (instrumentCueName . ,(make-bold-markup "cbsn."))
     (midiInstrument . "bassoon"))

\new Staff \with {
  instrumentName = #"Bassoon"
}
\relative c' {
  \clef tenor
  \compressFullBarRests
  c2 g'
  R1*16
  \instrumentSwitch "contrabassoon"
  c,,2 g \break
```

```
  c,1 ~ | c1
}
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Dimensionnement horizontal], page ⟨undefined⟩, ⟨undefined⟩ [Modification des greffons de contexte], page ⟨undefined⟩.

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, et vous évite de ressaisir la musique en question.
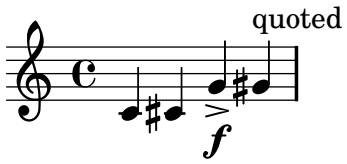
Avant qu’une partie ne puisse être mentionnée ailleurs, elle doit être considérée comme reproductible. C’est le but de l’instruction `\addQuote` qui prend en argument une chaîne d’identification et une expression musicale. Elle se place au niveau le plus haut, c’est à dire en dehors de tout bloc de musique :

```
flute = \relative c'' {
  a4 gis g gis
}
\addQuote "flute" { \flute }
```

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 – silences invisibles ou multi-mesures. Viendra alors s’insérer dans l’expression musicale le fragment correspondant de la voix originelle.

```
flute = \relative c'' {
  a4 gis g->\f gis^\markup{quoted}
}
\addQuote "flute" { \flute }
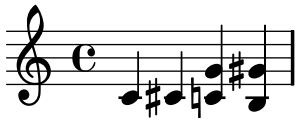
\relative c' {
  c4 cis \quoteDuring #"flute" { s2 }
}
```



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 :

```
flute = \relative c'' {
  a4 gis g gis
}
\addQuote "flute" { \flute }
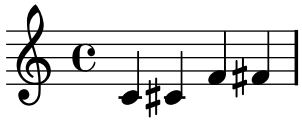
\relative c' {
  c4 cis \quoteDuring #"flute" { c4 b }
}
```



Les citations tiennent compte des transpositions, aussi bien celle de l'instrument d'origine que celle de la partie où elle intervient, dans la mesure où elles sont spécifiées par la commande `\transposition`. Voir [Instruments transpositeurs], page 18 pour plus de détails.

```
clarinet = \relative c'' {
  \transposition bes
  a4 gis g gis
}
\addQuote "clarinet" { \clarinet }

\relative c' {
  c4 cis \quoteDuring #"clarinet" { s2 }
}
```
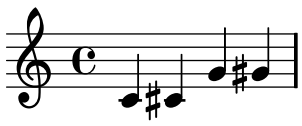


Les citations peuvent être « balisées » par un nom particulier afin de les utiliser de différentes manières. Pour de plus amples détails à ce propos, consultez le chapitre ⟨undefined⟩ [Utilisation de balises], page ⟨undefined⟩.

La propriété `quotedEventTypes` permet de définir précisément quels éléments de la voix originelle seront reproduits. Sa valeur par défaut est `#'(StreamEvent)` ; autrement dit, tout sera recopié. Lui affecter la valeur `#'(note-event rest-event tie-event)` fera que LilyPond reproduira les notes, silences et liaisons de prolongation, mais pas les articulations, annotations ni nuances.

```
clarinet = \relative c'' {
  a4 gis g->\f gis^\markup{quoted}
}
\addQuote "clarinet" { \clarinet }

\relative c' {
  \set Score.quotedEventTypes = #'(note-event rest-event tie-event)
  c4 cis \quoteDuring #"clarinet" { s2 }
```

```
}
```



Morceaux choisis

Citation d'une autre voix et transposition
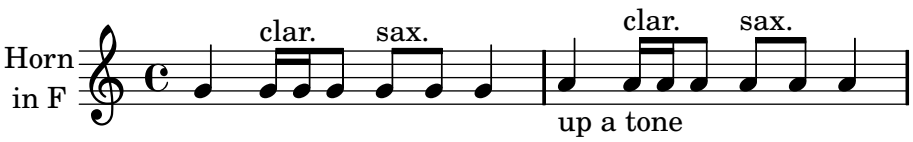
Les citations tiennent compte de la transposition, aussi bien celle de l'instrument d'origine que celle de la partie où elle intervient. Dans l'exemple suivant, tous les instruments sont en tonalité de concert et seront repris par un instrument en fa. Le destinataire de la citation peut à son tour transposer à l'aide de la commande `\transpose`. En pareil cas, toutes les hauteurs, y compris celle de la citation, seront transposées.

```
\addQuote clarinet {
  \transposition bes
  \repeat unfold 8 { d'16 d' d'8 }
}

\addQuote sax {
  \transposition es'
  \repeat unfold 16 { a8 }
}

quoteTest = {
  % french horn
  \transposition f
  g'4
  << \quoteDuring #"clarinet" { \skip 4 } s4^"clar." >>
  << \quoteDuring #"sax" { \skip 4 } s4^"sax." >>
  g'4
}

{
  \set Staff.instrumentName =
    \markup {
      \center-column { Horn \line { in F } }
    }
  \quoteTest
  \transpose c' d' << \quoteTest s4_"up a tone" >>
}
```



Citation d'une autre voix

Les types d'événement pris en charge pour la citation peuvent se régler avec la propriété `quotedEventTypes`. Par défaut, sa valeur est fixée à `(note-event rest-event tie-event beam-event tuplet-span-event)`, ce qui signifie que seuls les notes, silences, liaisons, ligatures
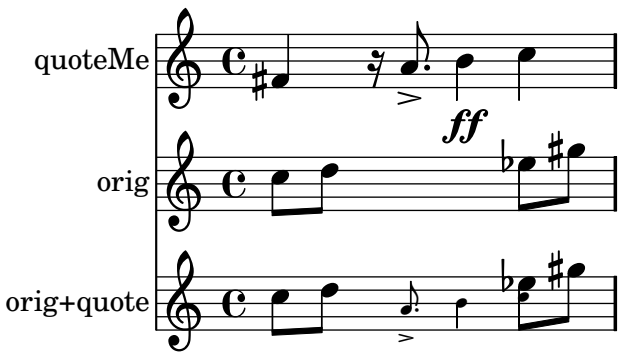
et nolets seront mentionnés par `\quoteDuring`. Dans l'exemple suivant, le quart de soupir n'est pas reproduit puisqu'il n'est pas mentionné parmi les `quotedEventTypes`.

Pour connaître la liste des types d'événements, reportez-vous au chapitre *Music classes* de la référence des propriétés internes.

```
quoteMe = \relative c' {
  fis4 r16 a8.-> b4\ff c
}
\addQuote quoteMe \quoteMe

original = \relative c'' {
  c8 d s2
  \once \override NoteColumn #'ignore-collision = ##t
  es8 gis8
}

<<
  \new Staff {
    \set Staff.instrumentName = #"quoteMe"
    \quoteMe
  }
  \new Staff {
    \set Staff.instrumentName = #"orig"
    \original
  }
  \new Staff \relative c'' <<
    \set Staff.instrumentName = #"orig+quote"
    \set Staff.quotedEventTypes =
      #'(note-event articulation-event)
    \original
    \new Voice {
      s4
      \set fontSize = #-4
      \override Stem #'length-fraction = #(magstep -4)
      \quoteDuring #"quoteMe" { \skip 2. }
    }
  >>
>>
```



Voir aussi

Manuel de notation : [Instruments transpositeurs], page 18, ⟨undefined⟩ [Utilisation de balises], page ⟨undefined⟩.

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

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 médiocre qualité.

Dans les versions précédentes de LilyPond (avant 2.11), `addQuote` était écrit entièrement en minuscules : `\addquote`.

Mise en forme d'une citation

La section précédente indiquait comment insérer des notes d'une autre voix. Nous allons maintenant voir une fonction musicale avancée, `\cueDuring`, qui facilite le formatage des petites notes.
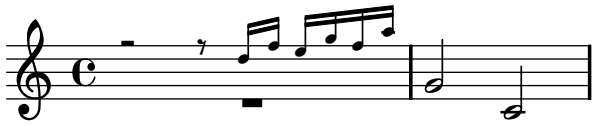
Sa syntaxe est :

```
\cueDuring #origine #voix musique
```

Des mesures issues de la partie d'***origine*** seront recopiées dans un contexte de `CueVoice`, créé implicitement, et synchronisées avec ***musique*** – habituellement un silence. L'apparition des petites notes initialise une polyphonie temporaire pour la portée concernée. L'argument ***voix*** détermine si ces petites notes seront attachées à la première ou à la seconde voix – `UP` pour la première ou `DOWN` pour la seconde.

```
oboe = \relative c'' {
  r2 r8 d16 f e g f a
  g8 g16 g g2.
}
\addQuote "oboe" { \oboe }

\new Voice \relative c'' {
  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```

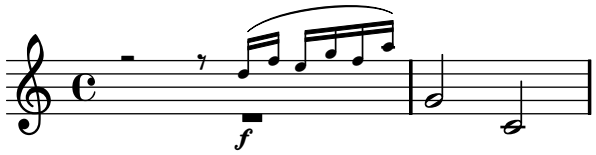


Dans cet exemple, il était nécessaire de déclarer explicitement le contexte `Voice`, sinon l'intégralité de l'expression musicale se serait retrouvée dans le contexte `CueVoice`.

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 nolets, mais pas les articulations, annotations ni nuances.

```
oboe = \relative c'' {
  r2 r8 d16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboe }
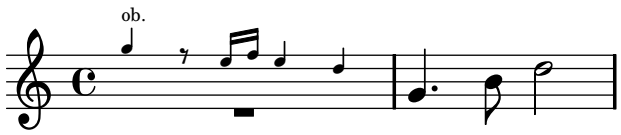
\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```



Le nom de l'instrument qui est répliqué sera imprimé dès lors que vous définirez la propriété `instrumentCueName` du contexte `CueVoice`.

```
oboe = \relative c''' {
  g4 r8 e16 f e4 d
}
\addQuote "oboe" { \oboe }

\new Staff \relative c'' <<
  \new CueVoice \with {
    instrumentCueName = "ob."
  }
  \new Voice {
    \cueDuring #"oboe" #UP { R1 }
    g4. b8 d2
  }
>>
```
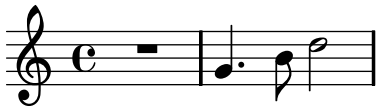


Lorsque la citation comporte une étiquette indiquant l'instrument qui joue, il est judicieux, au moment où la partie originale reprend sa place, de rappeler l'instrument et d'annuler toute autre modification introduite par cette citation. D'où l'utilité des commandes `\addInstrumentDefinition` et `\instrumentSwitch`. Pour plus d'information, reportez-vous au chapitre [Noms d'instrument], page 167.

La commande `killCues` permet de supprimer les notes d'une citation. Ceci est utile lorsque cette citation n'est pas imprimée dans un premier temps.

```
flute = \relative c''' {
  r2 cis2 r2 dis2
}
\addQuote "flute" { \flute }
```

```
\new Voice \relative c'' {
  \killCues {
    \cueDuring #"flute" #UP { R1 }
    g4. b8 d2
  }
}
```

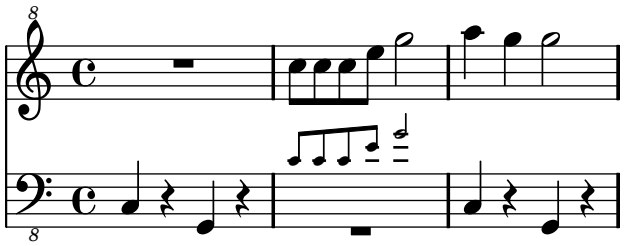


L'instruction `\transposedCueDuring` est particulièrement adaptée pour des instrument ayant une tessiture éloignée, comme dans le cas d'un piccolo cité dans une partie de contrebasson. Sa syntaxe est identique à celle de `\cueDuring`, à ceci près qu'elle nécessite un argument supplémentaire afin de spécifier la transposition à effectuer. Pour de plus amples informations sur la transposition, reportez-vous au chapitre [Instruments transpositeurs], page 18.

```
piccolo = \relative c''' {
  \clef "treble^8"
  R1
  c8 c c e g2
  a4 g g2
}
\addQuote "piccolo" { \piccolo }

cbassoon = \relative c, {
  \clef "bass_8"
  c4 r g r
  \transposedCueDuring #"piccolo" #UP c,, { R1 }
  c4 r g r
}
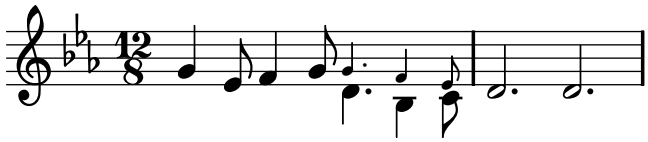
<<
  \new Staff = "piccolo" \piccolo
  \new Staff = "cbassoon" \cbassoon
>>
```



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 ⟨undefined⟩ [Utilisation de balises], page ⟨undefined⟩.

Un contexte `CueVoice` créé explicitement permet d'afficher des notes dans une taille plus petite dans le but, par exemple, d'indiquer des notes alternatives pour une voix un peu plus haute ou basse.

```
\time 12/8
\key ees \major
g4 ees8 f4 g8
\stemDown
<<
  { d4. bes4 c8 }
  \new CueVoice
  { g'4. f4 ees8 }
>>
\stemUp
d2. d2.
```



Voir aussi

Manuel de notation : [Instruments transpositeurs], page 18, [Noms d'instrument], page 167, ⟨undefined⟩ [Utilisation de balises], page ⟨undefined⟩.
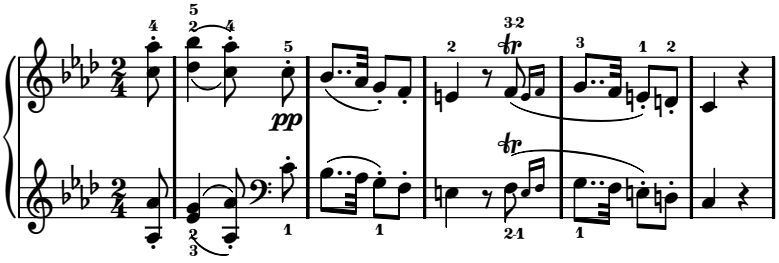
Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

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`.

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

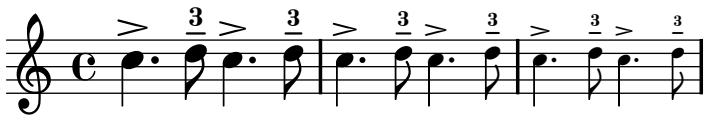
Nous allons voir ici comment mettre en exergue certains éléments à l'intérieur même de la portée.

Indication de la taille de fonte musicale

Le plus sûr moyen de régler la taille de la police, quel que soit le contexte, consiste à définir la propriété `fontSize`. Ceci ne modifiera en rien la taille des différents symboles tels que ligatures ou liaisons.
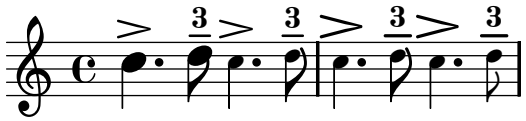
> **Note :** En matière de taille des éléments textuels, référez-vous au chapitre [Sélection de la fonte et de la taille], page 200.

```
\huge
c4.-> d8---3
\large
c4.-> d8---3
\normalsize
c4.-> d8---3
\small
c4.-> d8---3
\tiny
c4.-> d8---3
\teeny
c4.-> d8---3
```



En interne, la propriété `fontSize` d'un contexte aura pour effet de définir la propriété `font-size` pour tous les objets de rendu. La valeur de `font-size` est un entier représentant la taille proportionnellement à la hauteur standard de la portée en cours. Chaque incrément correspond à une augmentation d'environ 12 % de la taille de la police. Un pas de six aboutit exactement au doublement de la taille. La fonction Scheme `magstep` convertit le nombre affecté à `font-size` en facteur d'échelle. Vous pouvez aussi définir directement la propriété `font-size` de manière à n'affecter seulement que certains objets de rendu.

```
\set fontSize = #3
c4.-> d8---3
\override NoteHead #'font-size = #-4
c4.-> d8---3
\override Script #'font-size = #2
c4.-> d8---3
\override Stem #'font-size = #-5
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 10 points.

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`.

Commandes prédéfinies

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Voir aussi
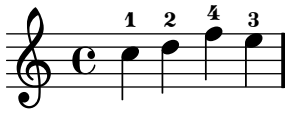
Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "font-interface" dans *Référence des propriétés internes*.

Doigtés
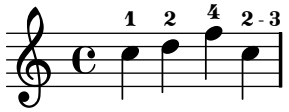
Les doigtés peuvent être indiqués comme suit : *note-chiffre_du_doigt*

```
c4-1 d-2 f-4 e-3
```



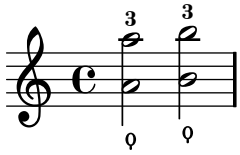
Pour les substitutions de doigts, on a recours à une indication textuelle (commande `\markup`) de doigté (commande `\finger`).

```
c4-1 d-2 f-4 c^\markup { \finger "2 - 3" }
```
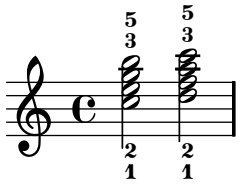


La commande `\thumb` peut être utilisée pour indiquer, par exemple dans une partition de violoncelle, si une note doit être jouée avec le pouce (*thumb* en anglais).

```
<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.

```
<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 ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩ à ce sujet.

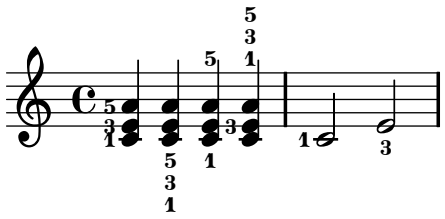
Morceaux choisis

Conrôle du positionnement des doigtés

Le positionnement des doigtés peut être contrôlé de manière très précise.

```
\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
```

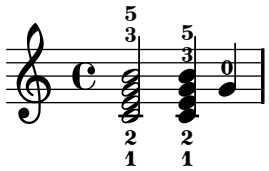
```
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



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. Néanmoins, il est possible d'annuler ce comportement.

```
\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>
}
```



Avoiding collisions with chord fingerings

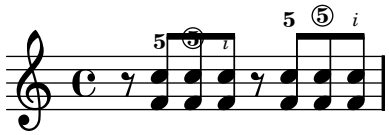
Fingerings and string numbers applied to individual notes will automatically avoid beams and stems, but this is not true by default for fingerings and string numbers applied to the individual notes of chords. The following example shows how this default behavior can be overridden.

```
\relative c' {
  \set fingeringOrientations = #'(up)
  \set stringNumberOrientations = #'(up)
  \set strokeFingerOrientations = #'(up)

  % Default behavior
  r8
  <f c'-5>8
  <f c'\5>8
  <f c'-\rightHandFinger #2 >8

  % Corrected to avoid collisions
  r8
  \override Fingering #'add-stem-support = ##t
  <f c'-5>8
  \override StringNumber #'add-stem-support = ##t
  <f c'\5>8
```

```
  \override StrokeFinger #'add-stem-support = ##t
  <f c'-\rightHandFinger #2 >8
}
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩

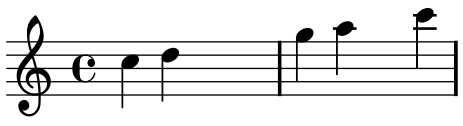
Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “FingeringEvent” dans *Référence des propriétés internes*, Section “fingering-event” dans *Référence des propriétés internes*, Section “Fingering_engraver” dans *Référence des propriétés internes*, Section “New_fingering_engraver” dans *Référence des propriétés internes*, Section “Fingering” dans *Référence des propriétés internes*.

Dictée à trous
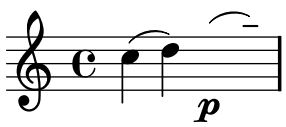
Les notes masquées – ou invisibles ou encore transparentes – sont utiles dans le cadre d’exercices de théorie ou de composition.

```
c4 d
\hideNotes
e4 f
\unHideNotes
g a
\hideNotes
b
\unHideNotes
c
```



Les objets de notation attachés à une note invisible ne seront pas masqués pour autant.

```
c4( d)
\hideNotes
e4(\p f)--
```



Commandes prédéfinies

`\hideNotes`, `\unHideNotes`.

Voir aussi

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

Manuel de notation : [Silences invisibles], page 49, ⟨undefined⟩ [Visibilité des objets], page ⟨undefined⟩, [Masquage de portées], page 163.

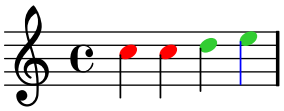
Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

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 listes des noms des couleurs disponibles se trouvent à l'annexe ⟨undefined⟩ [Liste des couleurs], page ⟨undefined⟩.

```
\override NoteHead #'color = #red
c4 c
\override NoteHead #'color = #(x11-color 'LimeGreen)
d
\override Stem #'color = #blue
e
```
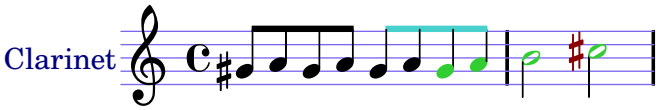


L'intégralité de la palette des couleurs définies pour X11 est accessible par la fonction Scheme `x11-color`. Cette fonction prend en argument une expression symbolique de la forme *'TaraTata* ou bien une chaîne de caractères comme "*TaraTata*". La première formulation est à la fois plus rapide à écrire et aussi plus efficace. Néanmoins, la deuxième forme permet d'accéder aux noms composés des couleurs de X11.

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.

```
\override Staff.StaffSymbol #'color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}
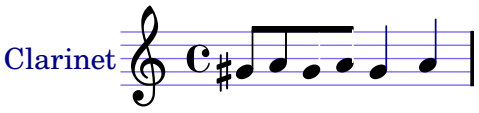
gis8 a
\override Beam #'color = #(x11-color "medium turquoise")
gis a
\override Accidental #'color = #(x11-color 'DarkRed)
gis a
\override NoteHead #'color = #(x11-color "LimeGreen")
gis a
% this is deliberate nonsense; note that the stems remain black
\override Stem #'color = #(x11-color 'Boggle)
b2 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*).

```
\override Staff.StaffSymbol #'color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}

\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 : ⟨undefined⟩ [Liste des couleurs], page ⟨undefined⟩, ⟨undefined⟩ [La commande d'affinage (*tweak*)], page ⟨undefined⟩.

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.
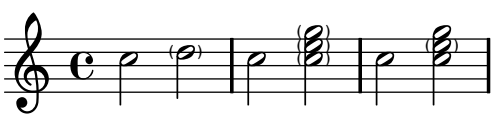
Les couleurs de X11 ne sont pas toutes perceptibles dans un navigateur internet. Aussi nous vous recommandons, dans le cadre d'une présentation multimedia, d'utiliser des couleurs de base `#blue`, `#green`, `#red` – bleu, vert, rouge.

Vous ne pouvez pas coloriser des notes à l'intérieur d'un accord avec `\override`. Si besoin est, utilisez `\tweak`. Pour plus de détails, consultez ⟨undefined⟩ [La commande d'affinage (*tweak*)], page ⟨undefined⟩.

Parenthèses
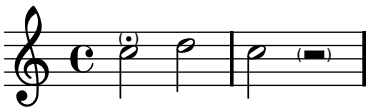
Des objets peuvent être mis entre parenthèses en saisissant `\parenthesize` juste avant l'événement musical. Si l'instruction préfixe un accord, chaque note le composant se présentera entre parenthèses. Vous pouvez aussi mettre individuellement entre parenthèses les notes d'un accord.

```
c2 \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.

```
c2-\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.

La direction des hampes peut être définie manuellement – voir ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩ à ce sujet.

Commandes prédéfinies
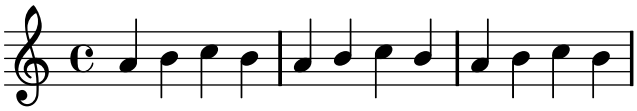
`\stemUp`, `\stemDown`, `\stemNeutral`.

Morceaux choisis

Default direction of stems on the center line of the staff

The default direction of stems on the center line of the staff is set by the `Stem` property `neutral-direction`.

```
\relative c'' {
  a4 b c b
  \override Stem #'neutral-direction = #up
  a4 b c b
  \override Stem #'neutral-direction = #down
  a4 b c b
}
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

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

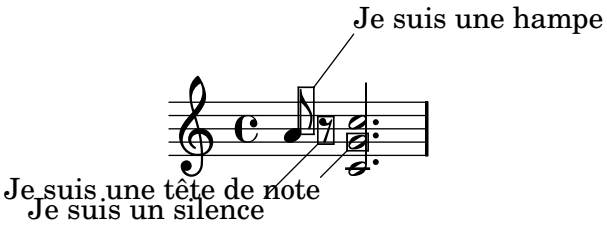
Nous allons nous intéresser ici à souligner des éléments inscrits dans la portée par des éléments qui lui seront externes.

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 :

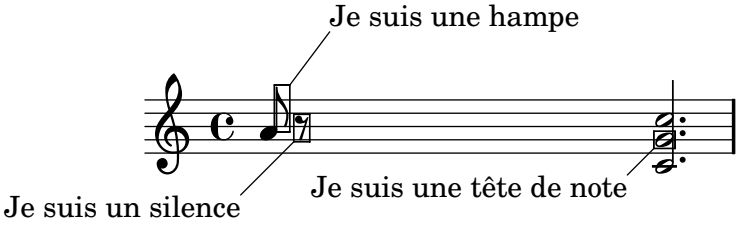
```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Je suis un silence" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "Je suis une tête de note" } c>2.
}
```



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 ont une influence sur les espacements, que l'on peut gérer ainsi :

```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonLengthOff
  \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*.

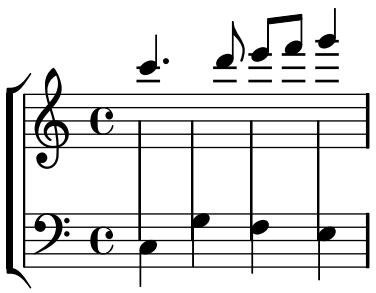
Quadrillage temporel

Vous pouvez tracer des lignes entre les portées, synchronisées avec les notes.

LilyPond a recours à deux graveurs distincts afin d’afficher le cadrillage : 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 c'' {
      \stemUp
      c'4. d8 e8 f g4
    }
    \new Staff \relative c {
    %% centre les lignes guides verticalement
      \clef bass
      \stemDown
      c4  g'  f  e
    }
  >>
}
```

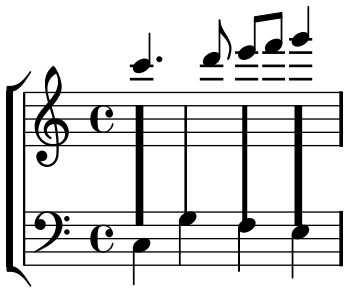


Morceaux choisis

Apparence du cadrillage temporel

Modifier certaines des propriétés du cadrillage 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
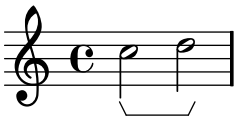
Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

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

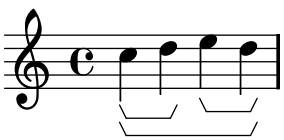
On utilise des crochets en analyse musicale, pour indiquer la structure d’une pièce.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c2\startGroup
  d\stopGroup
}
```



Les crochets d’analyses sont susceptibles d’être imbriqués :

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative c'' {
  c4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```
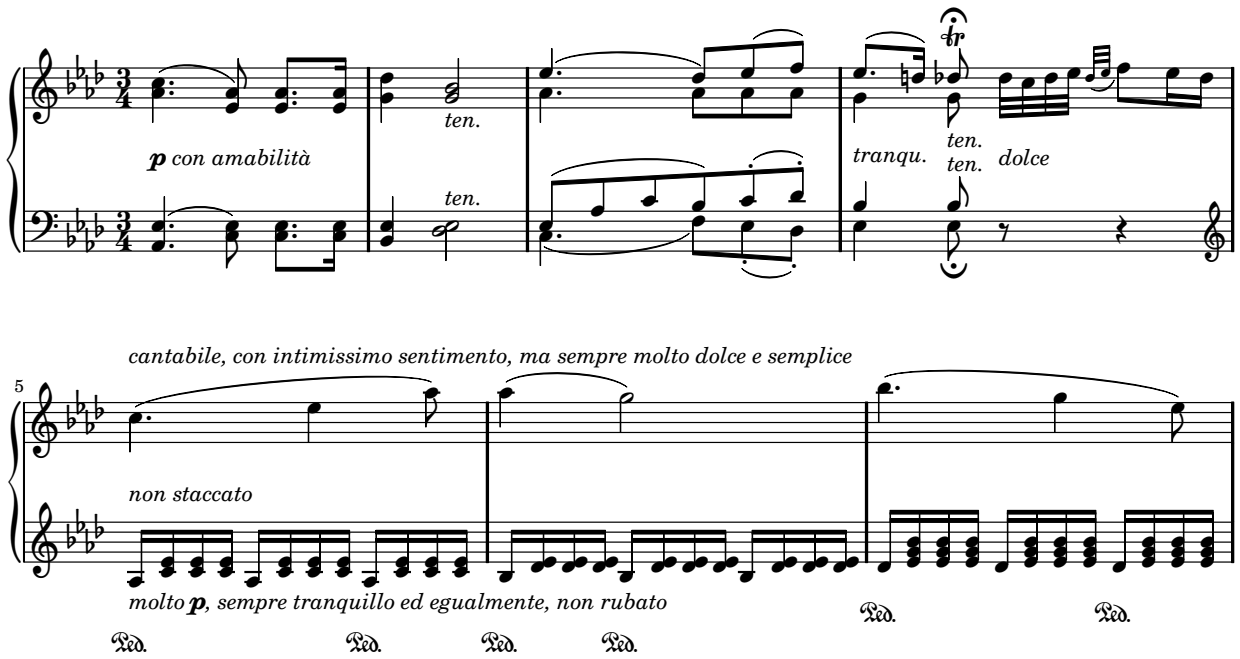


Voir aussi

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

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 “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.

Certains éléments de texte ne sont pas abordés ici mais dans des chapitres qui leur sont dédiés. C’est le cas de la Section 2.1 [Musique vocale], page 215 et des ⟨undefined⟩ [Titres et entêtes], page ⟨undefined⟩.

1.8.1 Ajout de texte

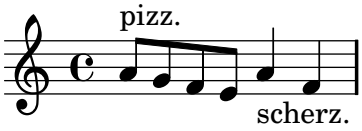
Cette partie constitue une introduction aux différentes manières d’ajouter du texte à une partition.

> **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 ⟨undefined⟩ [Codage du texte], page ⟨undefined⟩.

Commentaires textuels
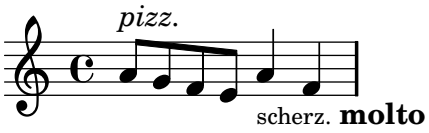
Vous pouvez ajouter à une partition des indications sous forme texuelle, comme dans l’exemple suivant. Ces indications se placeront manuellement au-dessus ou au-dessous de la portée selon la syntaxe utilisée – cf. ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

```
a8^"pizz." g f e a4-"scherz." f
```
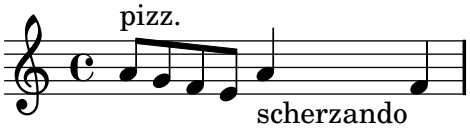


Cette syntaxe est en fait un raccourci. Des constructions plus élaborées d'annotation peuvent être obtenues en ayant recours à un bloc `\markup` et selon les préceptes énoncés dans Section 1.8.2 [Mise en forme du texte], page 198.

```
a8^\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.

```
a8^"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 99.

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 : Section 1.8.2 [Mise en forme du texte], page 198, ⟨undefined⟩ [Direction et positionnement], page ⟨undefined⟩.

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

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

Problèmes connus et avertissements

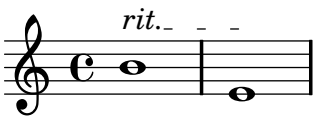
S'assurer que tous les éléments textuels et les paroles respectent les marges du document requiert des calculs relativement lourds ; c'est la raison pour laquelle LilyPond, par défaut, ne s'en préoccupe pas. Vous pouvez cependant l'y forcer en définissant

```
\override Score.PaperColumn #'keep-inside-line = ##t
```

Indication textuelle avec extension

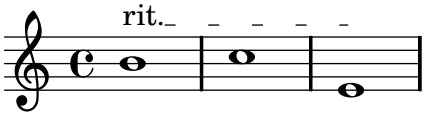
Certaines indications d'interprétation comme *rallentando*, *accelerando* ou *trilles*, s'inscrivent textuellement et se prolongent sur plusieurs notes à l'aide d'une ligne pleine, pointillée ou ondulée. Ces objets, que l'on appelle « extenseurs », se dessinent entre deux notes à l'aide de la syntaxe suivante :

```
\override TextSpanner #'(bound-details left text) = "rit."
b1\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 198.

```
\override TextSpanner #'(bound-details left text) =
  \markup { \upright "rit." }
b1\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 ⟨undefined⟩ [Styles de ligne], page ⟨undefined⟩. Les extenseurs sont rattachés au contexte `Dynamics` – cf. Section "Dynamics" dans *Référence des propriétés internes*.

Commandes prédéfinies

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.

Problèmes connus et avertissements

LilyPond ne peut traiter qu'un seul extenseur à la fois par voix.

Morceaux choisis
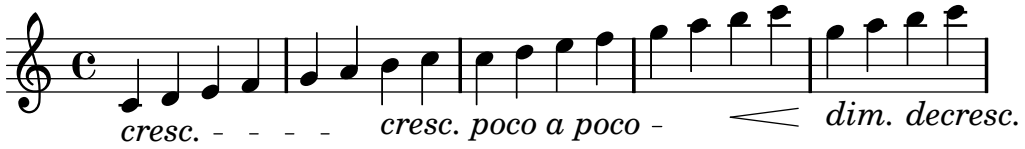
Extensions de nuance postfix

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")
% Redefine the existing \cresc, \dim and \decresc commands to use postfix syntax
cresc =
```

```
#(make-music 'CrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "cresc.")
dim =
#(make-music 'DecrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "dim.")
decresc =
#(make-music 'DecrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "decresc.")

\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\!
}
```



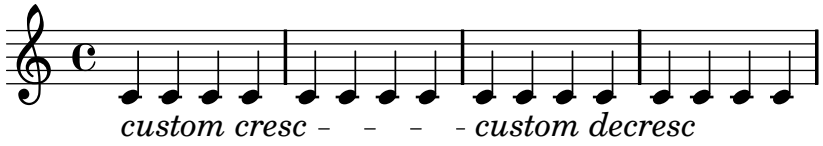
Personnalisation des extenseurs de nuance postfix

Il s'agit de fonctions postfix pour personnaliser l'extension des crescendos textuels. L'extension devrait débuter sur la première notte 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 (parser location mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (parser location 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 : ⟨undefined⟩ [Styles de ligne], page ⟨undefined⟩, [Nuances], page 101.
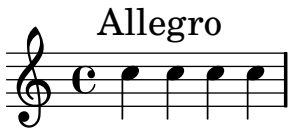
Morceaux choisis : Section “Texte” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextSpanner” dans *Référence des propriétés internes*, Section “Dynamics” dans *Référence des propriétés internes*.

Indications textuelles
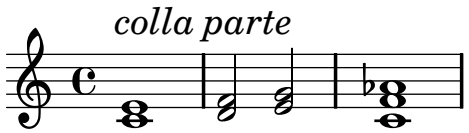
La commande `\mark` est tout d’abord conçue pour les [Indications de repère], page 90.

```
c4
\mark "Allegro"
c c c
```
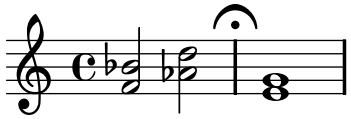


Cette syntaxe rend possible l’adjonction de n’importe quel texte à une barre de mesure. Ce texte peut être mis en forme de différentes manières dès lors qu’est utilisé un bloc `\markup`, comme indiqué au chapitre Section 1.8.2 [Mise en forme du texte], page 198.

```
<c e>1
\mark \markup { \italic { colla parte } }
<d f>2 <e g>
<c f aes>1
```



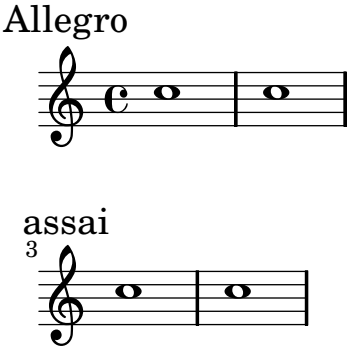
Elle peut aussi servir à insérer des signes de *coda* ou de *segno*, ou bien un point d’orgue, au dessus d’une barre de mesure. Couplez-la alors à la commande `\markup` pour avoir accès au symbole approprié, selon les indications contenues au chapitre [Notation musicale dans du texte formaté], page 208.

```
<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.

```
\mark "Allegro"
c1 c
\mark "assai" \break
c  c
```
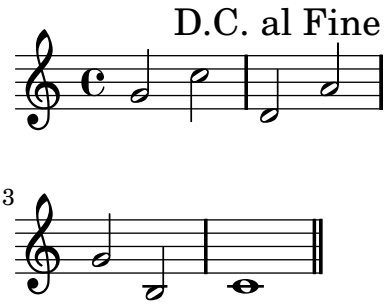


Morceaux choisis

Indication texuelle en fin de ligne

Les indications textuelles peuvent être imprimées à la fin d'une ligne plutôt qu'en tête de la suivante. Pensez alors à aligner l'extrémité droite de l'indication sur la barre de mesure.

```
\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 "||"
}
```



*Alignement des indications par rapport à divers objets*de notation

Les indications textuelles peuvent s'aligner par rapport à d'autres objets que des barres de mesure, tels que `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature`, ou `time-signature`.

Par défaut, les indications textuelles sont alignées avec le milieu des objets de notation. Bien entendu, vous pouvez modifier les propriétés des objets en question pour obtenir un autre résultat comme l'illustre la deuxième ligne de l'exemple. Dans le cas de portées multiples, ces réglages doivent être faits pour chacune d'entre elles.

```
\relative c' {
  e1

  % the RehearsalMark will be centered above the Clef
  \override Score.RehearsalMark #'break-align-symbols = #'(clef)
  \key a \major
  \clef treble
  \mark ""
  e1

  % the RehearsalMark will be centered above the TimeSignature
  \override Score.RehearsalMark #'break-align-symbols = #'(time-signature)
  \key a \major
  \clef treble
  \time 3/4
  \mark \markup { \char ##x2193 }
  e2.

  % the RehearsalMark will be centered above the KeySignature
  \override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
  \key a \major
  \clef treble
  \time 4/4
  \mark \markup { \char ##x2193 }
  e1

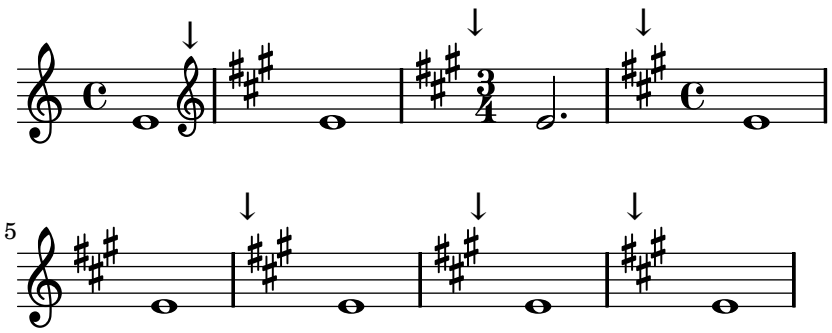
  \break
  e1

  % the RehearsalMark will be aligned with the left edge of the KeySignature
  \once \override Score.KeySignature #'break-align-anchor-alignment = #LEFT
  \mark \markup { \char ##x2193 }
  \key a \major
  e1

  % the RehearsalMark will be aligned with the right edge of the KeySignature
  \once \override Score.KeySignature #'break-align-anchor-alignment = #RIGHT
  \key a \major
  \mark \markup { \char ##x2193 }
  e1

  % the RehearsalMark will be aligned with the left edge of the KeySignature
  % and then shifted right by one unit.
  \once \override Score.KeySignature #'break-align-anchor = #1
```

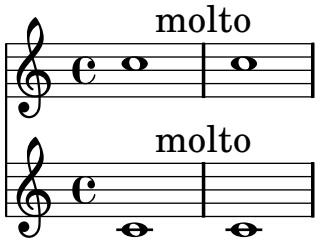
```
  \key a \major
  \mark \markup { \char ##x2193 }
  e1
}
```



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, vous pouvez forcer leur affectation 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 90, Section 1.8.2 [Mise en forme du texte], page 198, [Notation musicale dans du texte formaté], page 208, ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩.

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
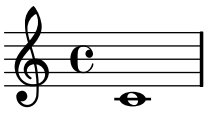
Un bloc `\markup` peut exister en lui-même, indépendamment de tout bloc `\score`, et venir en préambule par exemple – voir le chapitre ⟨undefined⟩ [Structure de fichier], page ⟨undefined⟩ à ce propos.

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

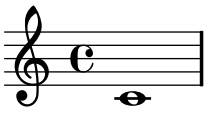
Tomorrow, and tomorrow, and tomorrow...

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 ⟨undefined⟩ [Plusieurs partitions dans un même ouvrage], page ⟨undefined⟩.

```
\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 211.

Commandes prédéfinies

`\markup`, `\markuplines`.

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 {
      \line { \italic { O sacred feast } }
      \line { \italic { in which Christ is received, } }
      \line { \italic { the memory of His Passion is renewed, } }
      \line { \italic { the mind is filled with grace, } }
      \line { \italic { and a pledge of future glory is given to us. } }
      \line { \italic { 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 198, ⟨undefined⟩ [Structure de fichier], page ⟨undefined⟩, ⟨undefined⟩ [Plusieurs partitions dans un même ouvrage], page ⟨undefined⟩, [Texte avec sauts de page], page 211.

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`.

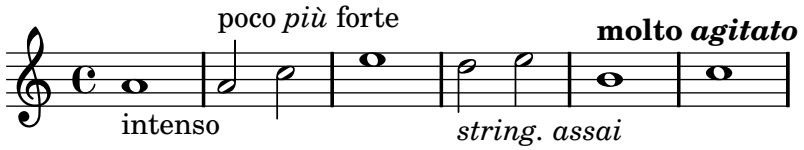
Introduction au formatage de texte

La commande `\markup` permet d'ajouter du texte et dispose de sa propre syntaxe que nous appellerons le « mode *markup* ».

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.
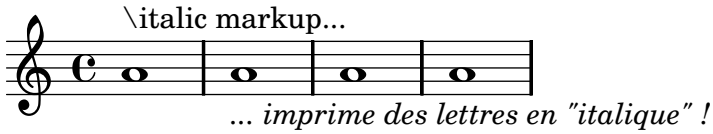
Contrairement aux indications simples, du type "entre guillemets", les blocs `\markup` peuvent contenir des expressions imbriquées ou d'autres commandes *markup*, dès lors qu'elles sont précédées du caractère `\`. Ces commandes n'affecteront que la première expression qui les suit.

```
a1-\markup intenso
a2^\markup { poco \italic più forte  }
c e1
d2_\markup { \italic "string. assai" }
e
b1^\markup { \bold { molto \italic  agitato } }
c
```



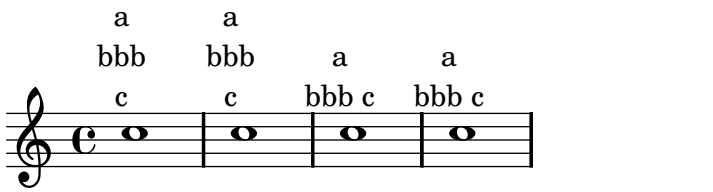
Un bloc `\markup` peut contenir du texte entre guillemets. De telles chaines seront considérées comme des expressions textuelles minimales ; à ce titre, toute commande de type *markup* ou tout caractère spécial – tel un `\` ou un `#` – sera imprimé littéralement et sans influer sur le formatage du texte. Il est de ce fait possible d'imprimer des guillemets informatiques `"` dès lors qu'ils sont précédés d'une oblique inverse.

```
a1^"\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 :

```
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" } }
```
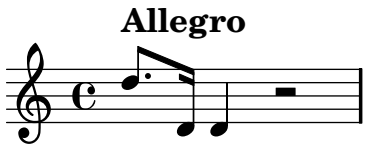


Vous pouvez stocker les étiquettes textuelles en tant que variables, et attacher ces identificateurs à des notes, comme ici :

```
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.8 [Text markup commands], page 515 (en anglais).

Voir aussi

Manuel de notation : Section A.8 [Text markup commands], page 515.

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

Fichiers d'initialisation : '`scm/markup.scm`'.
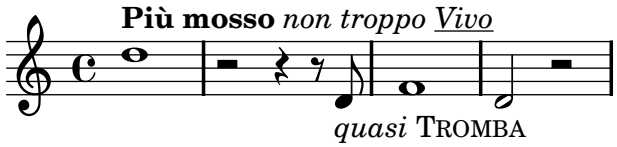
Problèmes connus et avertissements

Les erreurs de syntaxe en mode *markup* sont peu loquaces.

Sélection de la fonte et de la taille

Le mode *markup* autorise des changements élémentaires de la fonte :

```
d1^\markup {
  \bold { Più mosso }
  \italic { non troppo \underline Vivo }
}
r2 r4 r8
d,_\markup { \italic quasi \smallCaps Tromba }
f1 d2 r
```
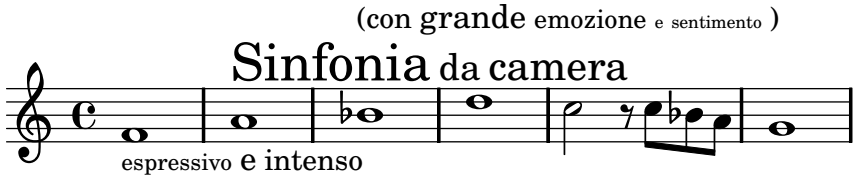


La taille des caractères se modifie de différentes manières :

- à partir de l'une des tailles standard prédéfinies,
- en étant définie en valeur absolue,
- en adoptant une valeur relative à celle précédemment définie.

Voici une illustration de ces trois différentes méthodes :

```
f1_\markup {
  \tiny espressivo
  \large e
  \normalsize intenso
}
a^\markup {
  \fontsize #5 Sinfonia
  \fontsize #2 da
  \fontsize #3 camera
}
bes^\markup { (con
  \larger grande
```

```
  \smaller emozione
  \magnify #0.6 { e sentimento } )
}
d c2 r8 c bes a g1
```



Vous pouvez imprimer du texte en indice ou en exposant. Celui-ci sera dans un 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 106 et [Indications de reprise manuelles], page 125.

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.8.1 [Font], page 515.

Pour savoir comment personnaliser des fontes, reportez-vous au chapitre Section 1.8.3 [Fontes], page 211.

Commandes prédéfinies

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

Voir aussi

Manuel de notation : Section A.8.1 [Font], page 515, [Personnalisation des indications de nuance], page 106, [Indications de reprise manuelles], page 125, Section 1.8.3 [Fontes], page 211.

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

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

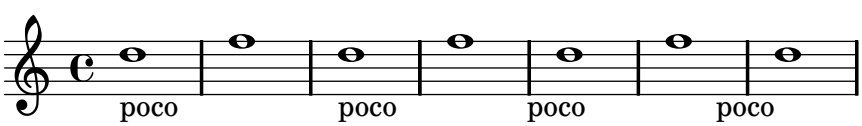
Fichiers d'initialisation : '`scm/define-markup-commands.scm`'.

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*.
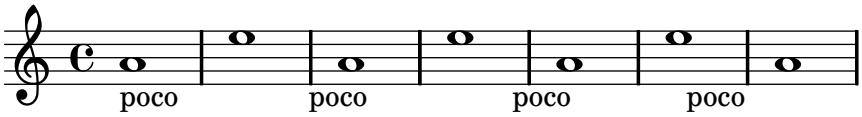
Les objets de type *markup* peuvent s'aligner de différentes manières. Une indication textuelle est par défaut alignée sur son extrémité gauche. Dans l'exemple qui suit, il n'y a aucune différence entre les deux premiers *markups*.

```
d1-\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 :

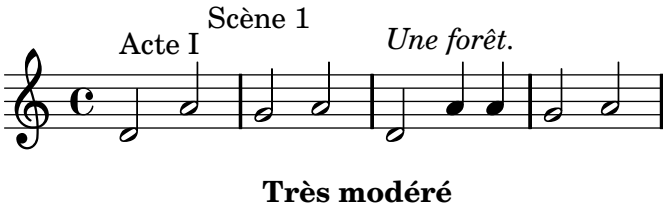
```
a1-\markup { \halign #-1 poco }
e'
a,-\markup { \halign #0 poco }
e'
a,-\markup { \halign #0.5 poco }
e'
a,-\markup { \halign #2 poco }
```



Certains objets possèdent leurs propres procédures d'alignement, qui annuleront toute spécification d'alignement que vous pourriez affecter à un argument de type *markup* que vous leur auriez fourni. La solution consiste alors à déplacer l'intégralité de ces objets *markup*, comme indiqué par exemple au chapitre [Indications textuelles], page 193.

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 néanmois 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.

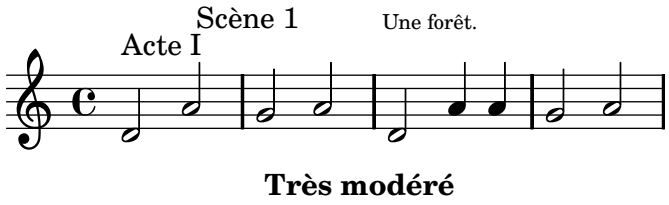
```
d2^\markup {
  Acte I
  \raise #2 { Scène 1 }
}
a'
g_\markup {
  \null
  \lower #4 \bold { Très modéré }
}
a
d,^\markup {
  \raise #4 \italic { Une forêt. }
}
a'4 a g2 a
```



Certaines commandes permettent de régler l'alignement des objets textuels en mode *markup*, tant au niveau horizontal que vertical. Tout objet soumis à ces commandes doit être précédé d'un point d'ancrage.

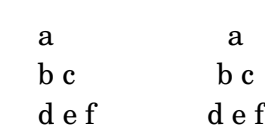
```
d2^\markup {
  Acte I
  \translate #'(-1 . 2) "Scène 1"
}
```

```
a'
g_\markup {
  \null
  \general-align #Y #3.2 \bold "Très modéré"
}
a
d,^\markup {
  \null
  \translate-scaled #'(-1 . 2) \teeny "Une forêt."
}
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 }
  }
}
```



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 multi-ligne 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 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.8.2 [Align], page 524.

Voir aussi

Manuel d'initiation : Section "Déplacement d'objets" dans *Manuel d'initiation*.

Manuel de notation : Section A.8.2 [Align], page 524, [Indications textuelles], page 193.

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

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

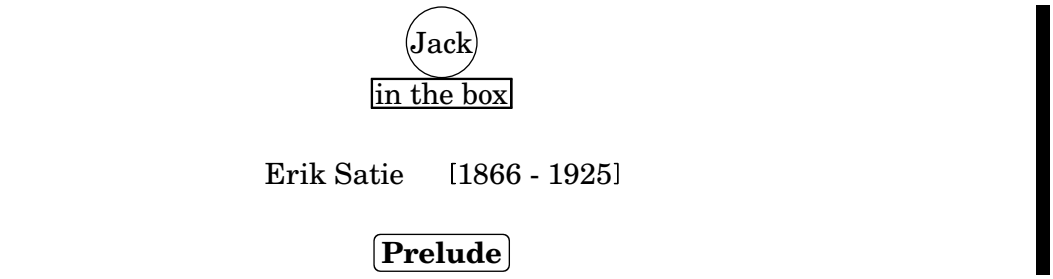
Fichiers d'initialisation : '`scm/define-markup-commands.scm`'.

Éléments graphiques dans du texte formaté

Vous pouvez, grâce aux 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
  }
}
```



Certaines directives peuvent nécessiter d'accroître l'espacement autour du texte – voir l'annexe Section A.8.2 [Align], page 524 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"
  }
  \pad-around #3
      "String quartet keeps very even time,
Flute quartet keeps very uneven time."
}
```

Charles Ives (1874 - 1954)

THE UNANSWERED QUESTION

A Cosmic Landscape

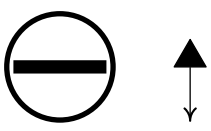
Largo to Presto

String quartet keeps very even time, Flute quartet keeps very uneven 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*.

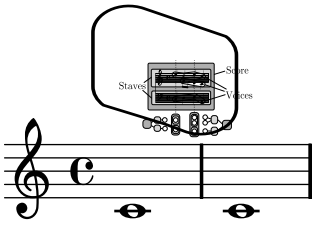
```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

  \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 :

```
c1^\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.8.3 [Graphic], page 537 répertorie les différentes commandes en matière de graphisme.

Voir aussi

Manuel de notation : Section A.8.3 [Graphic], page 537, Section 1.7 [Annotations éditoriales], page 177.

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

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

Fichiers d'initialisation : '`scm/define-markup-commands.scm`', '`scm/stencil.scm`'.

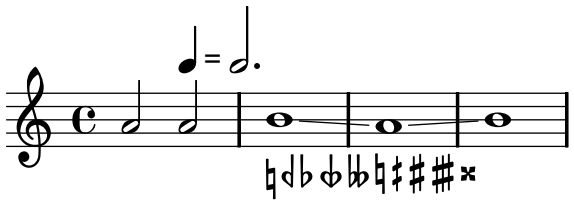
Notation musicale dans du texte formaté

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* :
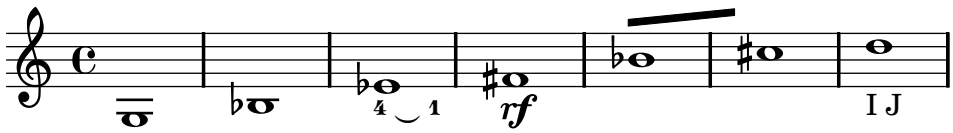
```
a2 a^\markup {
  \note #"4" #1
  =
  \note-by-number #1 #1 #1.5
}
b1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a1_\markup {
```

```
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b
```



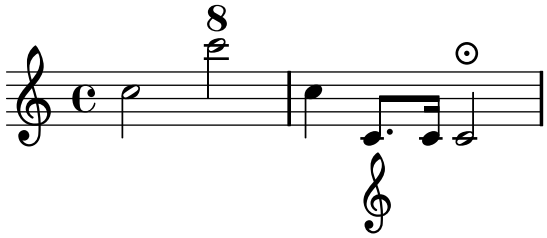
Le mode *markup* permet d'accéder à d'autres objets de notation :

```
g1 bes
ees-\markup {
  \finger 4
  \tied-lyric #"~"
  \finger 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 ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩.

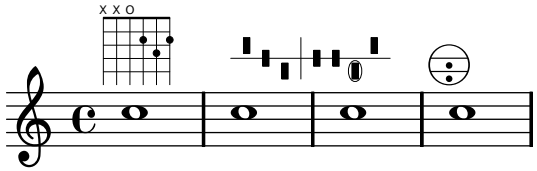
```
c2
c'^\markup { \musicglyph #"eight" }
c,4
c,8._\markup { \musicglyph #"clefs.G_change" }
c16
c2^\markup { \musicglyph #"timesig.neomensural94" }
```



Le sous-chapitre [Tout savoir sur les fontes], page 212 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 :
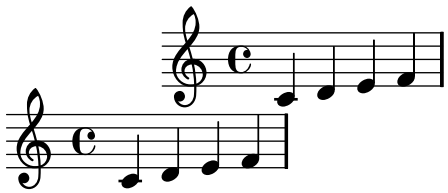
```
c1^\markup {
  \fret-diagram-terse #"x;x;o;2;3;2;"
}
c^\markup {
  \harp-pedal #"^-v|--ov^"
}
c
c^\markup {
  \combine
    \musicglyph #"accordion.discant"
    \combine
      \raise #0.5 \musicglyph #"accordion.dot"
      \raise #1.5 \musicglyph #"accordion.dot"
}
```



La documentation sur ces diagrammes se trouvent à l'annexe Section A.8.5 [Instrument Specific Markup], page 546.

Rien ne s'oppose à ce qu'une étiquette ne comporte une partition. Il suffit que l'expression en question contienne un bloc `\score` incluant un bloc `\layout`.

```
\relative {
  c4 d^\markup {
    \score {
      \relative { c4 d e f }
      \layout { }
    }
  }
  e f
}
```



Les différentes commandes *markup* relatives à la notation musicale sont répertoriées à l'annexe Section A.8.4 [Music], page 543.

Voir aussi

Manuel de notation : Section A.8.4 [Music], page 543, ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩, [Tout savoir sur les fontes], page 212.

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

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

Fichiers d'initialisation : ‘`scm/define-markup-commands.scm`’, ‘`scm/fret-diagrams.scm`’, ‘`scm/harp-pedals.scm`’.

Texte avec sauts de page

Alors que `\markup` s'utilise pour traiter un bloc de texte insécable, `\markuplines` permet, employé en tête de partition, d'obtenir un bloc de lignes réparties différemment et, le cas échéant, sur plusieurs pages.

```
\markuplines {
  \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.9 [Text markup list commands], page 554.

Voir aussi

Manuel de notation : Section A.9 [Text markup list commands], page 554,

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

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

Fichiers d'initialisation : ‘`scm/define-markup-commands.scm`’.

Commandes prédéfinies

`\markuplines`.

1.8.3 Fontes
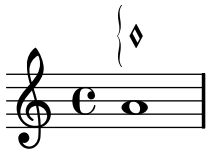
Ce chapitre est consacré aux fontes et polices de caractère, à leur gestion. Vous y apprendrez aussi comment changer de fonte en cours de partition.

Tout savoir sur les fontes

La gestion des fontes est assurée par plusieurs bibliothèques : FontConfig se charge de répertorier les différentes fontes installées sur votre système ; quant à Pango, elle se charge plus particulièrement de leur rendu.

Les fontes musicales peuvent se décrire comme un jeu de glyphes spécifiques regroupés dans plusieurs familles. L'exemple qui suit montre la syntaxe à utiliser pour accéder, en mode *markup*, aux différentes fontes `feta` non textuelles de LilyPond.

```
a1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup #"brace120"
    \override #'(font-encoding . fetaNumber)
    \column { 1 3 }
    \override #'(font-encoding . fetaDynamic)
    sf
    \override #'(font-encoding . fetaMusic)
    \lookup #"noteheads.s0petrucci"
  }
}
```
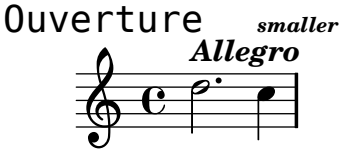


Tous ces glyphes, à l'exception des accolades qui sont regroupées dans `fetaBraces`, sont accessibles avec la syntaxe indiquée dans [Notation musicale dans du texte formaté], page 208.

Une remarque s'impose au sujet des glyphes contenus dans `fetaBraces` : chacun d'eux est formé du mot *brace* auquel est accolé un numéro d'ordre. Lorsque vous désirez imprimer une accolade, vous devez la « chercher » par son numéro d'ordre – d'où la fonction `\lookup` de l'exemple ci-dessus –, tout en sachant qu'il est compris entre `0` (la plus petite) et `575` (la plus grande). Vous serez souvent amené à procéder par tatonnement pour arriver au résultat optimal. Par ailleurs, `fetaBraces` ne comporte que des accolades ouvrantes. Pour obtenir une accolade fermante, il suffit d'appliquer une rotation au glyphe sélectionné, comme indiqué au chapitre ⟨undefined⟩ [Rotation des objets], page ⟨undefined⟩.

Vous disposez de trois familles de fontes textuelles : *roman* pour la police sérif ou avec empattement – fixée par défaut à New Century Schoolbook –, une police sans empattement (*sans* sérif) et une à chasse fixe (ou monospace) – les deux dernières étant déterminées par l'installation de Pango.

Chaque famille dispose en principe de différents styles et niveau de gras. 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
d2.^\markup "Allegro"
\override Voice.TextScript #'font-size = #-3
c4^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 200 :

```
\markup {
  \column {
    \line {
      \override #'(font-shape . italic)
      \override #'(font-size . 4)
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter)
      {
        \override #'(font-series . bold)
        re
        di
      }
      \override #'(font-family . sans)
      Creta
    }
  }
}
```



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 213 et [Choix des fontes par défaut], page 214.

Voir aussi
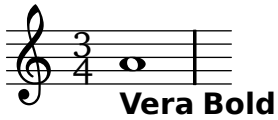
Manuel de notation : ⟨undefined⟩ [La fonte Feta], page ⟨undefined⟩, [Notation musicale dans du texte formaté], page 208, [Sélection de la fonte et de la taille], page 200, Section A.8.1 [Font], page 515.

Attribution d'une fonte en particulier

Vous pouvez utiliser n'importe quelle police installée sur votre système, du moment où elle est accessible par Fontconfig, en respectant la syntaxe suivante :

```
\override Staff.TimeSignature #'font-name = #"Bitstream Charter"
\override Staff.TimeSignature #'font-size = #2
\time 3/4

a1_\markup {
  \override #'(font-name . "Vera Bold")
    { Vera Bold }
}
```



Pour obtenir la liste de toutes les polices disponibles sur votre machine, lancez

```
lilypond -dshow-available-fonts toto
```

(quel qu'il soit, le dernier argument est obligatoire).

Voir aussi

Manuel de notation : [Tout savoir sur les fontes], page 212, [Choix des fontes par défaut], page 214.

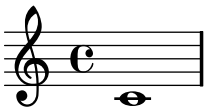
Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Choix des fontes par défaut

Vous pouvez tout à fait préférer un autre jeu de polices par défaut que celui de LilyPond. Il vous faudra alors spécifier les différentes familles, en respectant l'ordre *roman*, *sans empattement* et *monospace*, comme dans l'exemple suivant. Pour plus d'explications sur les fontes, relisez [Tout savoir sur les fontes], page 212.

```
\paper  {
  myStaffSize = #20
  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans"
                          "Luxi Mono"
                           (/ myStaffSize 20)))
}

\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```



roman, sans, typewriter.

Voir aussi

Manuel de notation : [Tout savoir sur les fontes], page 212, [Attribution d'une fonte en particulier], page 213, [Sélection de la fonte et de la taille], page 200, Section A.8.1 [Font], page 515.

2 Notation spécialisée

Ce chapitre explique comment créer la notation musicale spécifique à certains intruments ou certaines époques.

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 198.
- Une feuille de chant s'obtient en combinant une partie vocale et des accords. La syntaxe appropriée à ces derniers est abordée dans ⟨undefined⟩ [Notation des accords], page ⟨undefined⟩.
- L'impression d'un *ambitus* – ou tessiture – que l'on trouve en tête de certaines partitions, est abordée dans [Ambitus], page 27.
- L'utilisation de clefs traditionnelles est abordée dans [Clefs], page 12.
- LilyPond sait reproduire les usages historiques, comme l'explique ⟨undefined⟩ [Notations anciennes], page ⟨undefined⟩.

Opéra

En cours de rédaction.

Cahier de chants

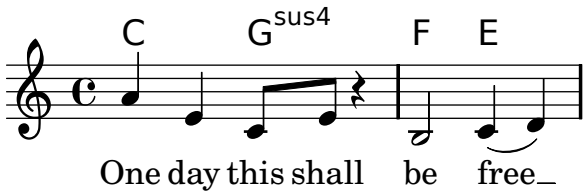
En cours de rédaction.

Morceaux choisis

Chanson simple

Assembler des noms d'accords, une mélodie et des paroles permet d'obtenir la partition d'un chanson :

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Manuel de notation : ⟨undefined⟩ [Notation des accords], page ⟨undefined⟩.

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 29.

Chants

En cours de rédaction.

Musique vocale ancienne

En cours de rédaction.

Voir aussi

Manuel de notation : ⟨undefined⟩ [Notations anciennes], page ⟨undefined⟩.

2.1.2 Saisie des paroles

Tout savoir sur les paroles

Le fait que LilyPond travaille sur des fichiers textuels nous amène au constat suivant : les paroles ne doivent pas être interprétées comme des notes – le caractère `a` ne saurait être confondu avec la note `a` (un *la* pour les non latinistes). C'est la raison pour laquelle nous utiliserons un mode dédié.

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 vous permet de saisir des paroles, ainsi que leur ponctuation, de telle sorte que le caractère `a` ne sera plus interprété comme une note, 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 { Twin-4 kle4 twin- kle litt- le star2 }
```

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`.

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 ponctiuation – étant considéré comme partie intégrante de cette même syllabe. L'exemple suivant comporte une faute de frappe évidente :

```
\lyricmode { 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.

De la même manière, un point concluant une suite de lettres sera partie intégrante de la chaîne résultante. Par conséquent, il est impératif d'insérer des espaces lorsque vous modifiez les propriétés d'une commande. En d'autres termes, **ne saisissez pas**

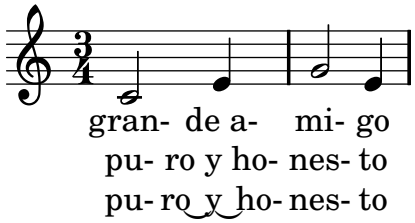
```
\override Score.LyricText #'font-shape = #'italic
```

mais plutôt

```
\override Score . LyricText #'font-shape = #'italic
```

Pour attribuer plus d'une syllabe à une même note, vous pouvez mettre ces syllabes entre guillemets, ou bien remplacer l'espace par un caractère souligné (`_`), ou encore utiliser un tilde (`~`) pour obtenir une liaison entre les syllabes.
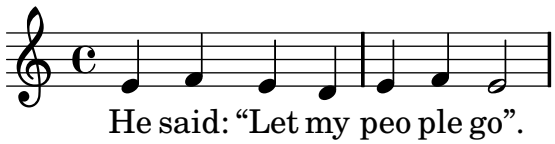
```
\time 3/4
\relative c' { c2 e4 g2 e4 }
\addlyrics { gran- de_a- mi- go }
\addlyrics { pu- "ro y ho-" nes- to }
\addlyrics { pu- ro~y~ho- nes- to }
```



Cette liaison adaptée aux paroles correspond au caractère Unicode `U+203F`, et n'apparaîtra dans la partition que s'il existe une police incluant ce symbole (par exemple DejaVuLGC) installée sur le système. De plus amples informations concernant les fontes textuelles ou non sont disponibles au chapitre Section 1.8.3 [Fontes], page 211.
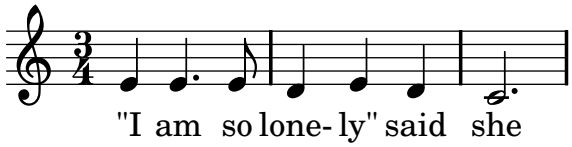
Pour utiliser des lettres accentuées ou des caractères spéciaux – cœurs ou guillemets inversés par exemple – il suffit de les insérer dans le fichier et de veiller à le sauvegarder avec le codage UTF-8. Voir à ce sujet ⟨undefined⟩ [Codage du texte], page ⟨undefined⟩ pour plus de détails.

```
\relative c' { e4 f e d e f e2 }
\addlyrics { He said: \Let my peo ple go". }
```



Pour utiliser des guillemets informatiques standard, faites-les précéder d'une barre oblique inverse :

```
\relative c' { \time 3/4 e4 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.

Pour définir un identificateur contenant des paroles, il faut utiliser la fonction `lyricmode`.

```
verseOne = \lyricmode { Joy to the world the Lord is come }
\score {
  <<
    \new Voice = "one" \relative c'' {
      \autoBeamOff
      \time 2/4
      c4 b8. a16 g4. f8 e4 d c2
    }
    \addlyrics { \verseOne }
  >>
}
```

Voir aussi

Manuel de notation : Section 1.8.3 [Fontes], page 211.

Référence des propriétés internes : Section “LyricText” dans *Référence des propriétés internes*, Section “LyricSpace” dans *Référence des propriétés internes*.
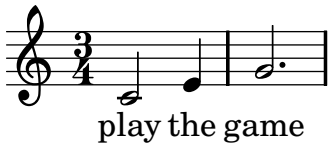
Écriture de chants simples

Le plus simple pour ajouter des paroles à une mélodie est d’adjoindre

```
\addlyrics { les paroles }
```

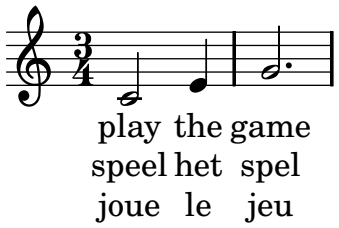
à la mélodie. En voici un exemple :

```
\time 3/4
\relative { c2 e4 g2. }
\addlyrics { play the game }
```



On peut ajouter davantage de couplets en multipliant le nombre de sections `\addlyrics`.

```
\time 3/4
\relative { c2 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. Dans ce cas, mieux vaut employer `\lyricsto` et `\lyricmode`, comme nous le verrons dans [Tout savoir sur les paroles], page 216.

Travail avec des paroles et variables

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.

```
verseOne = \lyricmode { Joy to the world the Lord is come }
\score {
 <<
   \new Voice = "one" \relative c'' {
     \autoBeamOff
     \time 2/4
     c4 b8. a16 g4. f8 e4 d c2
   }
   \addlyrics { \verseOne }
 >>
}
```

Pour une organisation différente ou plus complexe, mieux vaut commencer par définir la hiérarchie des portées et des paroles,

```
\new ChoirStaff <<
  \new Voice = "soprano" { musique }
  \new Lyrics = "sopranoParoles" { s1 }
  \new Lyrics = "tenorParoles" { s1 }
  \new Voice = "tenor" { musique }
>>
```

puis combiner correctement mélodies et paroles :

```
\context Lyrics = sopranoParoles \lyricsto "soprano"
les paroles
```

Le résultat final ressemblera à

```
<<\new ChoirStaff << définition de la musique >>
 \lyricsto "soprano" etc.
 \lyricsto "alto" etc.
etc.
>>
```

Voir aussi

Référence des propriétés internes : Section "LyricCombineMusic" dans *Référence des propriétés internes*, Section "Lyrics" dans *Référence des propriétés internes*.

2.1.3 Alignement des paroles sur une mélodie

Les paroles s'alignent par rapport à une mélodie soit de manière automatique, soit manuellement dès lors que vous aurez spécifié la durée de chaque syllabe. Vous pouvez y adjoindre des blancs, des traits d'union ou des lignes de prolongation.

Avant d'être imprimées, les paroles sont interprétées par le programme dans le contexte `Lyrics`.

```
\new Lyrics \lyricmode ...
```

Il y a deux grandes méthodes pour gérer le positionnement horizontal des syllabes :

- en alignant automatiquement les paroles sur une mélodie ou une autre voix, en ayant recours à `\addlyrics` ou `\lyricsto` ;
- en affectant explicitement à chaque syllabe une durée, au sein du contexte `\lyricmode`

Le contexte `Voice` qui contient la mélodie sur laquelle doivent s'aligner les paroles doit demeurer actif, au risque de perdre les paroles qui suivent. C'est le cas lorsque la voix en question est muette pendant un moment. La manière de garder un contexte actif est abordée plus en détail au chapitre ⟨undefined⟩ [Conservation d'un contexte], page ⟨undefined⟩.

Durée automatique des syllabes

Les paroles peuvent être automatiquement alignées sous une mélodie. Il suffit pour cela de combiner la mélodie et les paroles avec la commande `\lyricsto`.

```
\new Lyrics \lyricsto nom ...
```

Cette commande adapte les paroles aux notes de la voix (contexte `Voice` dans le jargon LilyPond) *nom*. Ce contexte `Voice` doit exister avant l'indication des paroles avec `\lyricsto`. La commande `\lyricsto` introduit automatiquement le mode `\lyricmode` ; il n'est alors pas nécessaire de rajouter `\lyricmode`.

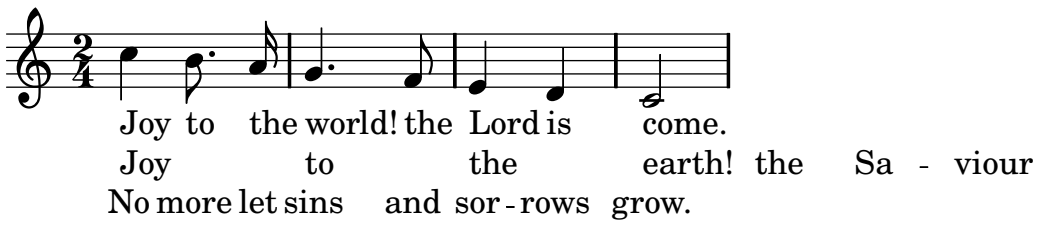
L'exemple suivant récapitule les différentes manières de saisir des paroles.

```
<<
  \new Voice = "one" \relative c'' {
    \autoBeamOff
    \time 2/4
    c4 b8. a16 g4. f8 e4 d c2
  }

% not recommended: left-aligned syllables
  \new Lyrics \lyricmode { Joy4 to8. the16 world!4. the8 Lord4 is come.2 }

% wrong: durations needed
  \new Lyrics \lyricmode { Joy to the earth! the Sa -- viour reigns. }

%correct
  \new Lyrics \lyricsto "one" { No more let sins and sor -- rows grow. }
>>
```



Le deuxième couplet n'est pas correctement disposé, aucune durée n'ayant été spécifiée. Dans un tel cas, il aurait mieux valu utiliser `\lyricsto`, comme dans le troisième couplet.

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 }
```

est exactement la même chose que

```
\new Voice = "blah" { musique }
\new Lyrics \lyricsto "blah" { PAROLES }
```

Durée explicite des syllabes

On peut aussi se passer de `\addlyrics` et de `\lyricsto` 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 :

```
play2 the4 game2.
sink2 or4 swim2.
```
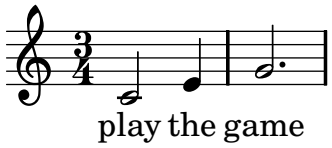
La mélodie associée à ces paroles peut être spécifiée par la propriété `associatedVoice` :

```
\set associatedVoice = #"lala"
```

La valeur que vous attribuez à cette propriété (ici `"lala"`) doit désigner un contexte `Voice`, sans quoi les mélismes ne seront pas imprimés correctement.

Voici un exemple de cette manière de procéder :

```
<< \new Voice = "melody" {
    \time 3/4
    c2 e4 g2.
 }
 \new Lyrics \lyricmode {
   \set associatedVoice = #"melody"
   play2 the4 game2.
 } >>
```



Voir aussi

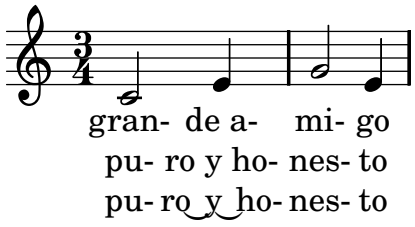
Manuel de notation : ⟨undefined⟩ [Conservation d'un contexte], page ⟨undefined⟩.

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[1].

```
\time 3/4
\relative c' { c2 e4 g2 e4 }
\addlyrics { gran- de_a- mi- go }
\addlyrics { pu- "ro y ho-" nes- to }
\addlyrics { pu- ro~y~ho- nes- to }
```



[1] Une liaison adaptée aux paroles correspond au caractère Unicode `U+203F`, et n'apparaîtra dans la partition que si le système dispose d'une police installée qui contient ce symbole (par exemple DejaVuLGC).

Voir aussi

Référence des propriétés internes : Section “LyricCombineMusic” dans *Référence des propriétés internes*.
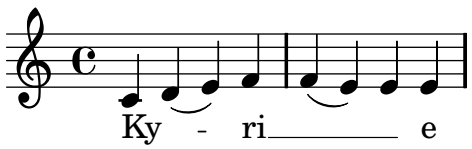
Plusieurs notes pour une même syllabe

Parfois, tout particulièrement dans la musique médiévale, plusieurs notes correspondent à une même syllabe. Ces vocalises sont appelées mélismes.

Il est possible d'indiquer tous les mélismes lors de la saisie des paroles. Il suffit pour cela d'utiliser le caractère _ pour chaque note du mélisme.

Vous pouvez aussi indiquer les mélismes par des lignes d'extension, indiquées par un double caractère souligné à la suite de la syllabe qui se prolonge. L'exemple qui suit utilise trois éléments distincts – notez bien les espaces qui les encadrent – : un double tiret pour séparer les syllabes d'un même mot, des soulignés pour ajouter des notes à la vocalise, et un double souligné pour la prolongation.
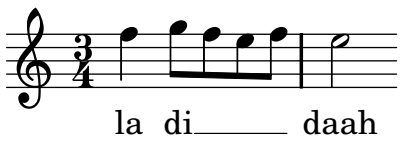
```
{ \set melismaBusyProperties = #'()
 c d( e) f f( e) e e  }
\addlyrics
 { Ky -- _ _ ri __ _ _ _  e }
```



En définissant, comme dans l'exemple ci-dessus, la propriété `melismaBusyProperties`, vous obtiendrez automatiquement toutes les liaisons de tenue et de phrasé requises.
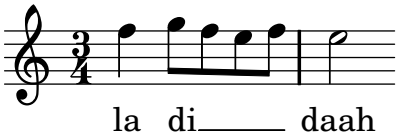
D'autre part, la commande `\lyricsto` arrive aussi à détecter automatiquement les mélismes : une seule syllabe sera placée sous un groupe de notes comprenant des liaisons de tenue ou de phrasé. Au cas où un mélisme doit s'appliquer à des notes non liées, il suffit d'adjoindre `\melisma` après la première note du groupe, et `\melismaEnd` après la dernière :

```
<<
  \new Voice = "lala" {
    \time 3/4
    f4 g8
    \melisma
    f e f
    \melismaEnd
    e2
  }
  \new Lyrics \lyricsto "lala" {
    la di __ daah
  }
>>
```



Enfin, lorsque la fonction de ligature automatique (cf. [Définition des règles de ligature automatique], page 71) est désactivée, le fait de connecter des notes manuellement créera un mélisme.

```
<<
  \new Voice = "lala" {
    \time 3/4
    \autoBeamOff
    f4 g8[ f e f]
    e2
  }
  \new Lyrics \lyricsto "lala" {
    la di __ daah
  }
>>
```



Vous trouverez un exemple complet de partition pour chœur dans la section Section “Ensemble vocal” dans *Manuel d’initiation*.

Commandes prédéfinies

`\melisma`, `\melismaEnd`.

Voir aussi

Morceaux choisis : Section “Vocal music” dans *Morceaux choisis*.

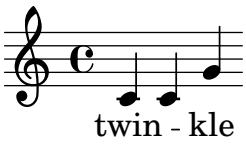
Problèmes connus et avertissements

Certains mélismes ne sont pas détectés automatiquement ; vous devrez alors prolonger vous-même les syllabes concernées.

Saut de notes

Pour que le texte aille moins vite que la musique, il suffit d’insérer des blancs, à l’aide de la commande `\skip` dans les paroles. Chaque `\skip` décalera la syllabe d’une note. Ce `\skip` doit comporter une durée quelle qu’elle soit – elle sera toujours ignorée en mode `Lyrics`.

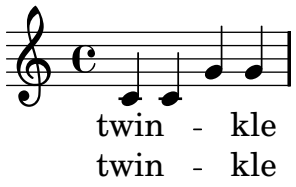
```
\relative c' { c c g' }
\addlyrics {
  twin -- \skip 4
  kle
}
```



Dans le cas où plusieurs notes consécutives seraient à « sauter », vous pouvez faire appel à la fonction `\repeat unfold` :

```
\relative c' { c c g' g }
\addlyrics {
  twin -- \skip 4 \skip 4
  kle
}
```

```
\addlyrics {
  twin -- \repeat unfold 2 { \skip 4 }
  kle
}
```



Traits d'union et de prolongation

Un mélisme est indiqué par une longue ligne horizontale basse centrée entre une syllabe et la suivante. Une telle ligne, que nous appellerons prolongateur ou extenseur, s'obtient en saisissant ' `__` ' (notez les espaces entourant le souligné double).

> **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 `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.

Voir aussi

Référence des propriétés internes : Section "LyricExtender" dans *Référence des propriétés internes*, Section "LyricHyphen" dans *Référence des propriétés internes*

Paroles et reprises

En cours de rédaction.

2.1.4 Situations particulères en matière de paroles
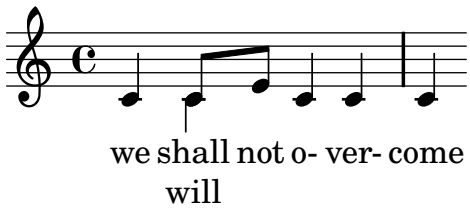
Une même mélodie peut se voir traitée différemment suivant les couplets. La commande `\lyricsto` permet de prendre en compte ces variantes.

Paroles alternatives

En donnant un nom à chaque voix et en leur attribuant spécifiquement des paroles, on peut créer des paroles alternatives – également qualifiées de *divisi*.

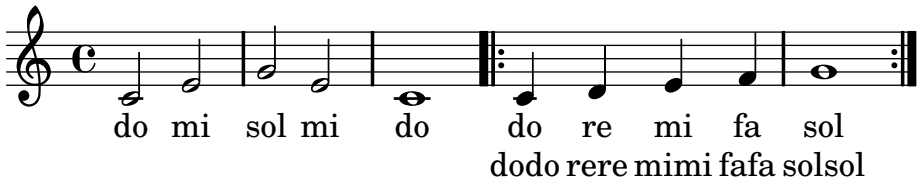
```
\score{ <<
  \new Voice = "melody" {
    \relative c' {
      c4
      <<
        { \voiceOne c8 e }
        \new Voice = "splitpart" { \voiceTwo c4 }
```
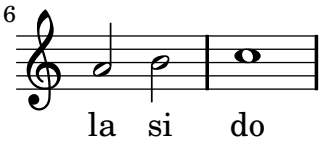
```
      >>
      \oneVoice c4 c | c
    }
  }
  \new Lyrics \lyricsto "melody" { we shall not o- ver- come }
  \new Lyrics \lyricsto "splitpart" { will }
>> }
```



Cette astuce permet de faire varier les paroles lorsqu'un fragment est repris.

```
\score{ <<
  \new Voice = "melody" \relative c' {
    c2 e | g e | c1 |
    \new Voice = "verse" \repeat volta 2 {c4 d e f | g1 | }
    a2 b | c1}
  \new Lyrics = "mainlyrics" \lyricsto melody \lyricmode {
    do mi sol mi do
    la si do }
  \context Lyrics = "mainlyrics" \lyricsto verse \lyricmode {
   do re mi fa sol }
  \new Lyrics = "repeatlyrics" \lyricsto verse \lyricmode {
   dodo rere mimi fafa solsol }
>>
}
```





Paroles indépendantes des notes

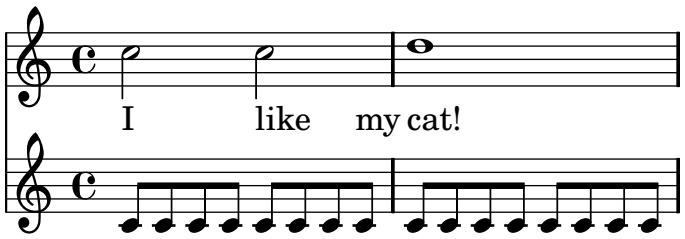
Dans certaines musiques vocales assez complexes, on peut avoir intérêt à imprimer les paroles indépendamment des notes. La mélodie sur laquelle s'aligne les paroles – marquée par le *tag* `lyricrhythm` dans l'exemple suivant – peut être insérée dans un contexte `Devnull`, ce qui indique à LilyPond de ne pas imprimer cette mélodie dans la partition. Seules subsistent alors de cette mélodie les valeurs rythmiques, sur lesquelles s'alignent les paroles.

```
voice = {
  c''2
  \tag #'music { c''2 }
```

```
  \tag #'lyricrhythm { c''4. c''8 }
  d''1
}

lyr = \lyricmode { I like my cat! }

<<
  \new Staff \keepWithTag #'music \voice
  \new Devnull="nowhere" \keepWithTag #'lyricrhythm \voice
  \new Lyrics \lyricsto "nowhere" \lyr
  \new Staff { c'8 c' c' c' c' c' c' c'
  c' c' c' c' c' c' c' c' }
>>
```
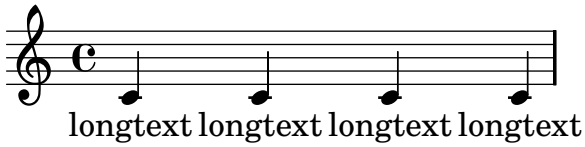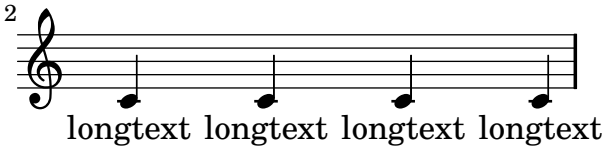


Cette méthode est inapropriée dès lors que la musique contenue dans le contexte `Devnull` comporte des mélismes. En effet, les mélismes sont définis par le contexte `Voice`. Connecter des paroles à un contexte `Devnull` rend les liens voix-syllabe inopérants, et par voie de conséquence fait disparaître les informations de mélisme.

Espacement des syllabes

La propriété `#'minimum-distance` de l'objet `LyricSpace` permet d'accroître l'espacement des paroles.

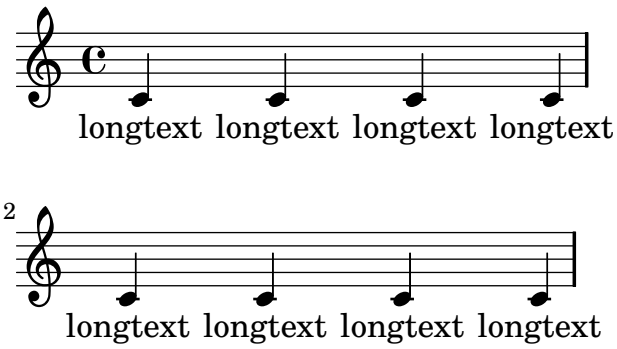
```
{
  c c c c
  \override Lyrics.LyricSpace #'minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```





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 c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace #'minimum-distance = #1.0
    }
  }
}
```



Morceaux choisis

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, LilyPond n'effectue pas ces calculs par défaut ; pour l'y obliger, vous devrez ajouter

```
\override Score.PaperColumn #'keep-inside-line = ##t
```

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"
      \override BarLine #'transparent = ##t
  }
}
```

Centrage des paroles entre les portées

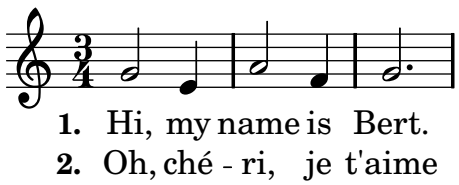
En cours de rédaction.

2.1.5 Couplets

Numérotation des couplets

On peut ajouter un numéro aux couplets en définissant la variable `stanza` :

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = #"2. "
  Oh, ché -- ri, je t'aime
}
```



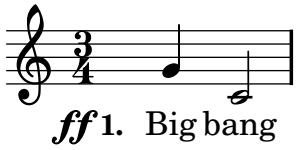
Ces numéros se placeront juste avant le début de la première syllabe.

Indication de nuance dans les couplets

Lorsque des couplets ont des nuances différentes, vous pouvez ajouter une nuance devant chacun d'eux. L'objet `StanzaNumber` contient tout ce qui se place au début du couplet. 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
>>
```

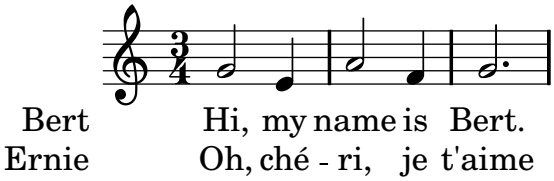


Indication du personnage et couplets

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 {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = #"Bert "
  Hi, my name is Bert.
```

```
} \addlyrics {
  \set vocalName = #"Ernie "
  Oh, ché -- ri, je t'aime
}
```



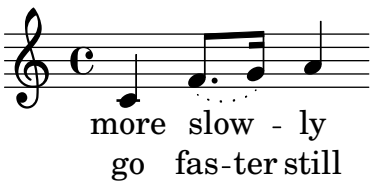
Rythme différent selon le couplet

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 c' \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
    \unset ignoreMelismata
    still
  }
>>
```

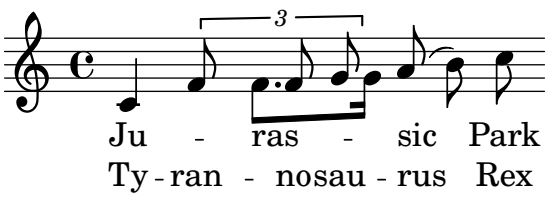


Problèmes connus et avertissements

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`.

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,



le texte du premier couplet s'aligne sur la mélodie nommée « lahlah »,

```
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
```

Le second couplet est tout d'abord rattaché au contexte `lahlah`, mais bascule sur une autre mélodie sur la syllabe « ran ». Pour cela, nous utilisons

```
\set associatedVoice = alternative
```

où `alternative` désigne le nom du contexte `Voice` qui contient le triolet.

Encore une fois, cette commande doit être spécifiée une syllabe en avance, c'est-à-dire ici avant la syllabe « Ty ».

```
\new Lyrics \lyricsto "lahlah" {
  \set associatedVoice = alternative % s'applique à "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = lahlah % s'applique à "rus"
  sau -- rus Rex
}
```

Comme vous pouvez le voir, nous revenons ensuite à l'alignement sur la mélodie d'origine, en assignant à la propriété `associatedVoice` la valeur `lahlah`.

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 en trouverez un exemple ci-dessous ; notez également les deux méthodes différentes employées pour indiquer les sauts de ligne, entre les couplets (*verses* en anglais) 2 et 3.
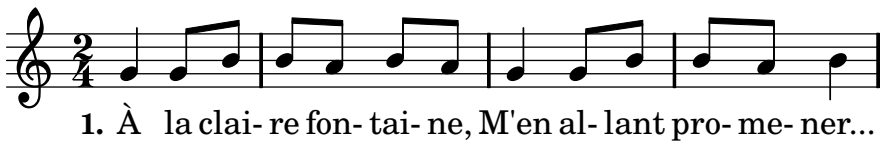
```
melody = \relative c'' {
  \time 2/4
  g4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

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{
  \line{ Verse 2. }
  \line{ Sous les feuilles d'un chêne }
  \line{ Je me suis fait sécher... }
  }
}
\markup{
  \wordwrap-string #"
  Verse 3.

  Chante, rossignol, chante,

  Toi qui as le coeur gai..."
}
```



1. À la clai- re fon- tai- ne, M'en al- lant pro- me- ner...

Verse 2.
Sous les feuilles d'un chêne
Je me suis fait sécher...

Verse 3.
Chante, rossignol, chante,
Toi qui as le coeur gai...

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.
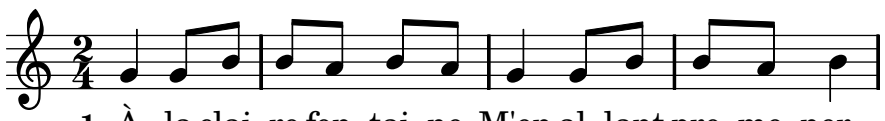
```
melody = \relative c'' {
  \time 2/4
  g4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

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 {
    \hspace #0.1 % décalage par rapport à la marge de gauche
        % peut être supprimé si l'espace sur la page est réduit
     \column {
      \line { \bold "2."
        \column {
          "Sous les feuilles d'un chêne"
          "Je me suis fait sécher..."
        }
      }
      \hspace #0.1 % ajout d'espace vertical entre les couplets
      \line { \bold "3."
        \column {
          "Chante, rossignol, chante,"
          "Toi qui as le coeur gai..."
        }
      }
    }
    \hspace #0.1  % ajout d'espace horizontal entre les colonnes
        % si elles sont toujours trop proches, ajouter d'autres paires de " "
        % jusqu'à ce que le résultat soit acceptable.
     \column {
      \line { \bold "4."
        \column {
          "J'ai perdu mon ami"
          "Sans l'avoir mérité..."
        }
      }
      \hspace #0.1 % ajout d'espace vertical entre les couplets
      \line { \bold "5."
        \column {
          "Je voudrais que la rose"
          "Fût encore au rosier..."
        }
      }
    }
  \hspace #0.1 % décalage par rapport à la marge de droite
      % peut être supprimé si l'espace sur la page est réduit
  }
}
```



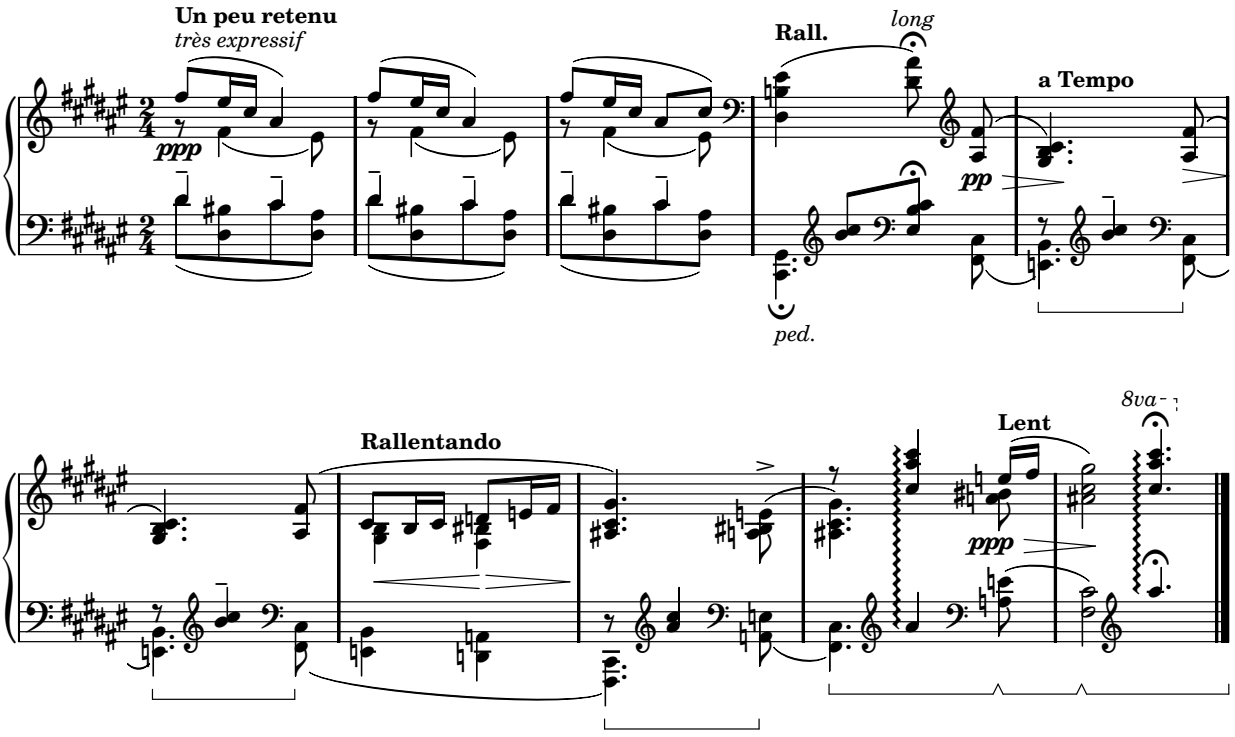
1. À la clai- re fon- tai- ne, M'en al- lant pro- me- ner...

2. Sous les feuilles d'un chêne
Je me suis fait sécher...
3. Chante, rossignol, chante,
Toi qui as le coeur 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*.

2.2 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 140.
- On peut écrire la musique pour claviers de façon parallèle, comme l'explique [Saisie de musique en parallèle], page 147.
- Les indications de doigté sont abrodées dans [Doigtés], page 179.
- Les indications en matière de pédalier d'orgue sont traitées comme des articulations. Voir à ce sujet Section A.10 [List of articulations], page 555.
- Pour générer des repères verticaux, voir [Quadrillage temporel], page 186.
- 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 44.
- Le traitement des arpèges couvrant plusieurs voix ou portées est abordé au chapitre [Arpèges], page 115.
- Une description des indications de tremolo est disponible au chapitre [Répétitions en trémolo], page 131.
- 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 "Exemple concret" dans *Manuel d'initiation*, Section "Autres utilisations des retouches" dans *Manuel d'initiation*.

Manuel de notation : [Regroupement de portées], page 151, [Noms d'instrument], page 167, [Résolution des collisions], page 140, [Saisie de musique en parallèle], page 147, [Doigtés], page 179, Section A.10 [List of articulations], page 555, [Quadrillage temporel], page 186, [Liaisons de prolongation], page 44, [Arpèges], page 115, [Répétitions en trémolo], page 131.

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

Exemples de code : Section "Keyboards" dans *Morceaux choisis*.

Problèmes connus et avertissements

Les nuances ne sont pas automatiquement centrées verticalement, mais il existe des astuces. Voir à ce sujet le modèle « alignement des nuances au piano » au chapitre Section "Modèles pour claviers" dans *Manuel d'initiation* ; une autre solution consiste à augmenter le décalage des nuances par rapport à la portée (`staff-pagging`) comme l'indique Section "Déplacement d'objets" dans *Manuel d'initiation*.

Changement de portée manuel

Il est possible de passer d'une portée à l'autre de façon manuelle, au moyen de la commande

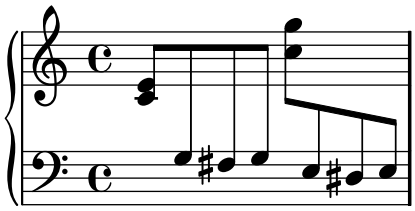
```
\change Staff = nomDeLaPortee
```

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 ⟨undefined⟩ [Conservation d'un contexte], page ⟨undefined⟩ à ce propos.

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.

Voir aussi

Manuel d'initiation : Section "Correction des collisions d'objets" dans *Manuel d'initiation*.

Manuel de notation : [Hampes], page 184, [Barres de ligature automatiques], page 69, ⟨undefined⟩ [Conservation d'un contexte], page ⟨undefined⟩.

Exemples de code : Section "Keyboards" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Beam" dans *Référence des propriétés internes*, Section "ContextChange" dans *Référence des propriétés internes*.

Changement de portée automatique
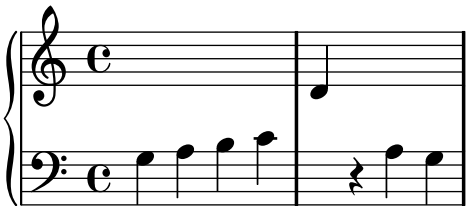
Les voix peuvent passer automatiquement d'une portée à l'autre, au moyen de la syntaxe suivante :

```
\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 clé 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
  }
```

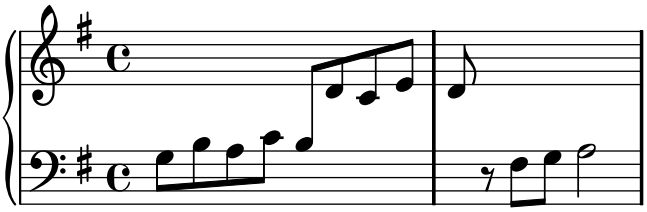
```
}
```



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 c' {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



Voir aussi

Dans ce même manuel : [Changement de portée manuel], page 234.

Exemples de code : Section "Keyboards" 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.

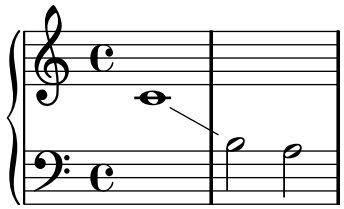
Un accord ne peut se répartir sur plusieurs portées ; sa portée d'affectation sera déterminée par la première hauteur mentionnée dans la construction de cet accord.

`\autochange` ne peut intervenir à l'intérieur d'une commande `\times`.

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
    c1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



Commandes prédéfinies

`\showStaffSwitch`, `\hideStaffSwitch`.

Voir aussi

Extraits de code : Section "Keyboards" dans *Morceaux choisis*.

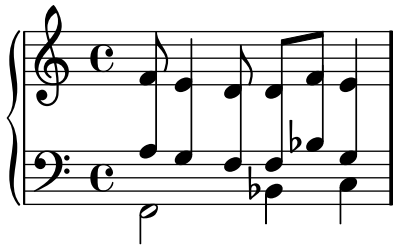
Référence des propriétés internes : Section "Note_head_line_engraver" dans *Référence des propriétés internes*, Section "VoiceFollower" dans *Référence des propriétés internes*.

Hampes et changements de portée

Pour écrire des accords qui enjambent deux portées, on allonge la hampe de l'accord de l'une des deux portées de façon à ce qu'elle rejoigne celle de l'autre portée.

```
\new PianoStaff <<
  \new Staff {
    \relative c' {
      f8 e4 d8 d f e4
    }
  }
  \new Staff {
    \relative c' {
      << {
        \clef bass
        % stems may overlap the other staff
        \override Stem #'cross-staff = ##t
        % extend the stems to reach the other staff
        \override Stem #'length = #12
        % do not print extra flags
        \override Stem #'flag-style = #'no-flag
```

```
        % prevent beaming as needed
        a8 g4 f8 f bes\noBeam g4
      }
      \\
      {
        f,2 bes4 c
      } >>
    }
  }
>>
```



Morceaux choisis

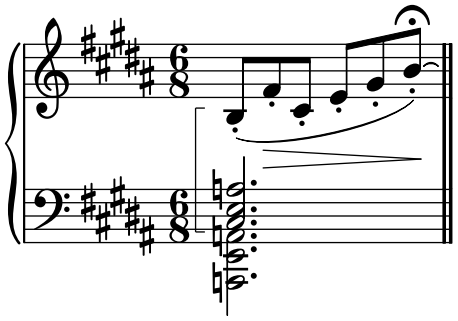
Indicating cross-staff chords with arpeggio bracket

An arpeggio bracket can indicate that notes on two different staves are to be played with the same hand. In order to do this, the `PianoStaff` must be set to accept cross-staff arpeggios and the arpeggios must be set to the bracket shape in the `PianoStaff` context.

(Debussy, Les collines d'Anacapri, m. 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

Extraits de code : Section “Keyboards” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Stem” dans *Référence des propriétés internes*.
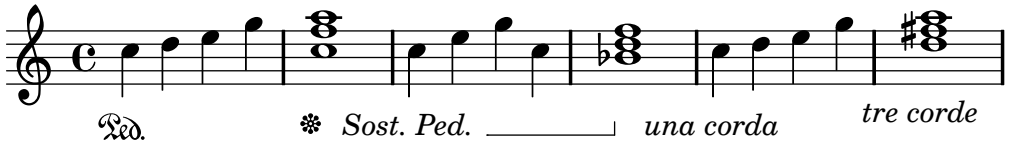
2.2.2 Piano

Ce chapitre traîte des aspects de la notation directement liés au piano.

Pédales de piano

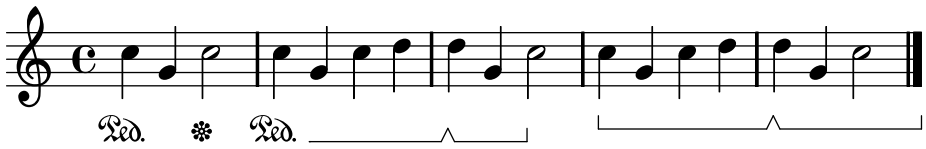
Le piano possède deux pédales, parfois trois, permettant de modifier l’émission du son : une pédale de *tenue* (*sustain*), une pédale de *sourdine* (*una corda* ou *U.C.*) et une pédale *tonale* (*sostenuto* ou *sos.*). La pédale *sustain* se rencontre aussi sur les vibraphones et celestas.

```
c4\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.

```
c4\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 enclanchée jusqu'à la barre finale s'obtient tout simplement en omettant la commande de relacher.

Voir aussi

Manuel de notation : [Liaisons de prolongation], page 44.

Extraits de code : Section "Keyboards" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SustainPedal" dans *Référence des propriétés internes*, Section "SustainPedalLineSpanner" dans *Référence des propriétés internes*, Section "SustainEvent" dans *Référence des propriétés internes*, Section "SostenutoPedal" dans *Référence des propriétés internes*, Section "SostenutoPedalLineSpanner" dans *Référence des propriétés internes*, Section "SostenutoEvent" dans *Référence des propriétés internes*, Section "UnaCordaPedal" dans *Référence des propriétés internes*, Section "UnaCordaPedalLineSpanner" dans *Référence des propriétés internes*, Section "UnaCordaEvent" dans *Référence des propriétés internes*, Section "PianoPedalBracket" dans *Référence des propriétés internes*, Section "Piano_pedal_engraver" 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.

Morceaux choisis

Accordion-discant symbols

Accordion discant-specific symbols are added using `\markup`. The vertical placement of the symbols can be tweaked by changing the `\raise` arguments.

```
discant = \markup {
  \musicglyph #"accordion.discant"
}
dot = \markup {
  \musicglyph #"accordion.dot"
}

\layout { ragged-right = ##t }

% 16 voets register
accBasson = ^\markup {
  \combine
  \discant
  \raise #0.5 \dot
}

% een korig 8 en 16 voets register
accBandon = ^\markup {
```

```
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \raise #1.5 \dot
}

accVCello = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \translate #'(1 . 0) \raise #1.5 \dot
}

% 4-8-16 voets register
accHarmon = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \raise #2.5 \dot
}

accTrombon = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \combine
          \translate #'(1 . 0) \raise #1.5 \dot
          \translate #'(-1 . 0) \raise #1.5 \dot
}

% eenkorig 4 en 16 voets register
accOrgan = ^\markup {
  \combine
    \discant
    \combine
      \raise #0.5 \dot
      \raise #2.5 \dot
}

accMaster = ^\markup {
  \combine
    \discant
```

```
    \combine
      \raise #0.5 \dot
      \combine
        \raise #1.5 \dot
        \combine
          \translate #'(1 . 0) \raise #1.5 \dot
          \combine
            \translate #'(-1 . 0) \raise #1.5 \dot
            \raise #2.5 \dot
}

accAccord = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \combine
          \translate #'(-1 . 0) \raise #1.5 \dot
          \raise #2.5 \dot
}

accMusette = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \translate #'(-1 . 0) \raise #1.5 \dot
}

accCeleste = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \translate #'(-1 . 0) \raise #1.5 \dot
}

accOboe = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \raise #2.5 \dot
}

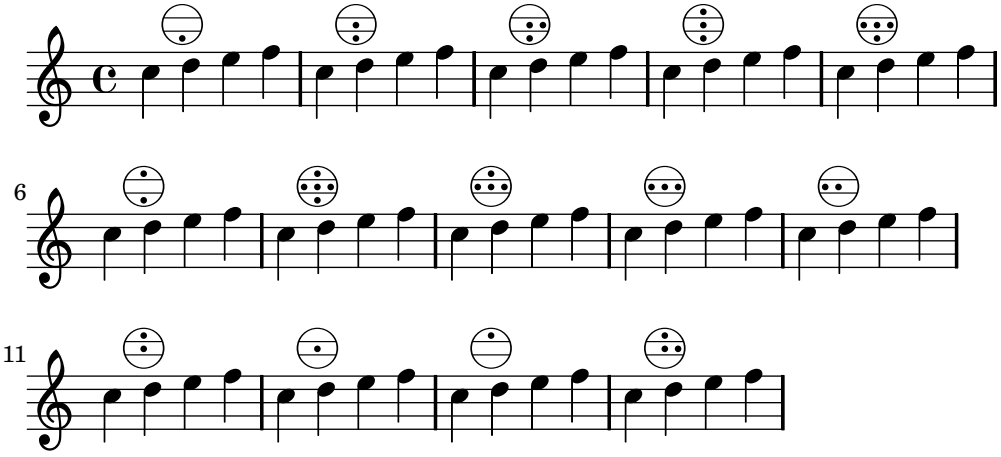
accClarin = ^\markup {
  \combine
    \discant
```

```
    \raise #1.5 \dot
}

accPiccolo = ^\markup {
    \combine
       \discant
       \raise #2.5 \dot
}

accViolin = ^\markup {
  \combine
    \discant
    \combine
      \raise #1.5 \dot
      \combine
        \translate #'(1 . 0) \raise #1.5 \dot
        \raise #2.5 \dot
}

\relative c'' {
  c4 d\accBasson e f
  c4 d\accBandon e f
  c4 d\accVCello e f
  c4 d\accHarmon e f
  c4 d\accTrombon e f
  \break
  c4 d\accOrgan e f
  c4 d\accMaster e f
  c4 d\accAccord e f
  c4 d\accMusette e f
  c4 d\accCeleste e f
  \break
  c4 d\accOboe e f
  c4 d\accClarin e f
  c4 d\accPiccolo e f
  c4 d\accViolin e f
}
```



Voir aussi

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

2.2.4 Harpe

This section discusses notation issues that are unique to the harp.

Généralités sur la harpe

Certaines caractéritiques de la musique pour harpes sont abordées dans d’autres chapitres, telles que :

- Les glissandos, l’une des techniques spécifique à la harpe, [Glissando], page 114.
- Le *bisbigliando*, qui s’écrit comme un trémolo, [Répétitions en trémolo], page 131.
- Les harmoniques naturelles sont présentées dans [Harmoniques], page 246.
- For directional arpeggios and non-arpeggios, see [Arpèges], page 115.
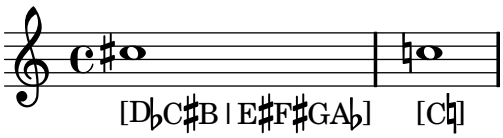
Voir aussi

Manuel de notation : [Répétitions en trémolo], page 131, [Glissando], page 114, [Arpèges], page 115, [Harmoniques], page 246.

Pédales de harpe

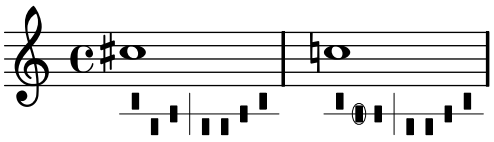
Les harpes comportent sept cordes par octave qui peuvent sonner naturel, dièse ou bémol. Si chacune des cordes de la harpe celtique (*lever harp*) est accordée individuellement, celles d’une harpe à pédalier ayant la même note de base sont contrôlées par une seule pédale. De gauche à droite, elles correspondent 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
cis1_\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 diagrammes :

```
\textLengthOn
cis1_\markup { \harp-pedal #"^v-|vv-^" }
c!1_\markup { \harp-pedal #"^o--|vv-^" }
```
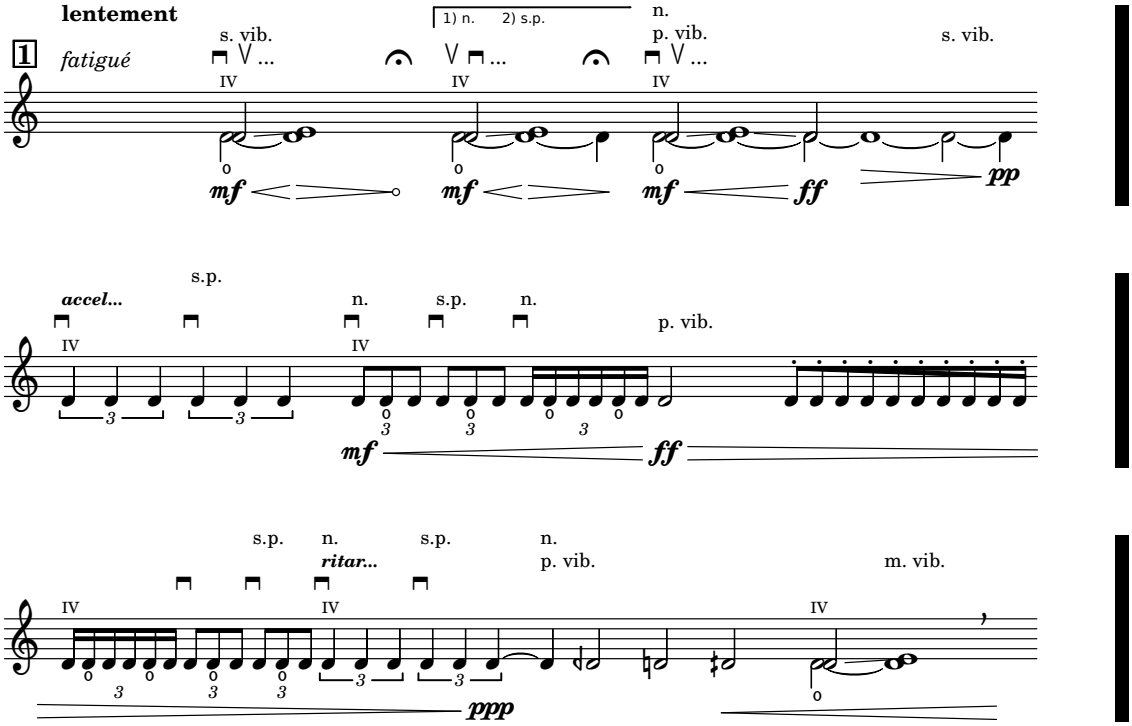


bémol si la pédale est relâchée (ou en haut), bécarre si elle est bloquée sur le cran du milieu, et dièse si elle est tout à fait enfoncée. La commande `\harp-pedal` prend en argument une chaîne de caractères, où `^` indique que la pédale est relâchée ou en haut (bémol), `-` qu’elle est bloquée sur le 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 189, Section A.8.5 [Instrument Specific Markup], page 546.

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 189.
- Les indications de doigtés, incluant les indications du pouce, sont décrites dans [Doigtés], page 179.
- Les doubles notes sont généralement indiquées en écrivant un accord, voir [Notes en accords], page 133. Les précisions pour jouer des accords peuvent être ajoutées, comme l'indique [Arpèges], page 115.
- Un modèle pour quatuor à cordes est disponible au chapitre Section "Quatuor à cordes" dans *Manuel d'initiation*. D'autres informations se trouvent dans les exemples de code.

Voir aussi

Manuel d'initiation : Section "Quatuor à cordes" dans *Manuel d'initiation*.

Manuel de notation : [Commentaires textuels], page 189, [Doigtés], page 179, [Notes en accords], page 133, [Arpèges], page 115.
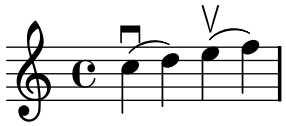
Morceaux choisis : Section "Unfretted strings" dans *Morceaux choisis*.

Indications d'archet

Les indications d'archet se créent comme des articulations, elles sont décrites dans [Articulations et ornements], page 99.
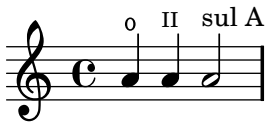
Les indications d'archet, poussé (`\upbow`) et tiré (`\downbow`), peuvent se combiner à des liaisons comme ici :

```
c4(\downbow d) e(\upbow f)
```



ou dans l'exemple suivant qui montre trois manières d'indiquer la corde de La, à vide, sur un violon :

```
a4 \open
a^\markup { \teeny "II" }
a2^\markup { \small "sul A" }
```



Commandes prédéfinies

`\downbow`, `\upbow`, `\open`.

Voir aussi
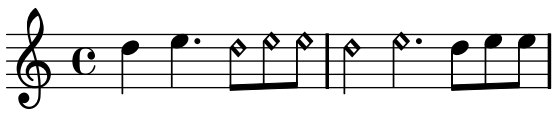
Manuel de notation : [Articulations et ornements], page 99, [Liaisons d'articulation], page 108.

Harmoniques

Harmoniques naturels
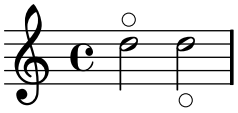
Les harmoniques naturels sont indiqués de différentes manières : une tête de note en forme de losange signifie généralement que vous devez effleurer la corde à l'endroit où vous l'auriez pincée si la note avait été normale.

```
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 doît être jouée en harmonique :

```
d2^\flageolet d_\flageolet
```
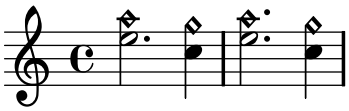


Un plus petit cercle peut être créé, comme indiqué dans les exemples de code contenus dans [Références en matière de cordes non frettées], page 245.

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
```



Note : La commande `\harmonic` **doit** être placée dans une construction de type accord même s'il n'y a qu'une seule note. On utiliserait plutôt la commande `\harmonicsOn` en pareil cas.

Voir aussi

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

Manuel de notation : [Têtes de note spécifiques], page 29, [Références en matière de cordes non frettées], page 245.

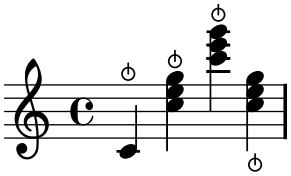
Snap (Bartók) pizzicato

Morceaux choisis

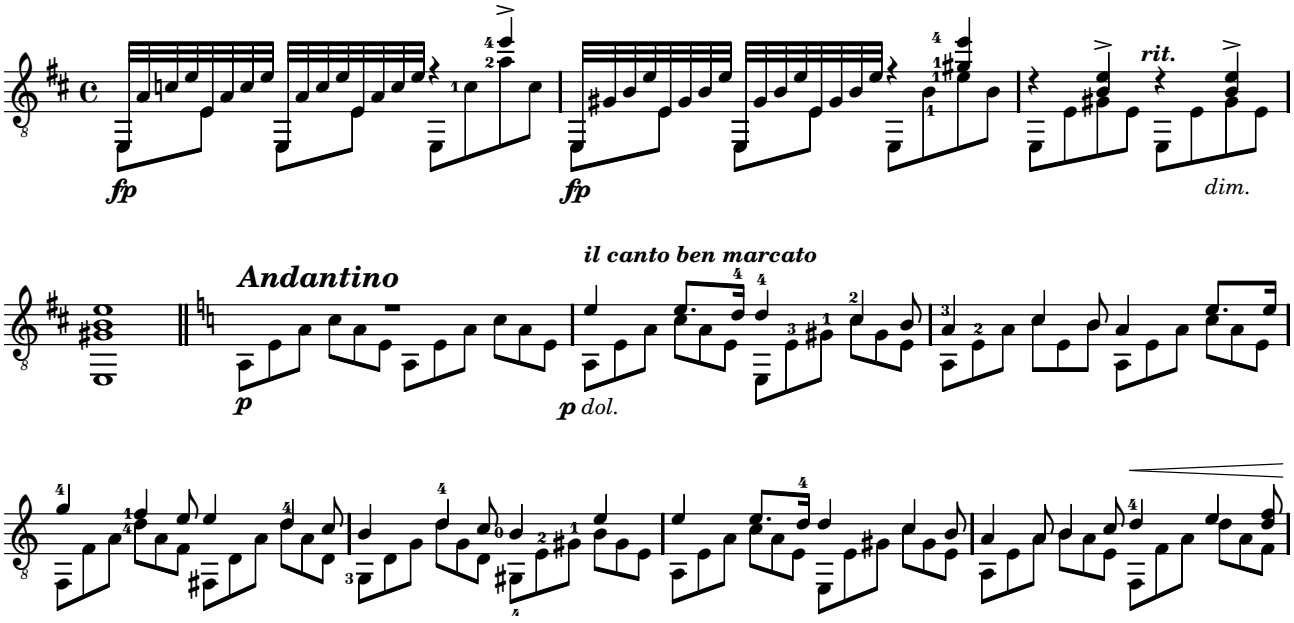
Snap-pizzicato ("Bartok pizzicato")

A snap-pizzicato (also known as « Bartok pizzicato ») is a « strong pizzicato where the string is plucked vertically by snapping and rebounds off the fingerboard of the instrument » (Wikipedia). It is denoted by a circle with a vertical line going from the center upwards outside the circle.

```
\relative c' {
  c4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



2.4 Fretted string instruments



This section discusses several aspects of music notation that are unique to fretted string instruments.

2.4.1 Common notation for fretted strings

This section discusses common notation that is unique to fretted string instruments.

References for fretted strings

Music for fretted string instruments is normally notated on a single staff, either in traditional music notation or in tablature. Sometimes the two types are combined, and it is especially common in popular music to use chord diagrams above a staff of traditional notation. The guitar and the banjo are transposing instruments, sounding an octave lower than written. Scores for these instruments should use the `"treble_8"` clef (or `\transposition c` to get correct MIDI output). Some other elements pertinent to fretted string instruments are covered elsewhere:

- Fingerings are indicated as shown in ⟨undefined⟩ [Fingering instructions], page ⟨undefined⟩.
- Instructions for *Laissez vibrer* ties as well as ties on arpeggios and tremolos can be found in ⟨undefined⟩ [Ties], page ⟨undefined⟩.
- Instructions for handling multiple voices can be found in ⟨undefined⟩ [Collision resolution], page ⟨undefined⟩.
- Instructions for indicating harmonics can be found in ⟨undefined⟩ [Harmonics], page ⟨undefined⟩.

Voir aussi

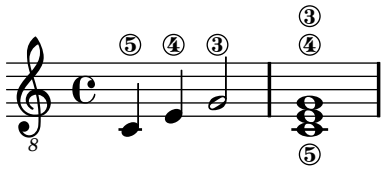
Notation Reference: ⟨undefined⟩ [Fingering instructions], page ⟨undefined⟩, ⟨undefined⟩ [Ties], page ⟨undefined⟩, ⟨undefined⟩ [Collision resolution], page ⟨undefined⟩, ⟨undefined⟩ [Instrument names], page ⟨undefined⟩, ⟨undefined⟩ [Writing music in parallel], page ⟨undefined⟩, ⟨undefined⟩ [Arpeggio], page ⟨undefined⟩, Section A.10 [List of articulations], page 555, ⟨undefined⟩ [Clef], page ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], page ⟨undefined⟩.

String number indications

The string on which a note should be played may be indicated by appending `\number` to a note inside a chord construct `<>`.

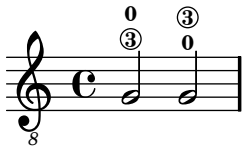
> **Note :** String numbers **must** be defined inside a chord construct even if there is only a single note.

```
\clef "treble_8"
<c\5>4 <e\4> <g\3>2
<c,\5 e\4 g\3>1
```



When fingerings and string indications are used together, their placement is controlled by the order in which the two items appear in the code:

```
\clef "treble_8"
<g\3-0>2
<g-0\3>
```

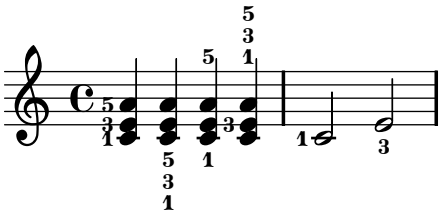


Morceaux choisis

Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, you must use a chord construct `<>` even if it is a single note.

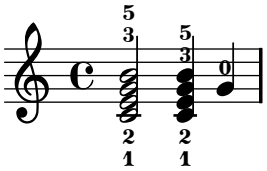
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff. However, this behavior can be canceled. Note: you must use a chord construct `<>`, even if it is only a single note.

```
\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>
}
```



Voir aussi

Notation Reference: ⟨undefined⟩ [Fingering instructions], page ⟨undefined⟩.

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

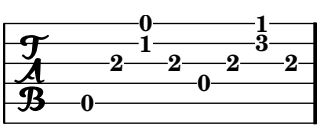
Internals Reference: Section “StringNumber” dans *Référence des propriétés internes*, Section “Fingering” dans *Référence des propriétés internes*.

Default tablatures

Music for plucked string instruments is frequently notated using a finger/touch notation or tablature. In contrast to traditional notation pitches are not denoted with note heads, but by numbers (or letterlike symbols in historical intavolatura). The numbers indicate on which string and fret a note must be played. The numbers are printed on top of each other if they are to be played simultaneously.

By default, string 1 is the highest, and the tuning defaults to the standard guitar tuning (with 6 strings). The notes are printed as tablature, by using `TabStaff` and `TabVoice` contexts. A calligraphic tablature clef is added automatically.

```
\new TabStaff \relative c' {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```

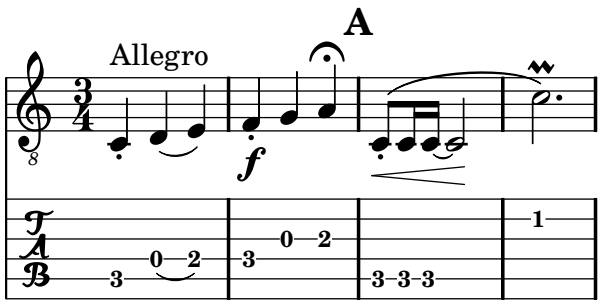


Default tablatures do not contain any symbols for tone duration nor any other musical symbols such as e.g. expressive marks.

```
symbols = {
  \time 3/4
```

```
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

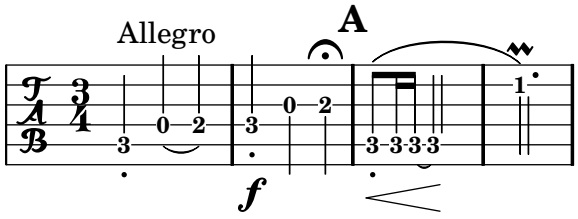
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff { \symbols }
  >>
}
```



If all musical symbols used in traditional notation should also show up in tablature one has to apply the command `\tabFullNotation` in a `TabStaff`-context. Please bear in mind that half notes are double-stemmed in tablature in order to distinguish them from quarter notes.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ c2\!
  c'2.\prall\)
}

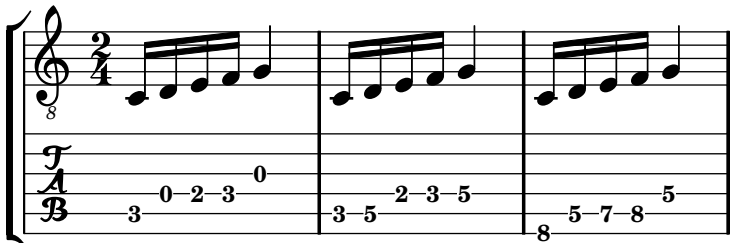
\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```



By default pitches are assigned to the lowest playing position on the fret-board (first position). Open strings are automatically preferred. If you would like a certain pitch to be played on a

specific string you can add a string number indication to the pitch name. If you define pitch names and string numbers without a chord construct (<>) the string number indications do not appear in traditional notation. It is much more comfortable to define the playing position by using the value of `minimumFret`. The default value for minimumFret is 0.

```
\new StaffGroup <<
   \new Staff \relative c {
     \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 c {
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     \set TabStaff.minimumFret = #5
     c,16 d e f g4
   }
>>
```

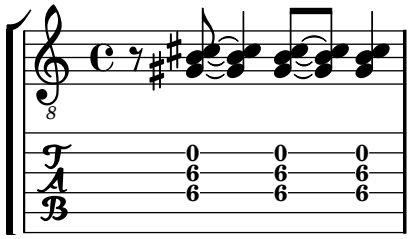


Chord constructs can be repeated by the chord repetition symbol `q`. To use this feature in combination with tablature, `\tabChordRepetition` is provided. It preserves the string information explicitly given within chord constructs so repeated chords get identical tablature representations.

```
\tabChordRepetition

guitar = \relative c' {
  r8 <gis\4 cis\3 b\2>~ q4 q8~ q q4
}

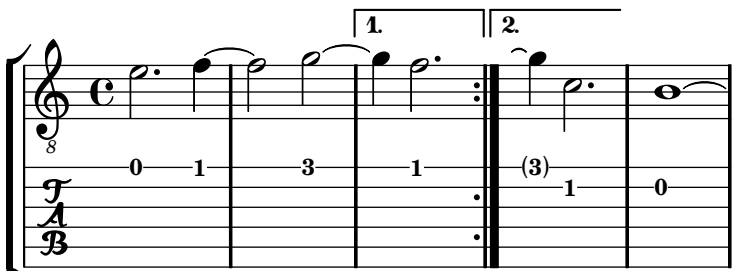
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \override Voice.StringNumber #'transparent = ##t
    \guitar
  }
  \new TabStaff {
    \guitar
  }
>>
```

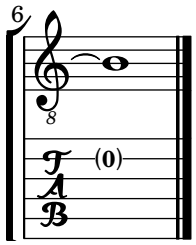


Ties over a line break are parenthesized by default. The same holds for the second alternative of a repeat.

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~
  }
  \alternative {
     { g4 f2. }
     { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \context Staff {
        \clef "treble_8"
        \ties
      }
      \context TabStaff {
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```

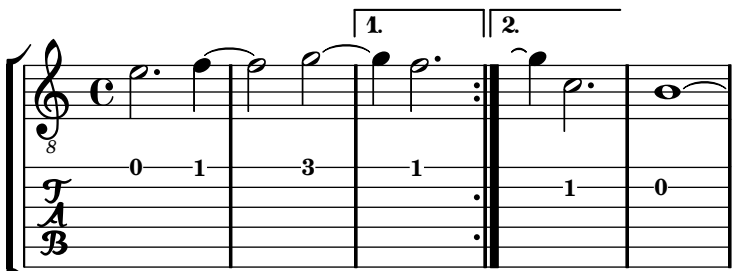


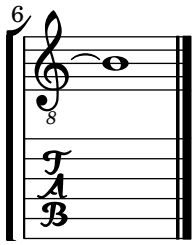


The command `\hideSplitTiedTabNotes` cancels the behavior of engraving fret numbers in parentheses:

```
ties = \relative c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

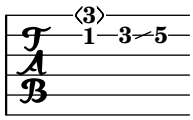
\score {
  <<
    \new StaffGroup  <<
      \context Staff {
        \clef "treble_8"
        \ties
      }
      \context TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```





Harmonic indications and slides can be added to tablature notation.

```
\new TabStaff {
  \new TabVoice {
    <c g'\harmonic>4 d\2\glissando e\2
  }
}
```

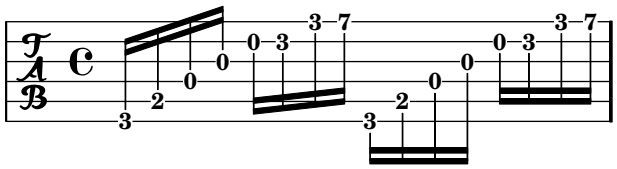


Morceaux choisis

Stem and beam behavior in tablature

The direction of stems is controlled the same way in tablature as in traditional notation. Beams can be made horizontal, as shown in this example.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam #'damping = #+inf.0
    g,,16 b d g b d g b
  }
}
```



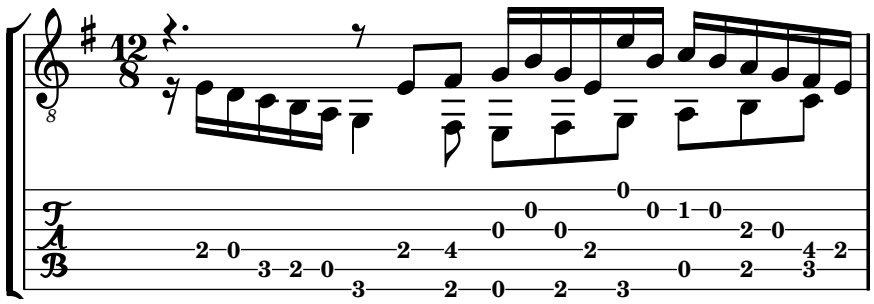
Polyphony in tablature

Polyphony is created the same way in a `TabStaff` as in a regular staff.

```
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"
        \context Voice = "upper" \upper
        \context Voice = "lower" \lower
      >>
      \new TabStaff = "guitar tab" <<
        \context TabVoice = "upper" \upper
        \context TabVoice = "lower" \lower
      >>
    >>
  >>
}
```



Voir aussi

Notation Reference: ⟨undefined⟩ [Stems], page ⟨undefined⟩.

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

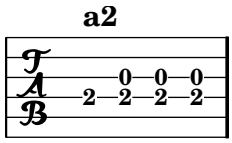
Internals Reference: Section “TabNoteHead” dans *Référence des propriétés internes*, Section “TabStaff” dans *Référence des propriétés internes*, Section “TabVoice” dans *Référence des propriétés internes*, Section “Beam” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Chords are not handled in a special way, and hence the automatic string selector may easily select the same string for two notes in a chord.

In order to handle `\partcombine`, a `TabStaff` must use specially-created voices:

```
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 }
  >>
>>
```



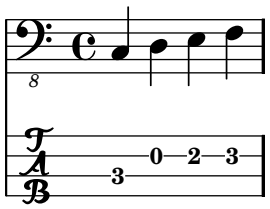
Guitar special effects are limited to harmonics and slides.

Custom tablatures

LilyPond tabulature automatically calculates the fret for a note based on the string to which the note is assigned. In order to do this, the tuning of the strings must be specified. The tuning of the strings is given in the `stringTunings` property.

LilyPond comes with predefined string tunings for banjo, mandolin, guitar, bass guitar and ukulele. LilyPond automatically sets the correct transposition for predefined tunings. The following example is for bass guitar, which sounds an octave lower than written.

```
<<
  \new Staff {
    \clef "bass_8"
    \relative c, {
      c4 d e f
    }
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #bass-tuning
    \relative c, {
      c4 d e f
    }
  }
>>
```



The default string tuning is `guitar-tuning`, which is the standard EADGBE tuning. Some other predefined tunings are `guitar-open-g-tuning`, `mandolin-tuning` and `banjo-open-g-tuning`. The predefined string tunings are found in `scm/tablature.scm`.

A string tuning is a Scheme list of string pitches, one for each string, ordered by string number from 1 to N, where string 1 is at the top of the tablature staff and string N is at the bottom. This ordinarily results in ordering from highest pitch to lowest pitch, but some instruments (e.g. ukulele) do not have strings ordered by pitch.

A string pitch in a string tuning list is the pitch difference of the open string from middle C measured in semitones. The string pitch must be an integer. LilyPond calculates the actual pitch of the string by adding the string tuning pitch to the actual pitch for middle C.

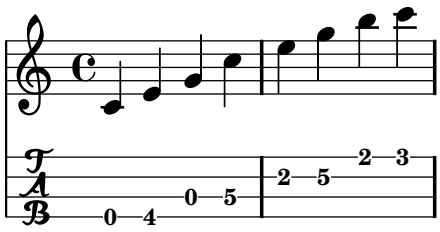
LilyPond automatically calculates the number of strings in the `TabStaff` as the number of elements in `stringTunings`.

Any desired string tuning can be created. For example, we can define a string tuning for a four-string instrument with pitches of `a''`, `d''`, `g'`, and `c'`:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
```

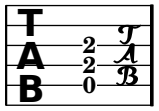
```
}

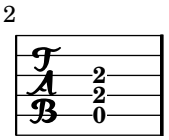
<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #'(21 14 7 0)
    \mynotes
  }
>>
```



A modern tab clef can also be used.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```





The modern tab clef supports tablatures from 4 to 7 strings.

Voir aussi

Installed Files: ‘`scm/tablature.scm`’.

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

Internals Reference: Section “Tab_note_heads_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

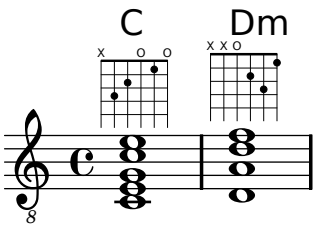
Automatic tablature calculations do not work properly in most cases for instruments where string pitches do not vary monotonically with string number, such as ukuleles.

Fret diagram markups

Fret diagrams can be added to music as a markup to the desired note. The markup contains information about the desired fret diagram. There are three different fret-diagram markup interfaces: standard, terse, and verbose. The three interfaces produce equivalent markups, but have varying amounts of information in the markup string. Details about the markup interfaces are found at Section A.8 [Text markup commands], page 515.

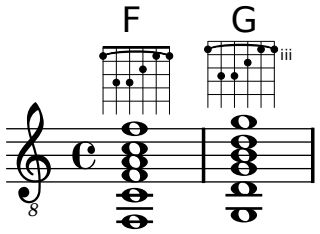
The standard fret diagram markup string indicates the string number and the fret number for each dot to be placed on the string. In addition, open and unplayed (muted) strings can be indicated.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context 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;"
    }
  }
>>
```



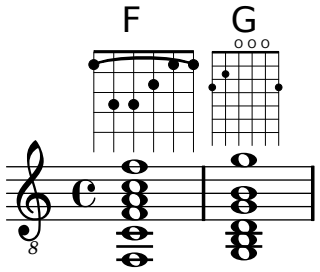
Barre indications can be added to the diagram from the fret-diagram markup string.

```
<<
  \context ChordNames {
     \chordmode {
       f1 g
     }
  }
  \context 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;"
    }
  }
>>
```



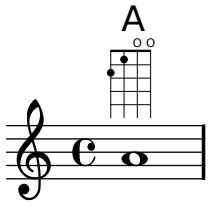
The size of the fret diagram, and the number of frets in the diagram can be changed in the fret-diagram markup string.

```
<<
  \context ChordNames {
     \chordmode {
       f1 g
     }
  }
  \context 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;"
    }
  }
>>
```



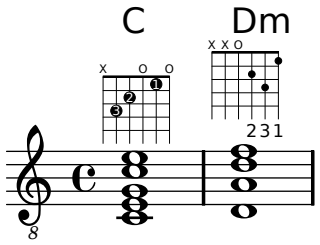
The number of strings in a fret diagram can be changed to accommodate different instruments such as banjos and ukuleles with the fret-diagram markup string.

```
<<
  \context ChordNames {
    \chordmode {
      a1
    }
  }
  \context Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram #"w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



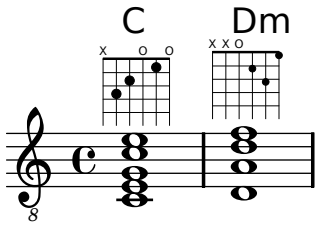
Fingering indications can be added, and the location of fingering labels can be controlled by the fret-diagram markup string.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context 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;"
    }
  }
>>
```



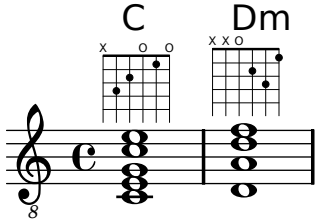
Dot radius and dot position can be controlled with the fret-diagram markup string.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context 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;"
    }
  }
>>
```



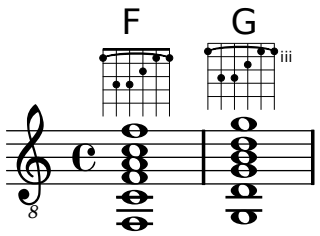
The fret-diagram-terse markup string omits string numbers; the string number is implied by the presence of semicolons. There is one semicolon for each string in the diagram. The first semicolon corresponds to the highest string number and the last semicolon corresponds to the first string. Mute strings, open strings, and fret numbers can be indicated.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context 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;"
    }
  }
>>
```



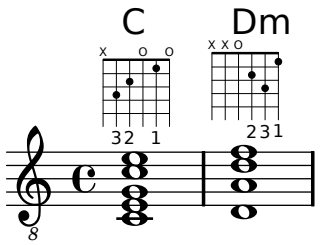
Barre indicators can be included in the fret-diagram-terse markup string.

```
<<
  \context ChordNames {
    \chordmode {
      f1 g
    }
  }
  \context 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-);"
    }
  }
>>
```



Fingering indications can be included in the fret-diagram-terse markup string.

```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context 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;"
    }
  }
>>
```

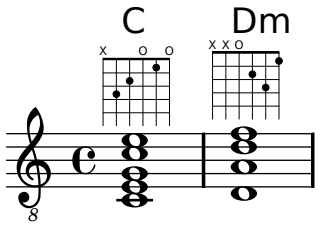


Other fret diagram properties must be adjusted using `\override` when using the fret-diagram-terse markup.

The fret-diagram-verbose markup string is in the format of a Scheme list. Each element of the list indicates an item to be placed on the fret diagram.

```
<<
    \context ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \context 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)
```

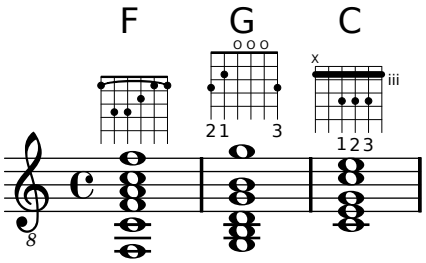
```
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



Fingering indications and barres can be included in a fret-diagram-verbose markup string. Unique to the fret-diagram-verbose interface is a capo indication that can be placed on the fret diagram. The capo indication is a thick bar that covers all strings. The fret with the capo will be the lowest fret in the fret diagram.

```
<<
    \context ChordNames {
      \chordmode {
        f1 g c
      }
    }
  \context 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)
        )
      }
    }
  >>
```



All other fret diagram properties must be adjusted using `\override` when using the fret-diagram-verbose markup.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Section "fret-diagram-interface" dans *Référence des propriétés internes*. For a fret diagram markup, the interface properties belong to `Voice.TextScript`.

Morceaux choisis

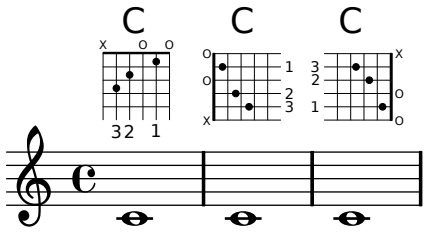
Changing fret orientations

Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

```
\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'
  }
>>
```



Customizing markup fret diagrams

Fret diagram properties can be set through `'fret-diagram-details`. For markup fret diagrams, overrides can be applied to the `Voice.TextScript` object or directly to the 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

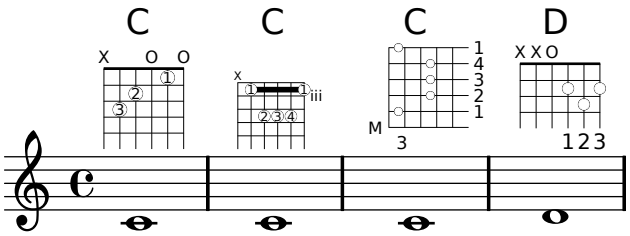
    %% C major for guitar, no barre, using defaults
       % terse style
    c'1^\markup { \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;" }

    %% 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

Notation Reference: Section A.8 [Text markup commands], page 515.

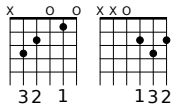
Snippets: Section “Fretted strings” dans *Morceaux choisis*.

Internals Reference: Section “fret-diagram-interface” dans *Référence des propriétés internes*.

Predefined fret diagrams

Fret diagrams can be displayed using the `FretBoards` context. By default, the `FretBoards` context will display fret diagrams that are stored in a lookup table:

```
\include "predefined-guitar-fretboards.ly"
\context FretBoards {
  \chordmode {
    c1 d
  }
}
```



The default predefined fret diagrams are contained in the file `predefined-guitar-fretboards.ly`. Fret diagrams are stored based on the pitches of a chord and the value of `stringTunings` that is currently in use. `predefined-guitar-fretboards.ly` contains predefined fret diagrams only for `guitar-tuning`. Predefined fret diagrams can be added for other instruments or other tunings by following the examples found in `predefined-guitar-fretboards.ly`.

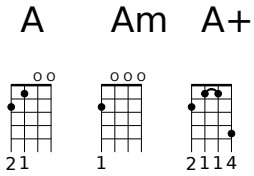
Fret diagrams for the ukulele are contained in the file `predefined-ukulele-fretboards.ly`.

```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

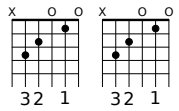
\new ChordNames {
  \myChords
}

\new FretBoards {
  \set stringTunings = #ukulele-tuning
  \myChords
}
```



Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], page 308).

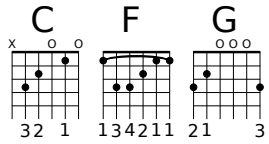
```
\include "predefined-guitar-fretboards.ly"
\context FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```



It is common that both chord names and fret diagrams are displayed together. This is achieved by putting a `ChordNames` context in parallel with a `FretBoards` context and giving both contexts the same music.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

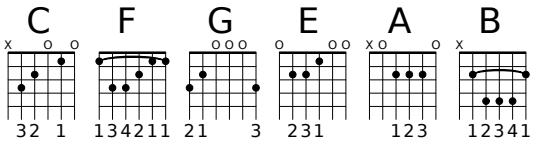
<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



Predefined fret diagrams are transposable, as long as a diagram for the transposed chord is stored in the fret diagram table.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}

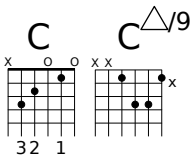
mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \context ChordNames {
    \mychordlist
  }
  \context FretBoards {
    \mychordlist
  }
>>
```



The predefined fret diagram table for guitar contains eight chords (major, minor, augmented, diminished, dominant seventh, major seventh, minor seventh, dominant ninth) for each of 17 keys. The predefined fret diagram table for ukulele contains these chords plus an additional three chords (major sixth, suspended second, and suspended fourth). A complete list of the predefined fret diagrams is shown in Section A.3 [Predefined fretboard diagrams], page 486. If there is no entry in the table for a chord, the FretBoards engraver will calculate a fret-diagram using the automatic fret diagram functionality described in [Automatic fret diagrams], page 277.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



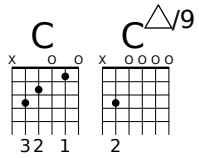
Fret diagrams can be added to the fret diagram table. To add a diagram, you must specify the hash table for the diagram, the chord for the diagram, the tuning to be used, and a definition for the diagram. Normally, the hash table will be *default-fret-table*. The diagram definition can be either a fret-diagram-terse definition string or a fret-diagram-verbose marking list.

```
\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
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



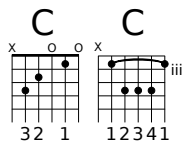
Different fret diagrams for the same chord name can be stored using different octaves of pitches. The different octave should be at least two octaves above or below the default octave, because the octaves above and below the default octave are used for transposing fretboards.

```
\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''
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



In addition to fret diagrams, LilyPond stores an internal list of chord shapes. The chord shapes are fret diagrams that can be shifted along the neck to different positions to provide different chords. Chord shapes can be added to the internal list and then used to define predefined fret diagrams. Because they can be moved to various positions on the neck, chord shapes will normally not contain any open strings. Like fret diagrams, chord shapes can be entered as either fret-diagram-terse strings or fret-diagram-verbose marking lists.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

\addChordShape #'powerf #guitar-tuning #"1-1;3-3;3-4;x;x;x;"

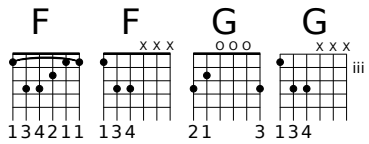
% add some new chords based on the power chord shape

\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''
}

<<
  \context ChordNames {
    \mychords
  }
  \context FretBoards {
    \mychords
  }
>>
```



The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Section "fret-diagram-interface" dans *Référence des propriétés internes*. For a predefined fret diagram, the interface properties belong to `FretBoards.FretBoard`.

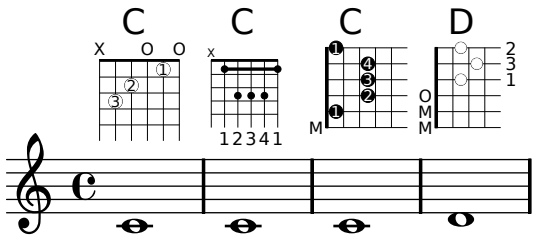
Morceaux choisis

Customizing fretboard fret diagrams

Fret diagram properties can be set through `'fret-diagram-details`. For FretBoard fret diagrams, overrides are applied to the `FretBoards.FretBoard` object. Like `Voice`, `FretBoards` is a bottom level context, therefore can be omitted in property overrides.

```
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        #"x;1-1-(;3-2;3-3;3-4;1-1-);"
<<
  \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
      \once \override FretBoard #'size = #'1.0
      \once \override FretBoard
        #'(fret-diagram-details barre-type) = #'straight
      \once \override FretBoard
        #'(fret-diagram-details dot-color) = #'black
```

```
      \once \override FretBoard
        #'(fret-diagram-details finger-code) = #'below-string
      c'
      \once \override FretBoard
        #'(fret-diagram-details barre-type) = #'none
      \once \override FretBoard
        #'(fret-diagram-details number-type) = #'arabic
      \once \override FretBoard
        #'(fret-diagram-details orientation) = #'landscape
      \once \override FretBoard
        #'(fret-diagram-details mute-string) = #"M"
      \once \override FretBoard
        #'(fret-diagram-details label-dir) = #LEFT
      \once \override FretBoard
        #'(fret-diagram-details dot-color) = #'black
      c'
      \once \override FretBoard
        #'(fret-diagram-details finger-code) = #'below-string
      \once \override FretBoard
        #'(fret-diagram-details dot-radius) = #0.35
      \once \override FretBoard
        #'(fret-diagram-details dot-position) = #0.5
      \once \override FretBoard
        #'(fret-diagram-details fret-count) = #3
      d
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>
```



Defining predefined fretboards for other instruments

Predefined fret diagrams can be added for new instruments in addition to the standards used for guitar. This file shows how this is done by defining a new string-tuning and a few predefined fretboards for the Venezuelan cuatro.

This file also shows how fingerings can be included in the chords used as reference points for the chord lookup, and displayed in the fret diagram and the `TabStaff`, but not the music.

These fretboards are not transposable because they contain string information. This is planned to be corrected in the future.

```
% 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 = #'(11 18 14 9)

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
```

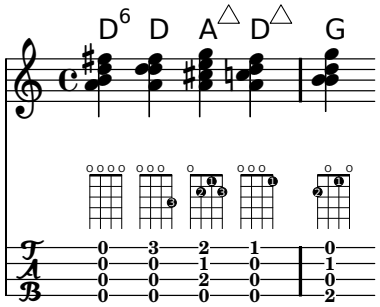
```
      }
    }

    \new FretBoards {
      \set 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 { }
}
```



chordChanges for FretBoards

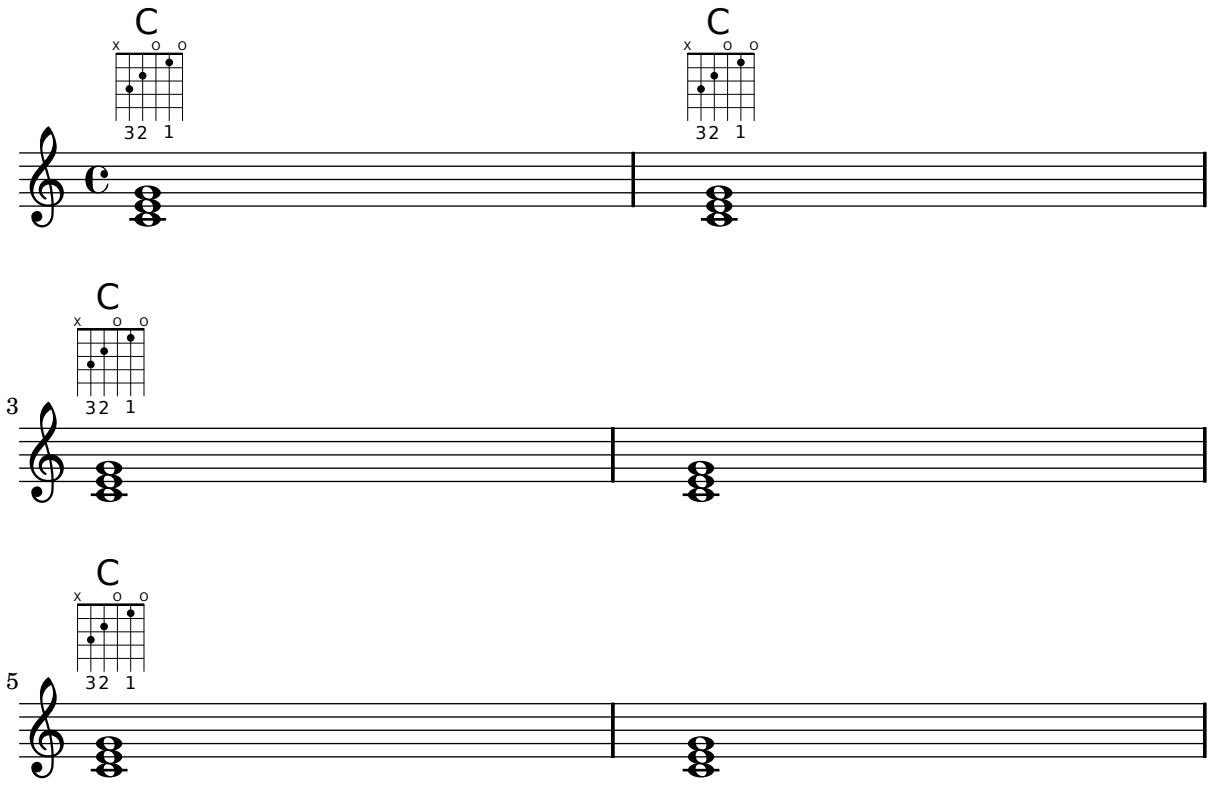
FretBoards can be set to display only when the chord changes or at the beginning of a new line.

```
\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1 \break
}

<<
```

```
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```



Alternate fretboard tables

Alternate fretboard tables can be created. These would be used in order to have alternate fretboards for a given chord.

In order to use an alternate fretboard table, the table must first be created. Fretboards are then added to the table.

The table to be used in displaying predefined fretboards is selected by the property `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"

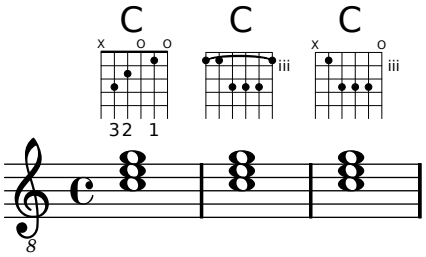
#(define custom-fretboard-table-one (make-hash-table 101))

#(define custom-fretboard-table-two (make-hash-table 101))

\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode{c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode{c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"
```

```
<<
  \chords {
    c1 |
    c1 |
    c1
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
      c1
      \set predefinedDiagramTable = #custom-fretboard-table-one
      c1
      \set predefinedDiagramTable = #custom-fretboard-table-two
      c1
    }
  }
  \new Staff {
    \clef "treble_8"
    \chordmode {
      c1 |
      c1 |
      c1
    }
  }
>>
```



Voir aussi

Notation Reference: [Custom tablatures], page 257, [Automatic fret diagrams], page 277, [Chord mode overview], page 308, Section A.3 [Predefined fretboard diagrams], page 486.

Installed Files: ‘`ly/predefined-guitar-fretboards.ly`’, ‘`ly/predefined-guitar-ninth-fretboards.ly`’, ‘`ly/predefined-ukulele-fretboards.ly`’.

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

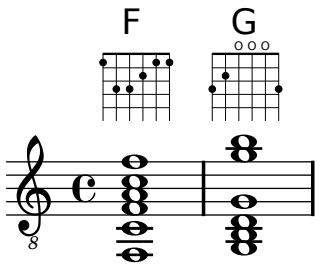
Internals Reference: Section “fret-diagram-interface” dans *Référence des propriétés internes*.

Automatic fret diagrams

Fret diagrams can be automatically created from entered notes using the `FretBoards` context. If no predefined diagram is available for the entered notes in the active `stringTunings`, this context calculates strings and frets that can be used to play the notes.

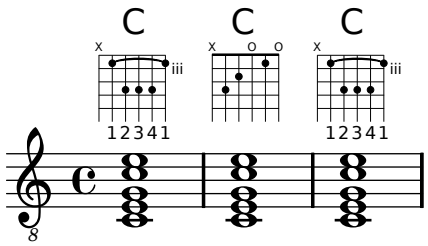
```
<<
  \context ChordNames {
    \chordmode {
```

```
      f1 g
    }
  }
  \context FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \context Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b' g'>1
  }
>>
```



As no predefined diagrams are loaded by default, automatic calculation of fret diagrams is the default behavior. Once default diagrams are loaded, automatic calculation can be enabled and disabled with predefined commands:

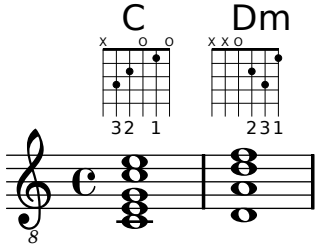
```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        #"x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \context ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \context FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
    <c e g c' e'>1
  }
>>
```



Sometimes the fretboard calculator will be unable to find an acceptable diagram. This can often be remedied by manually assigning a note to a string. In many cases, only one note need be manually placed on a string; the rest of the notes will then be placed appropriately by the `FretBoards` context.

Fingerings can be added to FretBoard fret diagrams.

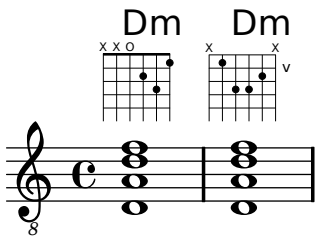
```
<<
  \context ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \context FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \context Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <d a d' f'>1
  }
>>
```



The minimum fret to be used in calculating strings and frets for the FretBoard context can be set with the `minimumFret` property.

```
<<
  \context ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \context FretBoards {
    <d a d' f'>1
    \set FretBoards.minimumFret = #5
    <d a d' f'>1
  }
  \context Staff {
    \clef "treble_8"
    <d a d' f'>1
```

```
    <d a d' f'>1
  }
>>
```



The strings and frets for the `FretBoards` context depend on the `stringTunings` property, which has the same meaning as in the TabStaff context. See [Custom tablatures], page 257 for information on the `stringTunings` property.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Section "fret-diagram-interface" dans *Référence des propriétés internes*. For a `FretBoards` fret diagram, the interface properties belong to `FretBoards.FretBoard`.

Commandes prédéfinies

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

Voir aussi

Notation Reference: [Custom tablatures], page 257.

Snippets: Section "Fretted strings" dans *Morceaux choisis*.

Internals Reference: Section "fret-diagram-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

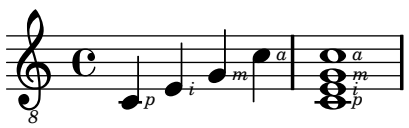
Automatic fretboard calculations do not work properly for instruments with non-monotonic tunings.

Right-hand fingerings

Right-hand fingerings *p-i-m-a* must be entered within a chord construct `<>` for them to be printed in the score, even when applied to a single note.

> **Note :** There **must** be a hyphen before `\rightHandFinger` and a space before the closing `>`.

```
\clef "treble_8"
<c-\rightHandFinger #1 >4
<e-\rightHandFinger #2 >
<g-\rightHandFinger #3 >
<c-\rightHandFinger #4 >
<c,-\rightHandFinger #1 e-\rightHandFinger #2
 g-\rightHandFinger #3 c-\rightHandFinger #4 >1
```



For convenience, you can abbreviate `\rightHandFinger` to something short, for example `RH`,

```
#(define RH rightHandFinger)
```

Morceaux choisis

Placement of right-hand fingerings

It is possible to exercise greater control over the placement of right-hand fingerings by setting a specific property, as demonstrated in the following example. Note: you must use a chord construct

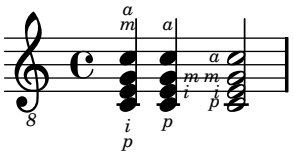
```
#(define RH rightHandFinger)

\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
}
```

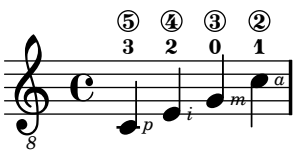


Fingerings string indications and right-hand fingerings

This example combines left-hand fingering, string indications, and right-hand fingering.

```
#(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

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

Internals Reference: Section “StrokeFinger” dans *Référence des propriétés internes*.
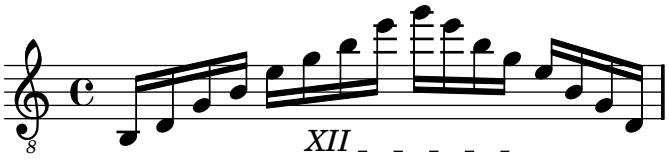
2.4.2 Guitar

Most of the notational issues associated with guitar music are covered sufficiently in the general fretted strings section, but there are a few more worth covering here. Occasionally users want to create songbook-type documents having only lyrics with chord indications above them. Since LilyPond is a music typesetter, it is not recommended for documents that have no music notation in them. A better alternative is a word processor, text editor, or, for experienced users, a typesetter like GuitarTeX.

Indicating position and barring

This example demonstrates how to include guitar position and barring indications.

```
\clef "treble_8"
b16 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
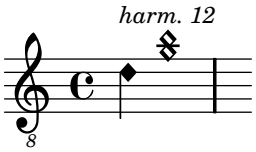
Notation Reference: ⟨undefined⟩ [Text spanners], page ⟨undefined⟩.

Snippets: Section “Fretted strings” dans *Morceaux choisis*, Section “Expressive marks” dans *Morceaux choisis*.

Indicating harmonics and dampened notes

Special note heads can be used to indicate dampened notes or harmonics. Harmonics are normally further explained with a text markup.

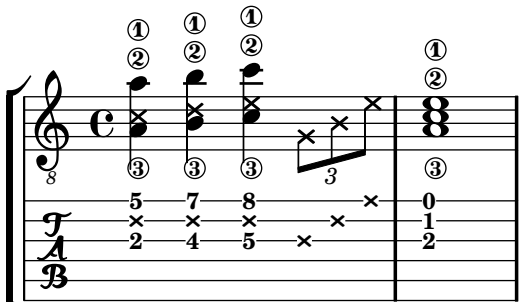
```
\relative c' {
  \clef "treble_8"
  \override Staff.NoteHead #'style = #'harmonic-mixed
  d^\markup { \italic { \fontsize #-2 { "harm. 12" }}} <g b>1
}
```



Dampened notes (also called *dead notes*) are supported within normal and tablature staves:

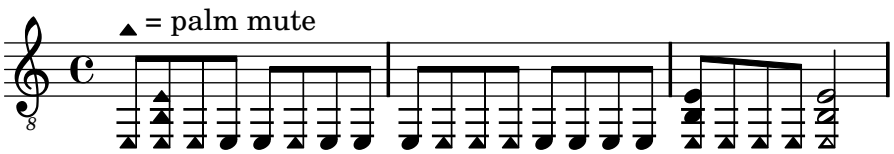
```
music = \relative c' {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \times 2/3 { g8 b e }
```

```
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
  \new TabStaff {
    \music
  }
>>
```



Another playing technique (especially used on electric guitars) is called *palm mute*. The string is hereby partly muted by the palm of the striking hand (hence the name). Lilypond supports the notation of palm mute-style notes by changing the note head to a triangle shape.

```
\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.u2do"  = 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

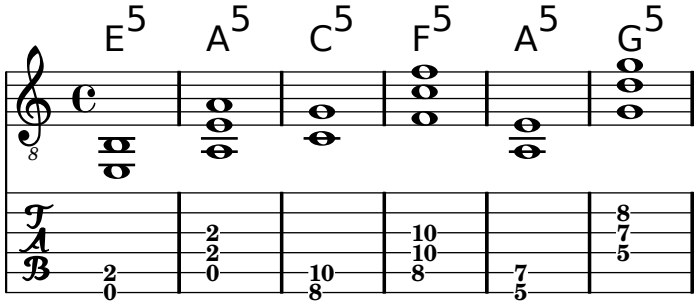
Snippets: Section “Fretted strings” dans *Morceaux choisis*.

Notation Reference: ⟨undefined⟩ [Special note heads], page ⟨undefined⟩, Section A.7 [Note head styles], page 514.

Indicating power chords

Power chords and their symbols can be engraved in chord mode or as chord constructs:

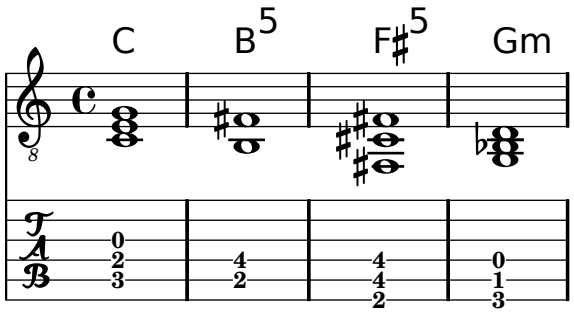
```
ChordsAndSymbols = {
  \chordmode {
    \powerChords
    e,,1:1.5
    a,,1:1.5.8
    \set minimumFret = #8
    c,1:1.5
    f,1:1.5.8
  }
  \set minimumFret = #5
  <a, e>1
  <g d' g'>1
}
\score {
  <<
    \new ChordNames {
    \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}
```



Power chord symbols are automatically switched off as soon as one of the other common chord modifier is used:

```
mixedChords = \chordmode {
  c,1
  \powerChords
  b,,1:1.5
  fis,,1:1.5.8
  g,,1:m
}
\score {
  <<
    \new ChordNames {
```

```
      \mixedChords
    }
    \new Staff {
      \clef "treble_8"
      \mixedChords
    }
    \new TabStaff {
      \mixedChords
    }
  >>
}
```



Voir aussi

Music Glossary: Section “power chord” dans *Glossaire*.

Notation Reference: [Extended and altered chords], page 310, [Printing chord names], page 313.

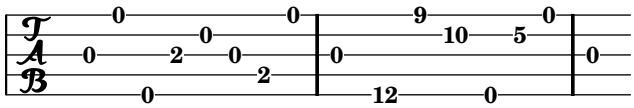
Snippets: Section “Fretted strings” dans *Morceaux choisis*.

2.4.3 Banjo

Banjo tablatures

LilyPond has basic support for the five-string banjo. When making tablatures for five-string banjo, use the banjo tablature format function to get correct fret numbers for the fifth string:

```
\new TabStaff <<
  \set TabStaff.tablatureFormat = #fret-number-tablature-format-banjo
  \set TabStaff.stringTunings = #banjo-open-g-tuning
  {
    \stemDown
    g8 d' g'\5 a b g e d' |
    g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
    g4
  }
>>
```



A number of common tunings for banjo are predefined in LilyPond: `banjo-c-tuning` (gCGBD), `banjo-modal-tuning` (gDGCD), `banjo-open-d-tuning` (aDF#AD) and `banjo-open-dm-tuning` (aDFAD).

These tunings may be converted to four-string banjo tunings using the `four-string-banjo` function:

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Voir aussi

Snippets: Section “Fretted strings” dans *Morceaux choisis*.

Installed Files: ‘`scm/output-lib.scm`’ contains predefined banjo tunings.

2.5 Percussion

2.5.1 Common notation for percussion

Rhythmic music is primarily used for percussion and drum notation, but it can also be used to show the rhythms of melodies.

References for percussion

TODO add more.

- Some percussion may be notated on a rhythmic staff; this is discussed in ⟨undefined⟩ [Showing melody rhythms], page ⟨undefined⟩, and ⟨undefined⟩ [Instantiating new staves], page ⟨undefined⟩.
- MIDI output is discussed in a separate section; please see Section 3.5.6 [Percussion in MIDI], page 388.

Voir aussi

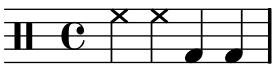
Notation Reference: ⟨undefined⟩ [Showing melody rhythms], page ⟨undefined⟩, ⟨undefined⟩ [Instantiating new staves], page ⟨undefined⟩. Section 3.5.6 [Percussion in MIDI], page 388.

Snippets: Section “Percussion” dans *Morceaux choisis*.

Basic percussion notation

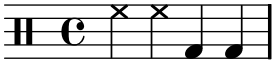
Percussion notes may be entered in `\drummode` mode, which is similar to the standard mode for entering notes. The simplest way to enter percussion notes is to use the `\drums` command, which creates the correct context and entry mode for percussion:

```
\drums {
  hihat4 hh bassdrum bd
}
```



This is shorthand for:

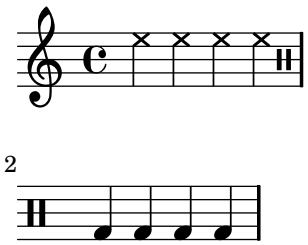
```
\new DrumStaff {
  \drummode {
    hihat4 hh bassdrum bd
  }
}
```



Each piece of percussion has a full name and an abbreviated name, and both can be used in input files. The full list of percussion note names may be found in Section A.11 [Percussion notes], page 557.

Note that the normal notation of pitches (such as `cis4`) in a `DrumStaff` context will cause an error message. Percussion clefs are added automatically to a `DrumStaff` context but they can also be set explicitly. Other clefs may be used as well.

```
\drums {
  \clef treble
  hh4 hh hh hh
  \break
  \clef percussion
  bd4 bd bd bd
}
```



There are a few issues concerning MIDI support for percussion instruments; for details please see Section 3.5.6 [Percussion in MIDI], page 388.

Voir aussi

Notation Reference: Section 3.5.6 [Percussion in MIDI], page 388, Section A.11 [Percussion notes], page 557.
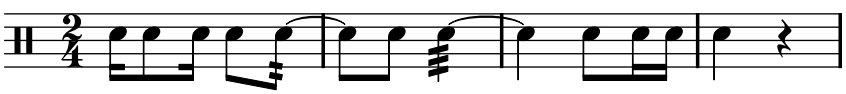
File: '`ly/drumpitch-init.ly`'

Snippets: Section "Percussion" dans *Morceaux choisis*.

Drum rolls

Drum rolls are indicated with three slashes across the stem. For quarter notes or longer the three slashes are shown explicitly, eighth notes are shown with two slashes (the beam being the third), and drum rolls shorter than eighths have one stem slash to supplement the beams. This is achieved with the tremolo notation, `:32`, as described in ⟨undefined⟩ [Tremolo repeats], page ⟨undefined⟩. Here is an example of some snare rolls:
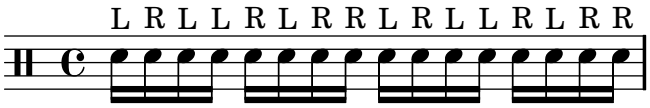
```
\drums {
  \time 2/4
  sn16 sn8 sn16 sn8 sn8:32 ~
  sn8 sn8 sn4:32 ~
  sn4 sn8 sn16 sn16
  sn4 r4
}
```



Sticking can be indicated by placing `^"R"` or `^"L"` after the note. The `staff-padding` property may be overridden to achieve a pleasing baseline.

```
\drums {
  \repeat unfold 2 {
    sn16 ^"L" sn^"R" sn^"L" sn^"L" sn^"R" sn^"L"  sn^"R" sn^"R"
  }
```

```
}
```



Voir aussi

Snippets: Section “Percussion” dans *Morceaux choisis*.

Pitched percussion

Certain pitched percussion instruments (e.g. xylophone, vibraphone, and timpani) are written using normal staves. This is covered in other sections of the manual.

Voir aussi

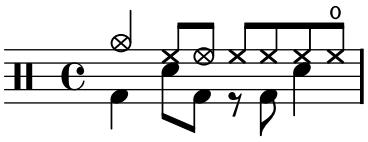
Notation Reference: Section 3.5.6 [Percussion in MIDI], page 388.

Snippets: Section “Percussion” dans *Morceaux choisis*.

Percussion staves

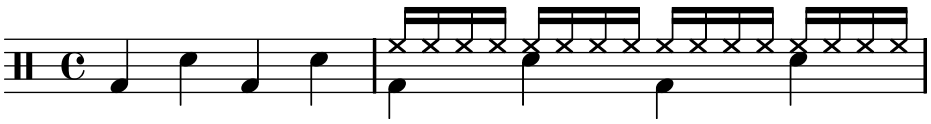
A percussion part for more than one instrument typically uses a multiline staff where each position in the staff refers to one piece of percussion. To typeset the music, the notes must be interpreted in `DrumStaff` and `DrumVoice` context.

```
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 }
>>
```



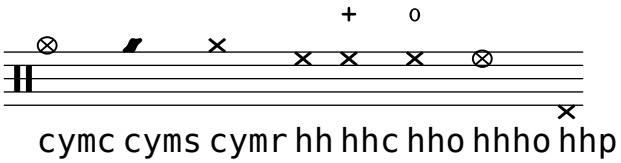
The above example shows verbose polyphonic notation. The short polyphonic notation, described in Section “I’m hearing Voices” dans *Manuel d’initiation*, can also be used. For example,

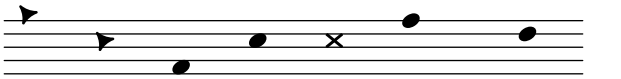
```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
  }
>>
```

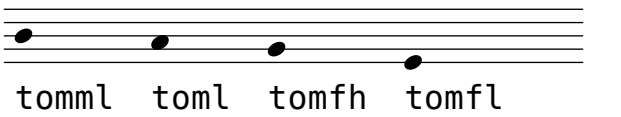


There are also other layout possibilities. To use these, set the property `drumStyleTable` in context `DrumVoice`. The following variables have been predefined:

`drums-style`
: This is the default. It typesets a typical drum kit on a five-line staff:

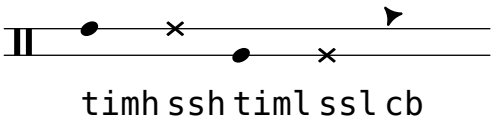




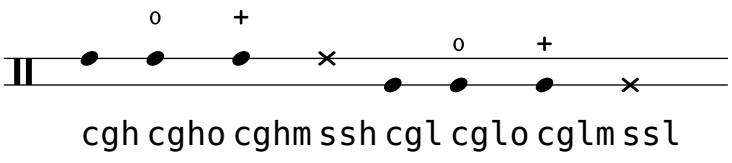


The drum scheme supports six different toms. When there are fewer toms, simply select the toms that produce the desired result. For example, to get toms on the three middle lines you use `tommh`, `tomml`, and `tomfh`.

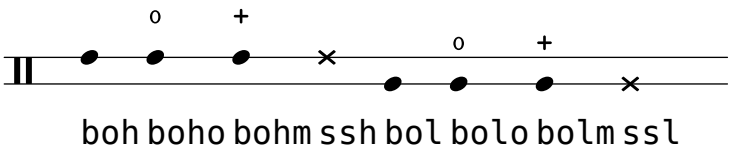
`timbales-style`
: This typesets timbales on a two line staff:



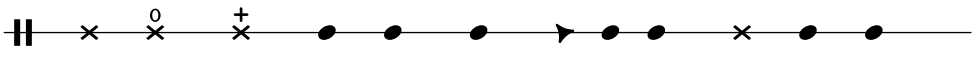
`congas-style`
: This typesets congas on a two line staff:



`bongos-style`
: This typesets bongos on a two line staff:



`percussion-style`
: To typeset all kinds of simple percussion on one line staves:

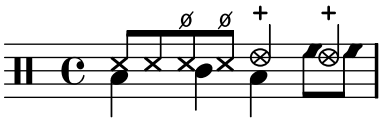


Custom percussion staves

If you do not like any of the predefined lists you can define your own list at the top of your file.

```
#(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 }
>>
```



Morceaux choisis

Here are some examples:

Two Woodblocks, entered with wbh (high woodblock) and wbl (low woodblock)

```
% These lines define the position of the woodblocks in the stave;
% if you like, you can change it or you can use special note heads
% for the woodblocks.
#(define mydrums '((hiwoodblock default #t  3)
                   (lowoodblock default #t -2)))

woodstaff = {
  % This defines a staff with only two lines.
  % It also defines the positions of the two lines.
  \override Staff.StaffSymbol #'line-positions = #'(-2 3)

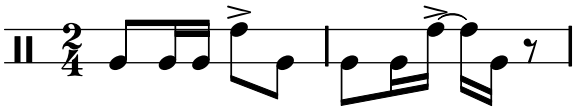
  % This is necessary; if not entered, the barline would be too short!
  \override Staff.BarLine #'bar-size = #3
}

\new DrumStaff {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  % with this you load your new drum style table
  \woodstaff

  \drummode {
    \time 2/4
```

```
    wbl8 wbl16 wbl wbh8-> wbl |
    wbl8 wbl16 wbh-> ~ wbh wbl16 r8 |
  }
}
```



Note that in this special case the length of the barline must altered with `\override Staff.BarLine #'bar-size #number`. Otherwise it would be too short. And you have also to define the positions of the two stafflines. For more information about these delicate things have a look at ⟨undefined⟩ [Staff symbol], page ⟨undefined⟩.

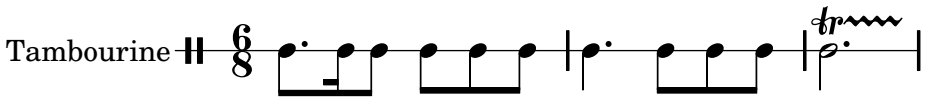
A tambourine, entered with 'tamb':

```
#(define mydrums '((tambourine default #t 0)))

tambustaff = {
  \override Staff.StaffSymbol #'line-positions = #'( 0 )
  \override Staff.BarLine #'bar-size = #3
  \set DrumStaff.instrumentName = #"Tambourine"
}

\new DrumStaff {
  \tambustaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    \time 6/8
    tamb8. tamb16 tamb8 tamb tamb tamb |
    tamb4. tamb8 tamb tamb |
    % the trick with the scaled duration and the shorter rest
    % is neccessary for the correct ending of the trill-span!
    tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
  }
}
```



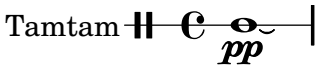
Music for Tam-Tam (entered with 'tt'):

```
#(define mydrums '((tamtam default #t 0)))

tamtamstaff = {
  \override Staff.StaffSymbol #'line-positions = #'( 0 )
  \override Staff.BarLine #'bar-size = #3
  \set DrumStaff.instrumentName = #"Tamtam"
}

\new DrumStaff {
  \tamtamstaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
```

```
  \drummode {
    tt 1 \pp \laissezVibrer
  }
}
```

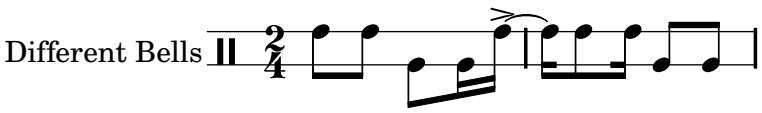


Two different bells, entered with 'cb' (cowbell) and 'rb' (ridebell)

```
#(define mydrums '((ridebell default #t  3)
                   (cowbell  default #t -2)))

bellstaff = {
  \override DrumStaff.StaffSymbol #'line-positions = #'(-2 3)
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.BarLine #'bar-size = #3
  \set DrumStaff.instrumentName = #"Different Bells"
}

\new DrumStaff {
  \bellstaff
  \drummode {
    \time 2/4
    rb8 rb cb cb16 rb-> ~ |
    rb16 rb8 rb16 cb8 cb |
  }
}
```



Here a short example taken from Stravinsky's 'L'histoire du Soldat'.

```
#(define mydrums '((bassdrum   default #t  4)
                   (snare      default #t -4)
                   (tambourine default #t  0)))

global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}

drumsA = {
  \context DrumVoice <<
    { \global }
    { \drummode {
        \autoBeamOff
        \stemDown sn8 \stemUp tamb s8 |
        sn4 \stemDown sn4 |
```
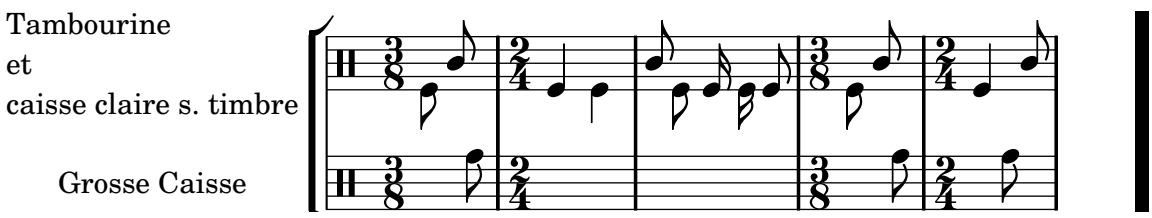
```
        \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
        \stemDown sn8 \stemUp tamb s8 |
        \stemUp sn4 s8 \stemUp tamb
      }
    }
  >>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

\score {
  \new StaffGroup <<
    \new DrumStaff {
      \set DrumStaff.instrumentName = \markup {
        \column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
      \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
      \drumsA
    }

   \new DrumStaff {
     \set DrumStaff.instrumentName = #"Grosse Caisse"
     \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
     \drumsB }
  >>
}
```



Voir aussi

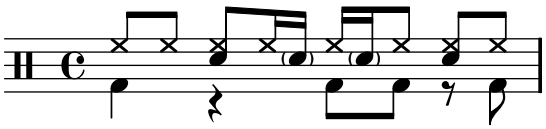
Snippets: Section “Percussion” dans *Morceaux choisis*.

Internals Reference: Section “DrumStaff” dans *Référence des propriétés internes*, Section “DrumVoice” dans *Référence des propriétés internes*.

Ghost notes

Ghost notes for drums and percussion may be created using the `\parenthesize` command detailed in [Parentheses], page 183. However, the default `\drummode` does not include the `Parenthesis_engraver` plugin which allows this.

```
\new DrumStaff \with {
  \consists "Parenthesis_engraver"
}
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ hh] <hh sn> hh16
        < \parenthesize sn > hh
        < \parenthesize sn > hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 bd r8 bd
      }
    >>
  }
>>
```

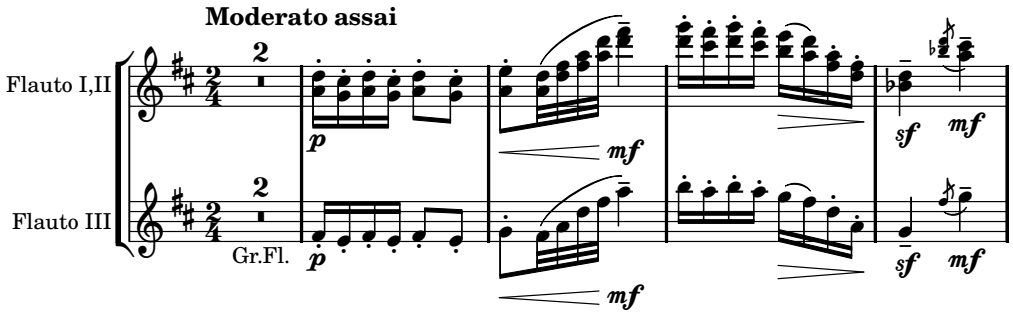


Also note that you must add chords (`< >` brackets) around each `\parenthesize` statement.

Voir aussi

Snippets: Section “Percussion” dans *Morceaux choisis*.

2.6 Wind instruments



This section includes elements of music notation that arise when writing specifically for wind instruments.

2.6.1 Common notation for wind instruments

This section discusses notation common to most wind instruments.

References for wind instruments

Many notation issues for wind instruments pertain to breathing and tonguing:

- Breathing can be specified by rests or ⟨undefined⟩ [Breath marks], page ⟨undefined⟩.
- Legato playing is indicated by ⟨undefined⟩ [Slurs], page ⟨undefined⟩.
- Different types of tonguings, ranging from legato to non-legato to staccato are usually shown by articulation marks, sometimes combined with slurs, see ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩ and Section A.10 [List of articulations], page 555.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See ⟨undefined⟩ [Tremolo repeats], page ⟨undefined⟩.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see ⟨undefined⟩ [Instrument transpositions], page ⟨undefined⟩.
- The slide glissando are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], page 114.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as ⟨undefined⟩ [Grace notes], page ⟨undefined⟩.
- Pitch inflections at the end of a note are discussed in ⟨undefined⟩ [Falls and doits], page ⟨undefined⟩.
- Key slaps or valve slaps are often shown by the `cross` style of ⟨undefined⟩ [Special note heads], page ⟨undefined⟩.
- Woodwinds can overblow low notes to sound harmonics. These are shown by the `flageolet` articulation. See Section A.10 [List of articulations], page 555.
- The use of brass mutes is usually indicated by a text markup, but where there are many rapid changes it is better to use the `stopped` and `open` articulations. See ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩ and Section A.10 [List of articulations], page 555.
- Stopped horns are indicated by the `stopped` articulation. See ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩.

Morceaux choisis

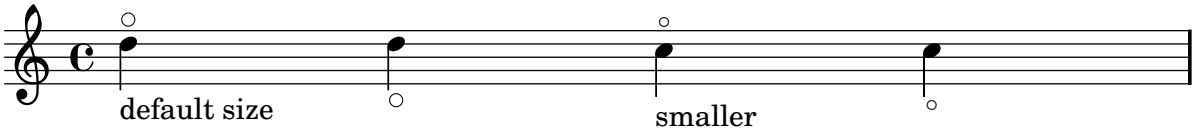
Changing \flageolet mark size

To make the `\flageolet` circle smaller use the following Scheme function.

```
smallFlageolet =
#(let ((m (make-articulation "flageolet")))
   (set! (ly:music-property m 'tweaks)
         (acons 'font-size -3
                (ly:music-property m 'tweaks)))
   m)

\layout { ragged-right = ##f }

\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



Voir aussi

Notation Reference: ⟨undefined⟩ [Breath marks], page ⟨undefined⟩, ⟨undefined⟩ [Slurs], page ⟨undefined⟩, ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩, Section A.10 [List of articulations], page 555, ⟨undefined⟩ [Tremolo repeats], page ⟨undefined⟩, ⟨undefined⟩ [Instrument transpositions], page ⟨undefined⟩, [Glissando], page 114, ⟨undefined⟩ [Grace notes], page ⟨undefined⟩, ⟨undefined⟩ [Falls and doits], page ⟨undefined⟩, ⟨undefined⟩ [Special note heads], page ⟨undefined⟩,

Snippets: Section “Winds” dans *Morceaux choisis*.

Fingerings

All wind instruments other than the trombone require the use of several fingers to produce each pitch. Some fingering examples are shown in the snippets below.

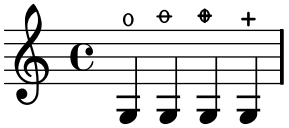
Woodwind diagrams can be produced and are described in Section 2.6.3.1 [Woodwind diagrams], page 300.

Morceaux choisis

Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```
centermarkup = {
  \once \override TextScript #'self-alignment-X = #CENTER
  \once \override TextScript #'X-offset =#(ly:make-simple-closure
    `(,+
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::centered-on-x-parent))
      ,(ly:make-simple-closure (list
        ly:self-alignment-interface::x-aligned-on-self))))
}
\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
  }
}
```



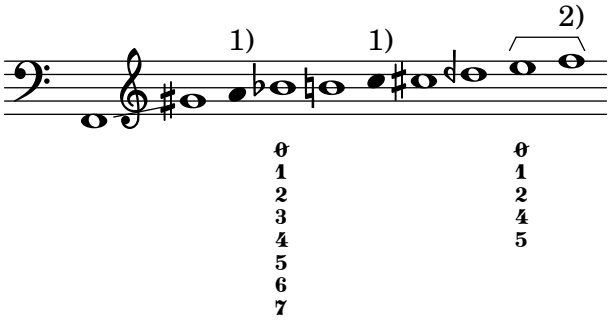
Recorder fingering chart

The following example demonstrates how fingering charts for wind instruments can be realized.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript #'self-alignment-X = #CENTER
  \once \override TextScript #'X-offset =#(ly:make-simple-closure
  `(,+
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::centered-on-x-parent))
  ,(ly:make-simple-closure (list
  ly:self-alignment-interface::x-aligned-on-self))))
}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \override Stem #'stencil = ##f
    \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

Notation Reference: Section 2.6.3.1 [Woodwind diagrams], page 300.

Snippets: Section “Winds” dans *Morceaux choisis*.

2.6.2 Bagpipes

This section discusses notation common bagpipes.

Bagpipe definitions

LilyPond contains special definitions for Scottish, Highland Bagpipe music; to use them, add
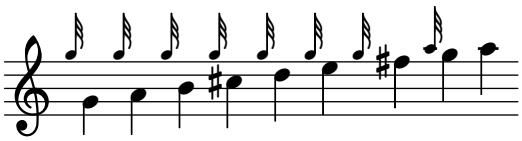
```
\include "bagpipe.ly"
```

to the top of your input file. This lets you add the special grace notes common to bagpipe music with short commands. For example, you could write `\taor` instead of

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` also contains pitch definitions for the bagpipe notes in the appropriate octaves, so you do not need to worry about `\relative` or `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



Bagpipe music nominally uses the key of D Major (even though that isn’t really true). However, since that is the only key that can be used, the key signature is normally not written out. To set this up correctly, always start your music with `\hideKeySignature`. If you for some reason want to show the key signature, you can use `\showKeySignature` instead.

Some modern music use cross fingering on c and f to flatten those notes. This can be indicated by `cflat` or `fflat`. Similarly, the piobaireachd high g can be written `gflat` when it occurs in light music.

Voir aussi

Section “Winds” dans *Morceaux choisis*.

Bagpipe example

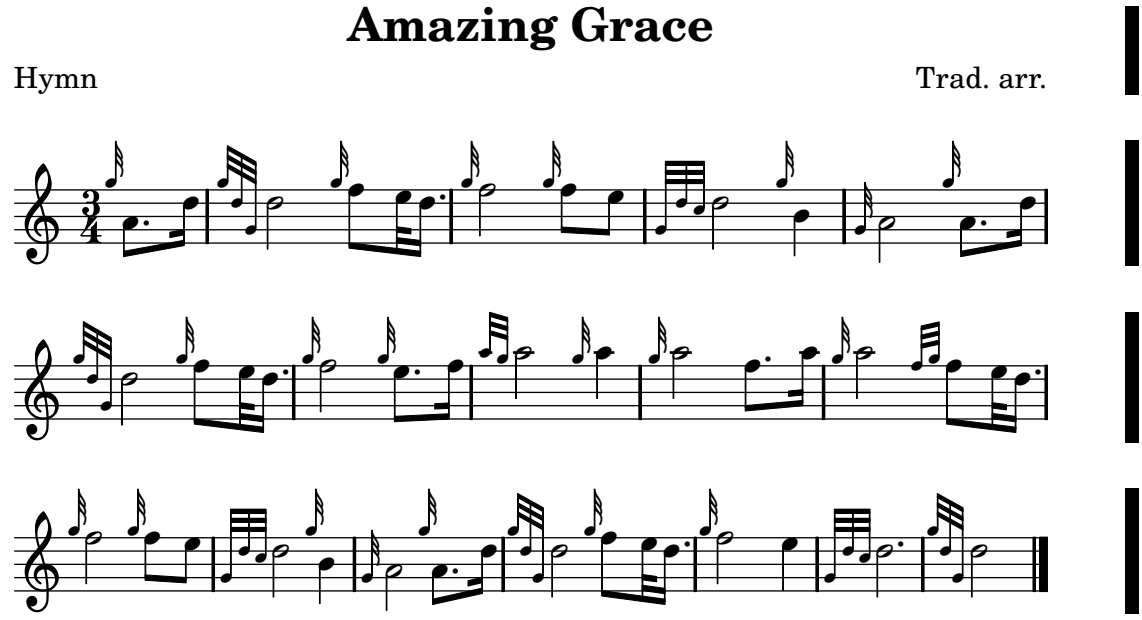
This is what the well known tune Amazing Grace looks like in bagpipe notation.

```
\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

Section “Winds” dans *Morceaux choisis*.

2.6.3 Woodwinds

This section discusses notation specifically for woodwind instruments.

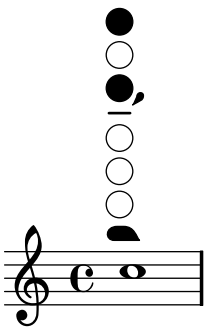
2.6.3.1 Woodwind diagrams

Woodwind diagrams can be used to indicate the fingering to be used for specific notes and are available for the following instruments:

- piccolo
- flute
- oboe
- clarinet
- bass clarinet
- saxophone
- bassoon
- contrabassoon

Woodwind diagrams are created as markups:

```
c1^\markup {
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
}
```



Keys can be open, partially-covered, ring-depressed, or fully covered:

```
\textLengthOn
c1^\markup {
  \center-column {
    "one quarter"
    \woodwind-diagram #'flute #'((cc . (one1q))
                                 (lh . ())
                                 (rh . ()))
  }
}

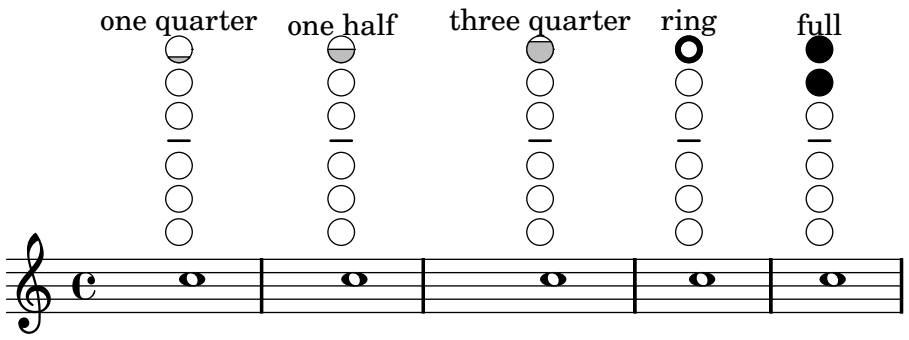
c1^\markup {
  \center-column {
    "one half"
    \woodwind-diagram #'flute #'((cc . (one1h))
                                 (lh . ())
                                 (rh . ()))
  }
```

```
}

c1^\markup {
  \center-column {
    "three quarter"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                 (lh . ())
                                 (rh . ()))
  }
}

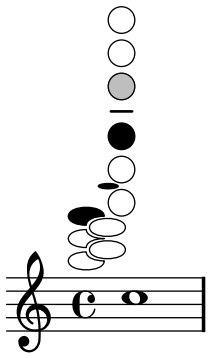
c1^\markup {
  \center-column {
    "ring"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "full"
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                 (lh . ())
                                 (rh . ()))
  }
}
```



Trills are indicated as shaded keys:

```
c1^\markup {
  \woodwind-diagram #'bass-clarinet
                    #'((cc . (threeT four))
                       (lh . ())
                       (rh . (b fis)))
}
```



A variety of trills can be displayed:

```
\textLengthOn
c1^\markup {
  \center-column {
    "one quarter to ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

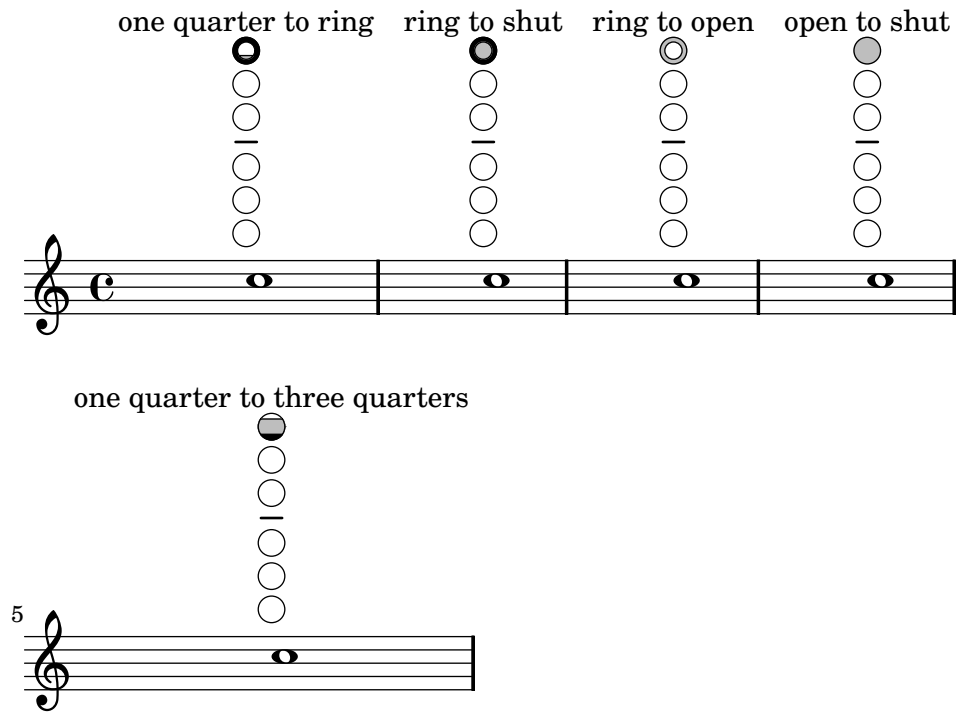
c1^\markup {
  \center-column {
    "ring to shut"
    \woodwind-diagram #'flute #'((cc . (oneTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "ring to open"
    \woodwind-diagram #'flute #'((cc . (oneRT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "open to shut"
    \woodwind-diagram #'flute #'((cc . (oneT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c1^\markup {
  \center-column {
    "one quarter to three quarters"
    \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                 (lh . ())
```

```
                                (rh . ()))
  }
}
```



The list of all possible keys and settings for a given instrument can be displayed on the console or in the log file, although they will not show up in the music output:

```
#(print-keys-verbose 'flute)
```

New diagrams can be created by following the patterns in '`scm/define-woodwind-diagrams.scm`' and '`scm/display-woodwind-diagrams.scm`'. However, this will require Scheme ability and may not be accessible to all users.

Commandes prédéfinies

Morceaux choisis

Woodwind diagrams listing

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```
\relative c' {
  \textLengthOn
  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
      #'()
  }
}

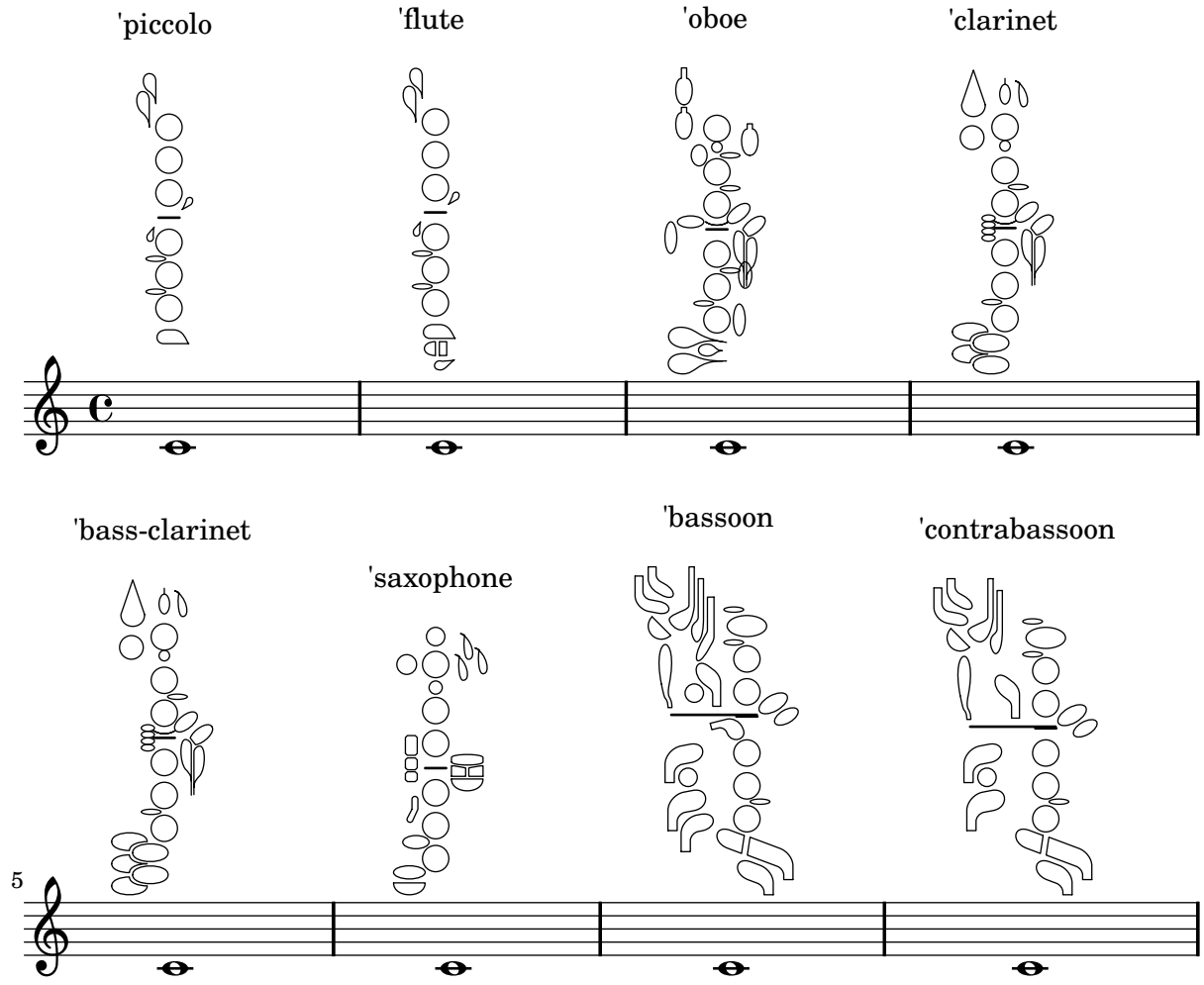
c1^\markup {
  \center-column {
    'bass-clarinet
    " "
    \woodwind-diagram
      #'bass-clarinet
      #'()
  }
}

c1^\markup {
  \center-column {
    'saxophone
    " "
    \woodwind-diagram
      #'saxophone
      #'()
  }
}

c1^\markup {
```

```
    \center-column {
      'bassoon
      " "
      \woodwind-diagram
        #'bassoon
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'contrabassoon
      " "
      \woodwind-diagram
        #'contrabassoon
        #'()
    }
  }
}
```

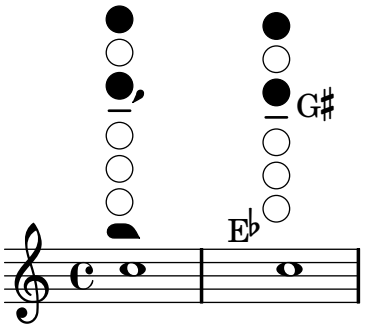


Graphical and text woodwind diagrams

In may cases, the keys other than the central column can be displayed by key name as well as by graphical means.

```
\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)))
    }
}
```

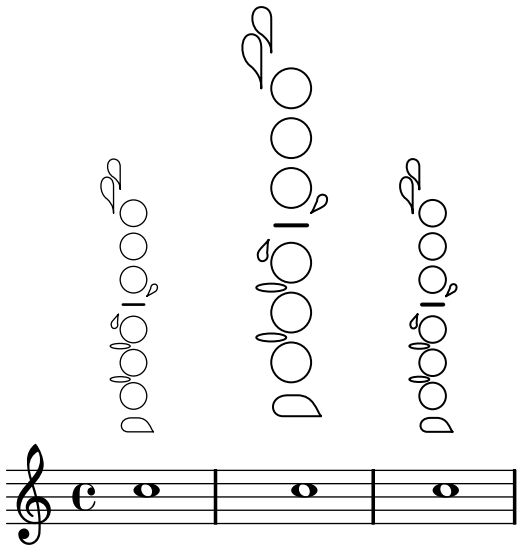


Changing the size of woodwind diagrams

The size and thickness of woodwind diagrams can be changed.

```
\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
        #'()
    }
}
```



Woodwind diagrams key lists

The snippet below produces a list of all possible keys and key settings for woodwind diagrams as defined in '`scm/define-woodwind-diagrams.scm`'. The list will be displayed on the console and in the log file, but not in the music.

```
#(print-keys-verbose 'piccolo)
#(print-keys-verbose 'flute)
#(print-keys-verbose 'flute-b-extension)
#(print-keys-verbose 'oboe)
#(print-keys-verbose 'clarinet)
#(print-keys-verbose 'bass-clarinet)
#(print-keys-verbose 'low-bass-clarinet)
#(print-keys-verbose 'saxophone)
#(print-keys-verbose 'baritone-saxophone)
#(print-keys-verbose 'bassoon)
#(print-keys-verbose 'contrabassoon)
```

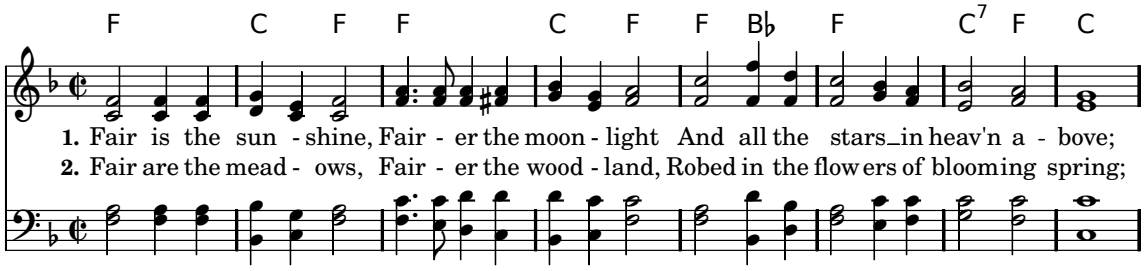
Voir aussi

Installed Files: '`scm/define-woodwind-diagrams.scm`', '`scm/display-woodwind-diagrams.scm`'.

Snippets: Section "Winds" dans *Morceaux choisis*.

Internals Reference: Section "TextScript" dans *Référence des propriétés internes*, Section "instrument-specific-markup-interface" dans *Référence des propriétés internes*.

2.7 Chord notation



Chords can be entered either as normal notes or in chord mode and displayed using a variety of traditional European chord naming conventions. Chord names and figured bass notation can also be displayed.

2.7.1 Chord mode

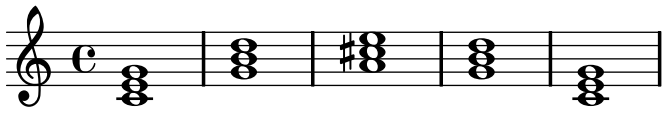
Chord mode is used to enter chords using an indicator of the chord structure, rather than the chord pitches.

Chord mode overview

Chords can be entered as simultaneous music, as discussed in ⟨undefined⟩ [Chorded notes], page ⟨undefined⟩.

Chords can also be entered in « chord mode », which is an input mode that focuses on the structures of chords in traditional European music, rather than on specific pitches. This is convenient for those who are familiar with using chord names to describe chords. More information on different input modes can be found at Section 5.4.1 [Input modes], page 457.

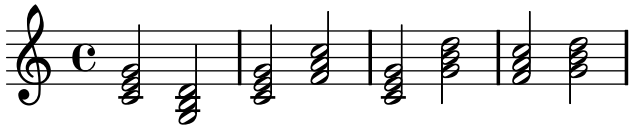
```
\chordmode { c1 g a g c }
```



Chords entered using chord mode are music elements, and can be transposed just like chords entered using simultaneous music. `\chordmode` is absolute, as `\relative` has no effect on `chordmode` blocks. However, in `\chordmode` the absolute pitches are one octave higher than in note mode.

Chord mode and note mode can be mixed in sequential music:

```
<c e g>2 <g b d>
\chordmode { c2 f }
<c e g>2 <g' b d>
\chordmode { f2 g }
```



Voir aussi

Music Glossary: Section “chord” dans *Glossaire*.

Notation Reference: ⟨undefined⟩ [Chorded notes], page ⟨undefined⟩, Section 5.4.1 [Input modes], page 457.
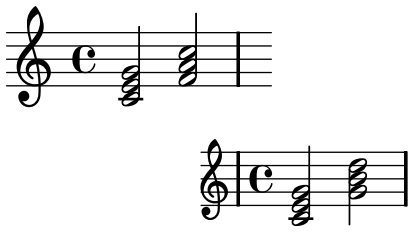
Snippets: Section “Chords” dans *Morceaux choisis*.

Problèmes connus et avertissements

Predefined shorthands for articulations and ornaments cannot be used on notes in chord mode, see ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩.
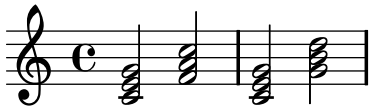
When chord mode and note mode are mixed in sequential music, and chord mode comes first, the note mode will create a new `Staff` context:

```
\chordmode { c2 f }
<c e g>2 <g' b d>
```



To avoid this behavior, explicitly create the `Staff` context:

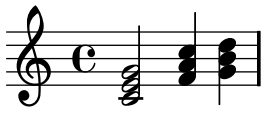
```
\new Staff {
  \chordmode { c2 f }
  <c e g>2  <g' b d>
}
```



Common chords
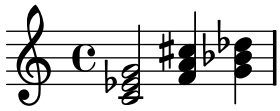
Major triads are entered by including the root and an optional duration:

```
\chordmode { c2 f4 g }
```



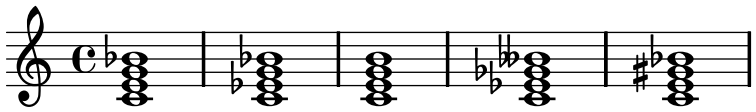
Minor, augmented, and diminished triads are entered by placing `:` and a quality modifier string after the duration:

```
\chordmode { c2:m f4:aug g:dim }
```

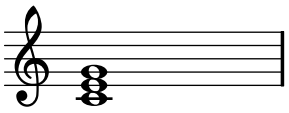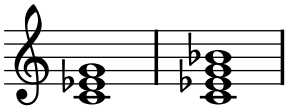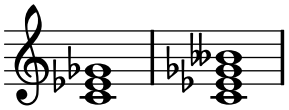


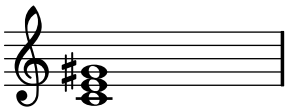
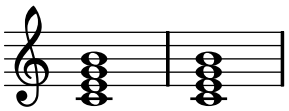
Seventh chords can be created:

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



The table belows shows the actions of the quality modifiers on triads and seventh chords. The default seventh step added to chords is a minor or flatted seventh, which makes the dominant seventh the basic seventh chord. All alterations are relative to the dominant seventh. A more complete table of modifier usage is found at Section A.2 [Common chord modifiers], page 483.

Modifier	**Action**	**Example**
None	The default action; produces a major triad.	

m, m7	The minor chord. This modifier lowers the 3rd.	
dim, dim7	The diminished chord. This modifier lowers the 3rd, 5th and (if present) the 7th step.	
aug	The augmented chord. This modifier raises the 5th step.	
maj, maj7	The major 7th chord. This modifier adds a raised 7th step. The 7 following `maj` is optional. Do NOT use this modifier to create a major triad.	

Voir aussi

Notation Reference: Section A.2 [Common chord modifiers], page 483, [Extended and altered chords], page 310.

Snippets: Section “Chords” dans *Morceaux choisis*.

Problèmes connus et avertissements

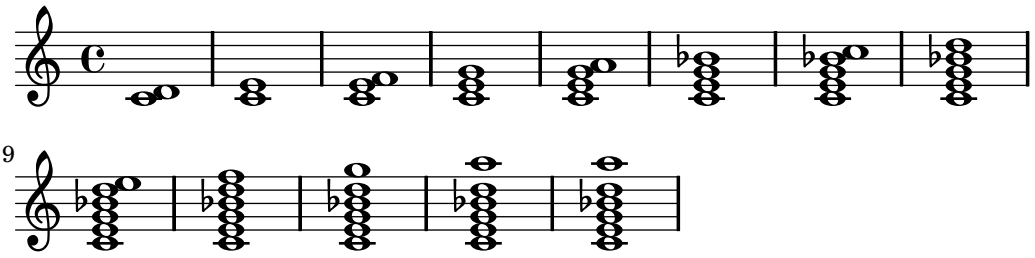
Only one quality modifier should be used per chord, typically on the highest step present in the chord. Chords with more than quality modifier will be parsed without an error or warning, but the results are unpredictable. Chords that cannot be achieved with a single quality modifier should be altered by individual pitches, as described in [Extended and altered chords], page 310.

Extended and altered chords

Chord structures of arbitrary complexity can be created in chord mode. The modifier string can be used to extend a chord, add or remove chord steps, raise or lower chord steps, and add a bass note or create an inversion.

The first number following the `:` is taken to be the extent of the chord. The chord is constructed by sequentially adding thirds to the root until the specified number has been reached. Note that the seventh step added as part of an extended chord will be the minor or flatted seventh, not the major seventh. If the extent is not a third (e.g., 6), thirds are added up to the highest third below the extent, and then the step of the extent is added. The largest possible value for the extent is 13. Any larger value is interpreted as 13.

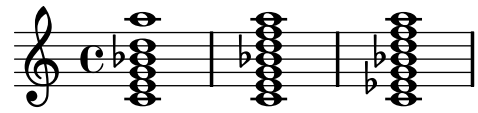
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



Note that both `c:5` and `c` produce a C major triad.
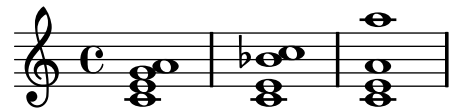
Since an unaltered 11 does not sound good when combined with an unaltered 13, the 11 is removed from a `:13` chord (unless it is added explicitly).

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



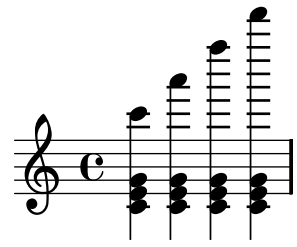
Individual steps can be added to a chord. Additions follow the extent and are prefixed by a dot (`.`). The basic seventh step added to a chord is the minor or flatted seventh, rather than the major seventh.

```
\chordmode {
  c1:5.6 c:3.7.8 c:3.6.13
}
```
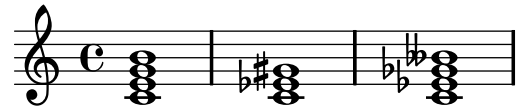


Added steps can be as high as desired.

```
\chordmode {
  c4:5.15 c:5.20 c:5.25 c:5.30
}
```



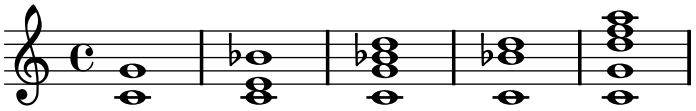
Added chord steps can be altered by suffixing a `-` or `+` sign to the number. To alter a step that is automatically included as part of the basic chord structure, add it as an altered step.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



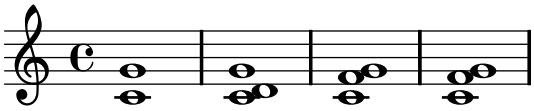
Following any steps to be added, a series of steps to be removed is introduced in a modifier string with a prefix of `^`. If more than one step is to be removed, the steps to be removed are separated by `.` following the initial `^`.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
}
```



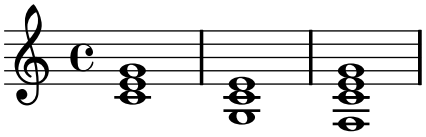
The modifier `sus` can be added to the modifier string to create suspended chords. This removes the 3rd step from the chord. Append either `2` or `4` to add the 2nd or 4th step to the chord. `sus` is equivalent to `^3`; `sus4` is equivalent to `.4^3`.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4^3
}
```



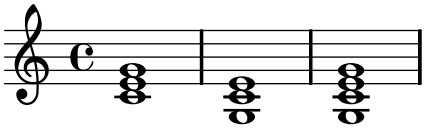
Inversions (putting a pitch other than the root on the bottom of the chord) and added bass notes can be specified by appending */pitch* to the chord.

```
\chordmode {
  c1 c/g c/f
}
```



A bass note that is part of the chord can be added, instead of moved as part of an inversion, by using */+pitch*.

```
\chordmode {
  c1 c/g c/+g
}
```



Chord modifiers that can be used to produce a variety of standard chords are shown in Section A.2 [Common chord modifiers], page 483.

Voir aussi

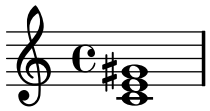
Notation Reference: Section A.2 [Common chord modifiers], page 483.

Snippets: Section “Chords” dans *Morceaux choisis*.

Problèmes connus et avertissements

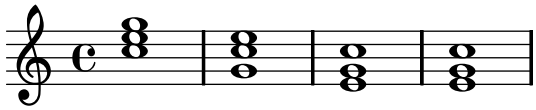
Each step can only be present in a chord once. The following simply produces the augmented chord, since `5+` is interpreted last.

```
\chordmode { c1:5.5-.5+ }
```



Only the second inversion can be created by adding a bass note. The first inversion requires changing the root of the chord.

```
\chordmode {
  c'1: c':/g e:6-3-^5 e:m6-^5
}
```



2.7.2 Displaying chords

Chords can be displayed by name, in addition to the standard display as notes on a staff.

Printing chord names

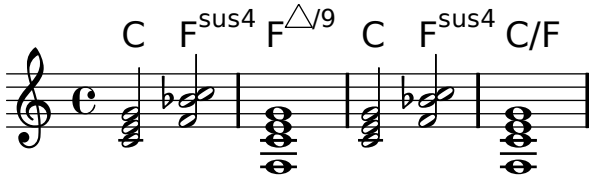
Chord names are printed in the `ChordNames` context:

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```
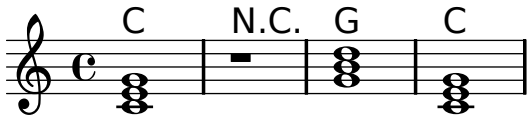


Chords can be entered as simultaneous notes or through the use of chord mode. The displayed chord name will be the same, regardless of the mode of entry, unless there are inversions or added bass notes:

```
<<
  \new ChordNames {
    <c e g>2  <f bes c>
    <f c' e g>1
    \chordmode {
      c2 f:sus4 c1:/f
    }
  }
  {
    <c e g>2  <f bes c>
    <f, c' e g>1
    \chordmode {
      c2 f:sus4 c1:/f
    }
  }
>>
```



Rests passed to a `ChordNames` context will cause the `noChordSymbol` markup to be displayed.

```
<<
  \new ChordNames \chordmode {
    c1
    r1
    g1
    c1
  }
  \chordmode {
    c1
    r1
    g1
    c1
  }
>>
```



`\chords { ... }` is a shortcut notation for `\new ChordNames { \chordmode { ... } }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```

C Fm G△

```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```

C Fm G△

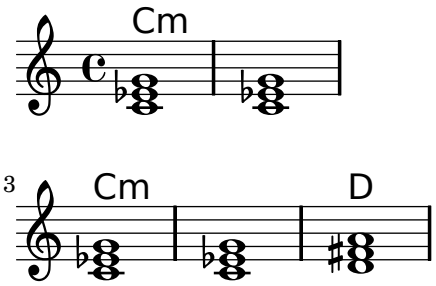
Morceaux choisis

Showing chords at changes

Chord names can be displayed only at the start of lines and when the chord changes.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
```

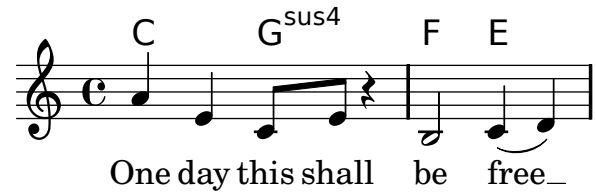
```
    \relative c' { \harmonies }
  }
>>
```



Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Music Glossary: Section “chord” dans *Glossaire*.

Notation Reference: ⟨undefined⟩ [Writing music in parallel], page ⟨undefined⟩.

Snippets: Section “Chords” dans *Morceaux choisis*.

Internals Reference: Section “ChordNames” dans *Référence des propriétés internes*, Section “ChordName” dans *Référence des propriétés internes*, Section “Chord_name_engraver” dans *Référence des propriétés internes*, Section “Volta_engraver” dans *Référence des propriétés internes*, Section “Bar_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

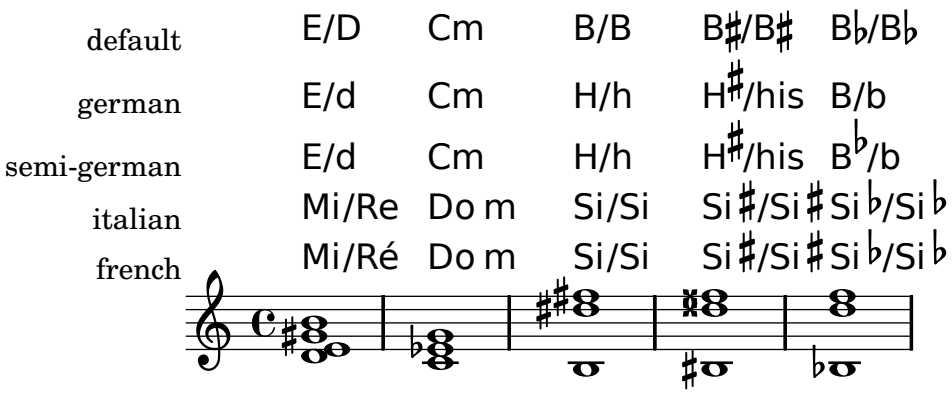
Chords containing inversions or altered bass notes are not named properly if entered using simultaneous music.

Customizing chord names

There is no unique system for naming chords. Different musical traditions use different names for the same set of chords. There are also different symbols displayed for a given chord name. The names and symbols displayed for chord names are customizable.

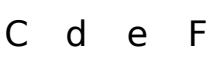
The basic chord name layout is a system for Jazz music, proposed by Klaus Ignatzek (see Section "Literature list" dans *Essai*). The chord naming system can be modified as described below. An alternate jazz chord system has been developed using these modifications. The Ignatzek and alternate Jazz notation are shown on the chart in Section A.1 [Chord name chart], page 482.

In addition to the different naming systems, different note names are used for the root in different languages. The predefined variables `\germanChords`, `\semiGermanChords`, `\italianChords` and `\frenchChords` set these variables. The effect is demonstrated here:



German songbooks may indicate minor chords as lowercase letters, without any *m* suffix. This can be obtained by setting the `chordNameLowercaseMinor` property:

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```



If none of the existing settings give the desired output, the chord name display can be tuned through the following properties.

`chordRootNamer`
: The chord name is usually printed as a letter for the root with an optional alteration. The transformation from pitch to letter is done by this function. Special note names (for example, the German 'H' for a B-chord) can be produced by storing a new function in this property.

`majorSevenSymbol`
: This property contains the markup object used to follow the output of `chordRootNamer` to identify a major 7 chord. Predefined options are `whiteTriangleMarkup` and `blackTriangleMarkup`.

`chordNoteNamer`
: When the chord name contains additional pitches other than the root (e.g., an added bass note), this function is used to print the additional pitch. By default the pitch is printed using `chordRootNamer`. The `chordNoteNamer` property can be set to a specialized function to change this behavior. For example, the bass note can be printed in lower case.

`chordNameSeparator`
: Different parts of a chord name are normally separated by a slash. By setting `chordNameSeparator`, you can use any desired markup for a separator.

`chordNameExceptions`
: This property is a list of pairs. The first item in each pair is a set of pitches used to identify the steps present in the chord. The second item is a markup that will follow the `chordRootNamer` output to create the chord name.

`chordPrefixSpacer`
: The 'm' for minor chords is usually printed immediately to the right of the root of the chord. A spacer can be placed between the root and 'm' by setting `chordPrefixSpacer`. The spacer is not used when the root is altered.

Commandes prédéfinies

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Morceaux choisis

Chord name exceptions

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

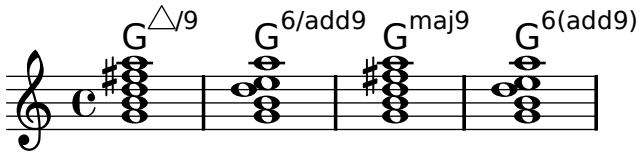
```
% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% 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
}

<< \context ChordNames \theMusic
   \context Voice \theMusic
>>
```



chord name major7

The layout of the major 7 can be tuned with `majorSevenSymbol`.

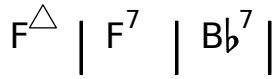
```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```



Adding bar lines to ChordNames context

To add bar line indications in the `ChordNames` context, add the `Bar_engraver`.

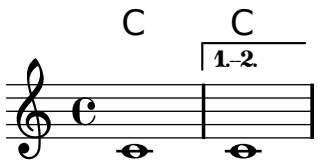
```
\new ChordNames \with {
  \override BarLine #'bar-size = #4
  \consists "Bar_engraver"
}
\chordmode {
  f1:maj7 f:7 bes:7
}
```



Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put under chords.

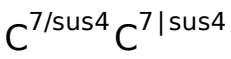
```
\score {
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



Changing chord separator

The separator between different parts of a chord name can be set to any markup.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
}
```



Voir aussi

Notation Reference: Section A.1 [Chord name chart], page 482, Section A.2 [Common chord modifiers], page 483.

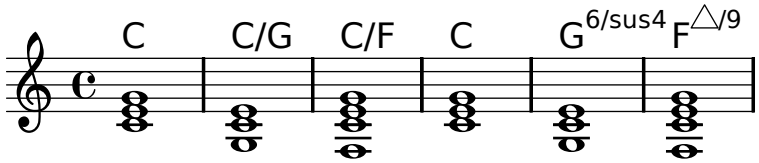
Installed Files: ‘scm/chords-ignatzek.scm’, ‘scm/chord-entry.scm’, ‘ly/chord-modifier-init.ly’.

Snippets: Section “Chords” dans *Morceaux choisis*.
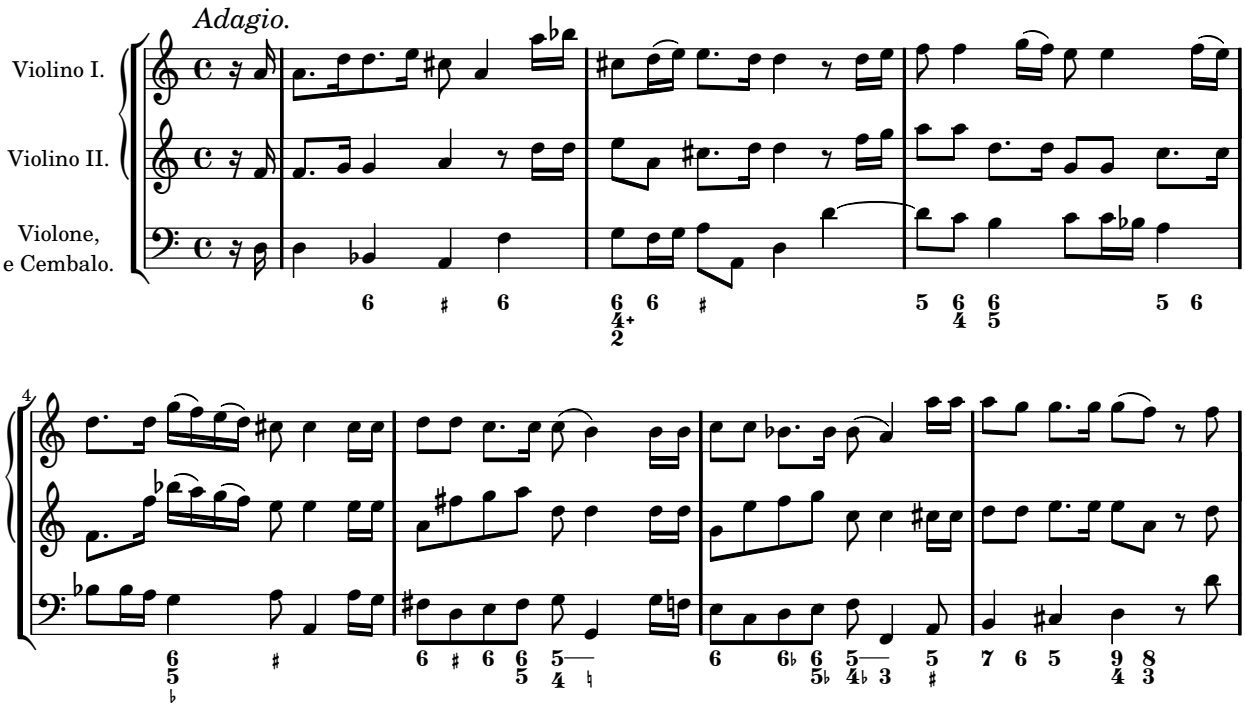
Problèmes connus et avertissements

Chord names are determined from both the pitches that are present in the chord and the information on the chord structure that may have been entered in `\chordmode`. If the simultaneous pitches method of entering chords is used, undesired names result from inversions or bass notes.

```
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 Figured bass

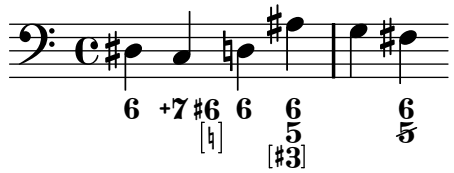


Figured bass notation can be displayed.

Introduction to figured bass

LilyPond has support for figured bass, also called thorough bass or basso continuo:

```
<<
  \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
    }
  }
>>
```



The support for figured bass consists of two parts: there is an input mode, introduced by `\figuremode`, that accepts entry of bass figures, and there is a context named `FiguredBass` that takes care of displaying `BassFigure` objects. Figured bass can also be displayed in `Staff` contexts.

`\figures{ ... }` is a shortcut notation for `\new FiguredBass { \figuremode { ... } }`.

Although the support for figured bass may superficially resemble chord support, it is much simpler. `\figuremode` mode simply stores the figures and the `FiguredBass` context prints them as entered. There is no conversion to pitches.

Voir aussi

Music Glossary: Section “figured bass” dans *Glossaire*.

Snippets: Section “Chords” dans *Morceaux choisis*.

Entering figured bass

`\figuremode` is used to switch the input mode to figure mode. More information on different input modes can be found at Section 5.4.1 [Input modes], page 457.

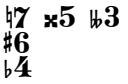
In figure mode, a group of bass figures is delimited by `<` and `>`. The duration is entered after the `>`.

```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```



Accidentals (including naturals) can be added to figures:

```
\figures {
  <7! 6+ 4-> <5++> <3-->
}
```



Augmented and diminished steps can be indicated:

```
\figures {
  <6\+ 5/> <7/>
}
```



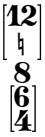
A backward slash through a figure (typically used for raised sixth steps) can be created:

```
\figures {
  <6> <6\\>
}
```



Vertical spaces and brackets can be included in figures:

```
\figures {
  <[12 _!] 8 [6  4]>
}
```



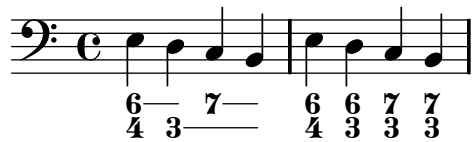
Any text markup can be inserted as a figure:

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```



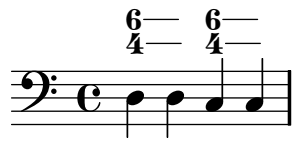
Continuation lines can be used to indicate repeated figures:

```
<<
  {
    \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>
  }
>>
```

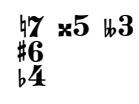


In this case, the extender lines replace existing figures, unless the continuation lines have been explicitly terminated.

```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



The table below summarizes the figure modifiers available.

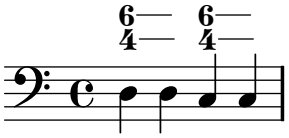
Modifier	Purpose	Example
+, -, !	Accidentals	

\+, / Augmented and diminished steps



\\ Raised sixth step

\! End of continuation line



Commandes prédéfinies

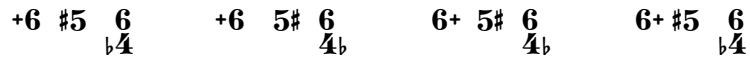
`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Morceaux choisis

Changing the positions of figured bass alterations

Accidentals and plus signs can appear before or after the numbers, depending on the `figuredBassAlterationDirection` and `figuredBassPlusDirection` properties.

```
\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

Snippets: Section “Chords” dans *Morceaux choisis*.

Internals Reference: Section “BassFigure” dans *Référence des propriétés internes*, Section “BassFigureAlignment” dans *Référence des propriétés internes*, Section “BassFigureLine” dans *Référence des propriétés internes*, Section “BassFigureBracket” dans *Référence des propriétés internes*, Section “BassFigureContinuation” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*.

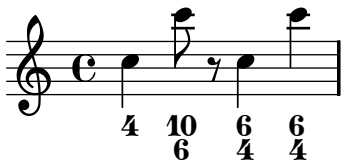
Displaying figured bass

Figured bass can be displayed using the `FiguredBass` context, or in most staff contexts.

When displayed in a `FiguredBass` context, the vertical location of the figures is independent of the notes on the staff.

```
<<
  \relative c'' {
    c4 c'8 r8 c,4 c'
  }
```
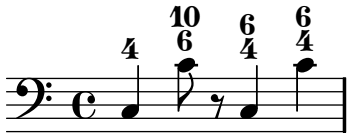
```
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



In the example above, the `FiguredBass` context must be explicitly instantiated to avoid creating a second (empty) staff.

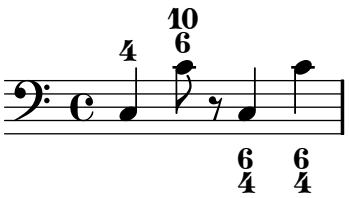
Figured bass can also be added to `Staff` contexts directly. In this case, the vertical position of the figures is adjusted automatically.

```
<<
  \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'
  }
>>
```



When added in a `Staff` context, figured bass can be displayed above or below the staff.

```
<<
  \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
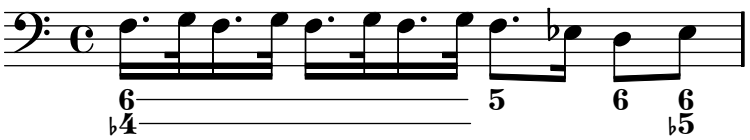
Snippets: Section “Chords” dans *Morceaux choisis*.
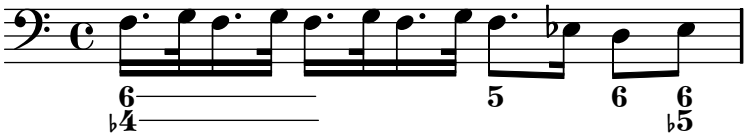
Internals Reference: Section “BassFigure” dans *Référence des propriétés internes*, Section “BassFigureAlignment” dans *Référence des propriétés internes*, Section “BassFigureLine” dans *Référence des propriétés internes*, Section “BassFigureBracket” dans *Référence des propriétés internes*, Section “BassFigureContinuation” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

To ensure that continuation lines work properly, it is safest to use the same rhythm in the figure line as in the bass line.

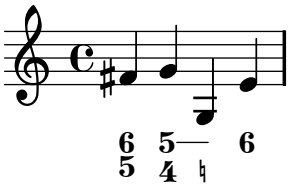
```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are correct here, with the same rhythm as the bass
    \repeat unfold 4 { <6 4->16. <6 4->32 }
    <5>8. r16 <6>8 <6\! 5->
  }
>>
<<
  {
    \clef bass
    \repeat 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->
  }
>>
```





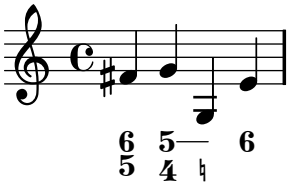
When using extender lines, adjacent figures with the same number in a different figure location can cause the figure positions to invert.

```
<<
  { fis4 g g, e' }
  \figures {
    \bassFigureExtendersOn
    <6 5>4 <5\! 4> < 5 _!> <6>
  }
>>
```
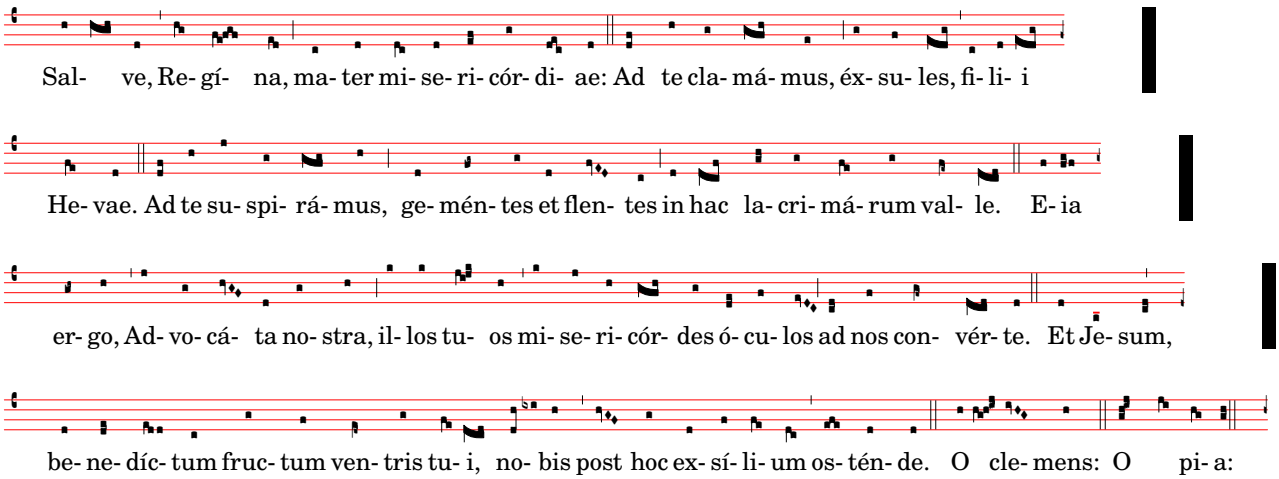


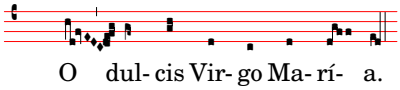
To avoid this problem, simply turn on extenders after the figure that begins the extender line and turn them off at the end of the extender line.

```
<<
  { fis4 g g, e' }
  \figures {
    <6 5>4 <5 4>
    \bassFigureExtendersOn
    < 5 _!>4 <6>
    \bassFigureExtendersOff
  }
>>
```



2.8 Ancient notation





Support for ancient notation includes features for mensural notation and Gregorian chant notation. These features can be accessed either by modifying style properties of graphical objects such as note heads and rests, or by using one of the pre-defined contexts for mensural or Gregorian notation.

Many graphical objects, such as note heads and flags, accidentals, time signatures, and rests, provide a `style` property, which can be changed to emulate several different styles of ancient notation. See

- [Mensural note heads], page 332,
- [Mensural accidentals and key signatures], page 334,
- [Mensural rests], page 333,
- [Mensural clefs], page 330,
- [Gregorian clefs], page 337,
- [Mensural flags], page 332,
- [Mensural time signatures], page 331.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], page 329,
- [Divisiones], page 338,
- [Ligatures], page 328.

2.8.1 Overview of the supported styles

Three styles are available for typesetting Gregorian chant:

- *Editio Vaticana* is a complete style for Gregorian chant, following the appearance of the Solesmes editions, the official chant books of the Vatican since 1904. LilyPond has support for all the notational signs used in this style, including ligatures, *custodes*, and special signs such as the quilisma and the oriscus.
- The *Editio Medicaea* style offers certain features used in the Medicaea (or Ratisbona) editions which were used prior to the Solesmes editions. The most significant differences from the *Vaticana* style are the clefs, which have downward-slanted strokes, and the noteheads, which are square and regular.
- The *Hufnagel* (« horseshoe nail ») or *Gothic* style mimics the writing style in chant manuscripts from Germany and Central Europe during the middle ages. It is named after the basic note shape (the *virga*), which looks like a small nail.

Three styles emulate the appearance of late-medieval and renaissance manuscripts and prints of mensural music:

- The *Mensural* style most closely resembles the writing style used in late-medieval and early renaissance manuscripts, with its small and narrow, diamond-shaped noteheads and its rests which approach a hand-drawn style.
- The *Neomensural* style is a modernized and stylized version of the former: the noteheads are broader and the rests are made up of straight lines. This style is particularly suited, e.g., for incipits of transcribed pieces of mensural music.
- The *Petrucci* style is named after Ottaviano Petrucci (1466-1539), the first printer to use movable type for music (in his *Harmonice musices odhecaton*, 1501). The style uses larger note heads than the other mensural styles.

Baroque and *Classical* are not complete styles but differ from the default style only in some details: certain noteheads (Baroque) and the quarter rest (Classical).

Only the mensural style has alternatives for all aspects of the notation. Thus, there are no rests or flags in the Gregorian styles, since these signs are not used in plainchant notation, and the Petrucci style has no flags or accidentals of its own.

Each element of the notation can be changed independently of the others, so that one can use mensural flags, petrucci noteheads, classical rests and vaticana clefs in the same piece, if one wishes.

2.8.2 Ancient notation—common features

Pre-defined contexts

For Gregorian chant and mensural notation, there are pre-defined voice and staff contexts available, which set all the various notation signs to values suitable for these styles. If one is satisfied with these defaults, one can proceed directly with note entry without worrying about the details on how to customize a context. See one of the pre-defined contexts `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice`, and `MensuralStaff`. See further
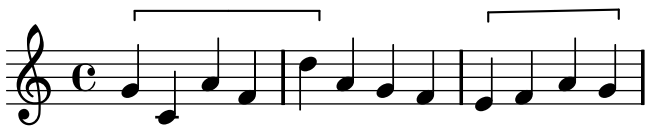
- [Gregorian chant contexts], page 336,
- [Mensural contexts], page 329.

Ligatures

A ligature is a graphical symbol that represents at least two distinct notes. Ligatures originally appeared in the manuscripts of Gregorian chant notation to denote ascending or descending sequences of notes on the same syllable. They are also used in mensural notation.

Ligatures are entered by *enclosing* them in `\[` and `\]`. Some ligature styles may need additional input syntax specific for this particular type of ligature. By default, the `LigatureBracket` engraver just puts a square bracket above the ligature.

```
\transpose c c' {
  \[ g c a f d' \]
  a g f
  \[ e f a g \]
}
```



Two other ligature styles are available: the Vaticana for Gregorian chant, and the Mensural for mensural music (only white mensural ligatures are supported for mensural music, and with certain limitations). To use any of these styles, the default `Ligature_bracket_engraver` has to be replaced with one of the specialized ligature engravers in the `Voice` context, as explained in [White mensural ligatures], page 335 and [Gregorian square neume ligatures], page 340.

Problèmes connus et avertissements

Ligatures need special spacing that has not yet been implemented. As a result, there is too much space between ligatures most of the time, and line breaking often is unsatisfactory. Also, lyrics do not correctly align with ligatures.

Accidentals must not be printed within a ligature, but instead need to be collected and printed in front of it.

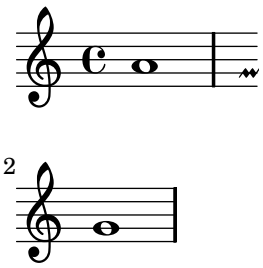
The syntax still uses the deprecated infix style `\[ music expr \]`. For consistency reasons, it will eventually be changed to postfix style `note\[ ... note\]`.

Custodes

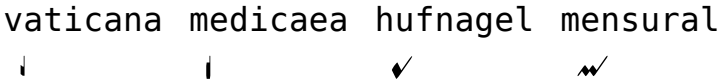
A *custos* (plural: *custodes*; Latin word for « guard ») is a symbol that appears at the end of a staff. It anticipates the pitch of the first note of the following line, thus helping the performer to manage line breaks during performance.

Custodes were frequently used in music notation until the seventeenth century. Nowadays, they have survived only in a few particular forms of musical notation such as contemporary editions of Gregorian chant like the *Editio Vaticana*. There are different custos glyphs used in different flavors of notational style.

For typesetting custodes, just put a `Custos_engraver` into the `Staff` context when declaring the `\layout` block, and change the style of the custos with an `\override` if desired, as shown in the following example:



The custos glyph is selected by the `style` property. The styles supported are `vaticana`, `medicaea`, `hufnagel`, and `mensural`. They are demonstrated in the following fragment.



Voir aussi

Internals Reference: Section “Custos” dans *Référence des propriétés internes*.

Snippets: Section “Ancient notation” dans *Morceaux choisis*.

Figured bass support

There is limited support for figured bass notation from the Baroque period; see Section 2.7.3 [Figured bass], page 320.

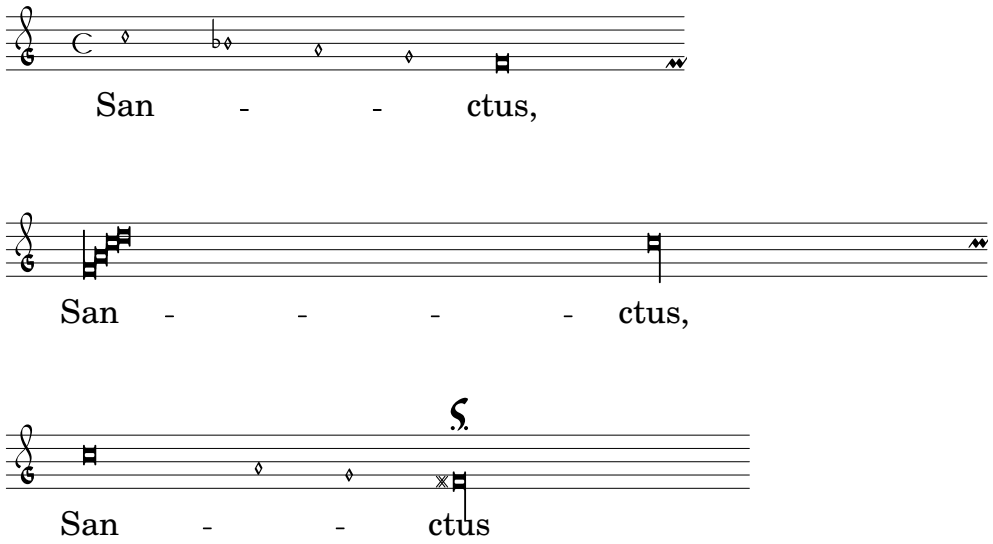
2.8.3 Typesetting mensural music

Mensural contexts

The predefined `MensuralVoice` and `MensuralStaff` contexts can be used to engrave a piece in mensural style. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\score {
  <<
    \new MensuralVoice = "discantus" \transpose c c' {
      \override Score.BarNumber #'transparent = ##t {
        c'1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c'\breve d'\melismaEnd \]
        c'\longa
```
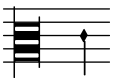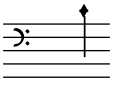
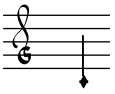
```
        c'\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```
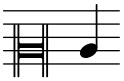
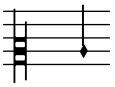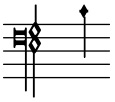
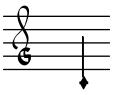


Mensural clefs

The following table shows all mensural clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. Still, you can manually force a clef glyph to be typeset on an arbitrary line, as described in ⟨undefined⟩ [Clef], page ⟨undefined⟩. The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Petrucci used C clefs with differently balanced left-side vertical beams, depending on which staff line it is printed.

Description	**Supported Clefs**	**Example**
mensural C clef	`mensural-c1`, `mensural-c2`, `mensural-c3`, `mensural-c4`	
mensural F clef	`mensural-f`	
mensural G clef	`mensural-g`	

neomensural C clef	`neomensural-c1`, `neomensural-c2`, `neomensural-c3`, `neomensural-c4`	
petrucci style C clefs, for use on different staff lines (the example shows the 2nd staff line C clef)	`petrucci-c1`, `petrucci-c2`, `petrucci-c3`, `petrucci-c4`, `petrucci-c5`	
petrucci style F clef	`petrucci-f`	
petrucci style G clef	`petrucci-g`	

Voir aussi

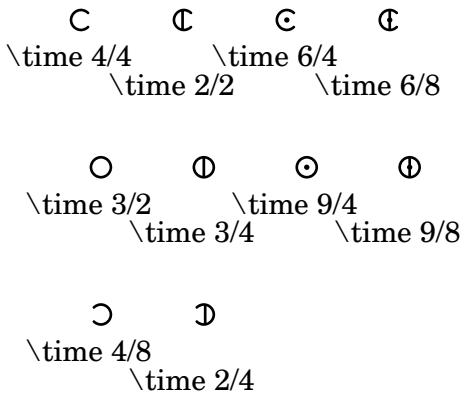
Notation Reference: ⟨undefined⟩ [Clef], page ⟨undefined⟩.

Problèmes connus et avertissements

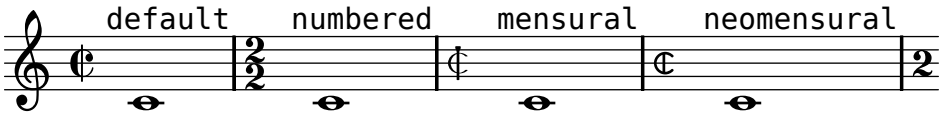
The mensural g clef is mapped to the Petrucci g clef.

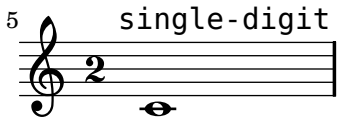
Mensural time signatures

There is limited support for mensuration signs (which are similar to, but not exactly the same as time signatures). The glyphs are hard-wired to particular time fractions. In other words, to get a particular mensuration sign with the `\time n/m` command, `n` and `m` have to be chosen according to the following table



Use the `style` property of grob `TimeSignature` to select ancient time signatures. Supported styles are `neomensural` and `mensural`. The above table uses the `neomensural` style. The following examples show the differences in style:





Voir aussi

Notation Reference: ⟨undefined⟩ [Time signature], page ⟨undefined⟩, gives a general introduction to the use of time signatures.

Problèmes connus et avertissements

Ratios of note durations do not change with the time signature. For example, the ratio of 1 breve = 3 semibreves (*tempus perfectum*) must be made by hand, by setting

```
breveTP = #(ly:make-duration -1 0 3 2)
...
{ c\breveTP f1 }
```

This sets `breveTP` to 3/2 times 2 = 3 times a whole note.

The `mensural68alt` and `neomensural68alt` symbols (alternate symbols for 6/8) are not addressable with `\time`. Use `\markup {\musicglyph #"timesig.mensural68alt" }` instead.

Mensural note heads

For ancient notation, a note head style other than the `default` style may be chosen. This is accomplished by setting the `style` property of the `NoteHead` object to `baroque`, `neomensural`, `mensural` or `petrucci`.

The `baroque` style differs from the `default` style by:

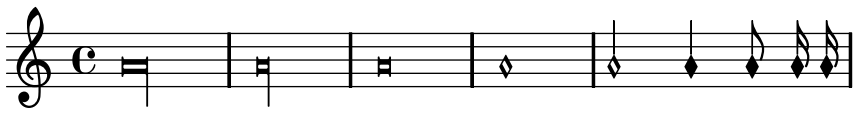
- Providing a `maxima` notehead, and
- Using a square shape for `\breve` note heads.

The `neomensural`, `mensural`, and `petrucci` styles differ from the `baroque` style by:

- Using rhomboidal heads for semibreves and all smaller durations, and
- Centering the stems on the note heads.

The following example demonstrates the `petrucci` style:

```
\set Score.skipBars = ##t
\autoBeamOff
\override NoteHead #'style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
```



Voir aussi

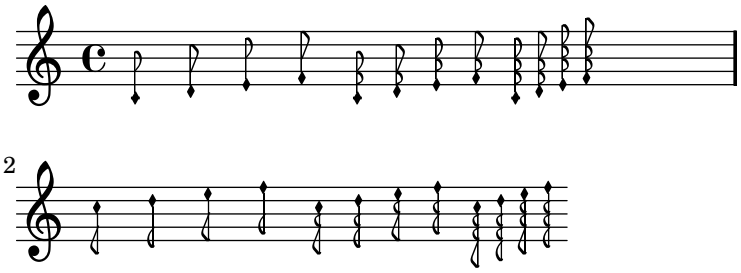
Notation Reference: Section A.7 [Note head styles], page 514, gives an overview of all available note head styles.

Mensural flags

Use the `flag-style` property of grob `Stem` to select ancient flags. Besides the `default` flag style, only the `mensural` style is supported.

```
\override Stem #'flag-style = #'mensural
\override Stem #'thickness = #1.0
\override NoteHead #'style = #'mensural
```

```
\autoBeamOff
c'8 d'8 e'8 f'8 c'16 d'16 e'16 f'16 c'32 d'32 e'32 f'32 s8
c''8 d''8 e''8 f''8 c''16 d''16 e''16 f''16 c''32 d''32 e''32 f''32
```



Note that the innermost flare of each mensural flag always is vertically aligned with a staff line.

There is no particular flag style for neo-mensural or Petrucci notation. There are no flags in Gregorian chant notation.

Problèmes connus et avertissements

The attachment of ancient flags to stems is slightly off.

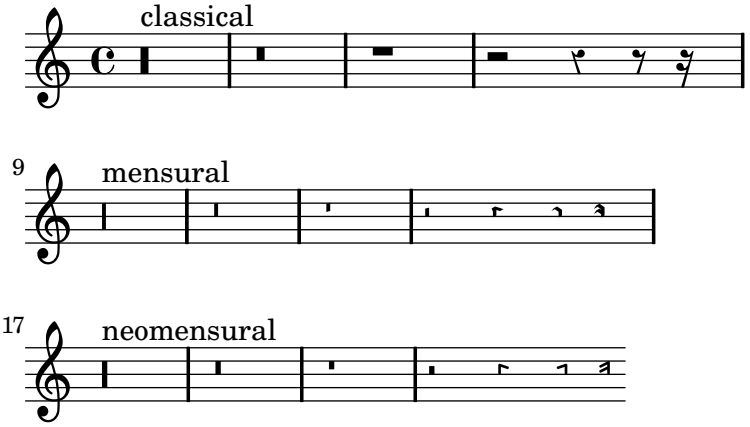
Vertically aligning each flag with a staff line assumes that stems always end either exactly on or exactly in the middle between two staff lines. This may not always be true when using advanced layout features of classical notation (which however are typically out of scope for mensural notation).

Mensural rests

Use the `style` property of grob `Rest` to select ancient rests. Supported styles are `classical`, `neomensural`, and `mensural`. `classical` differs from the `default` style only in that the quarter rest looks like a horizontally mirrored 8th rest. The `mensural` and the `neomensural` styles mimic the appearance of rests in manuscripts and prints up to the 16th century.

The following example demonstrates the `mensural` and `neomensural` styles:

```
\set Score.skipBars = ##t
\override Rest #'style = #'classical
r\longa^"classical" r\breve r1 r2 r4 r8 r16 s \break
\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
```



There are no 32th and 64th rests specifically for the mensural or neo-mensural style. Instead, the rests from the default style will be taken.

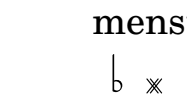
See Section “Ancient notation” dans *Morceaux choisis* for a chart of all rests.

Voir aussi

Notation Reference: ⟨undefined⟩ [Rests], page ⟨undefined⟩, gives a general introduction into the use of rests.

Mensural accidentals and key signatures

The `mensural` style provides a sharp and a flat sign different from the default style. If called for, the natural sign will be taken from the `vaticana` style.



The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental #'glyph-name-alist = #alteration-mensural-glyph-name-alist
```

Voir aussi

Notation Reference: ⟨undefined⟩ [Pitches], page ⟨undefined⟩, ⟨undefined⟩ [Accidentals], page ⟨undefined⟩, and ⟨undefined⟩ [Automatic accidentals], page ⟨undefined⟩, give a general introduction of the use of accidentals. ⟨undefined⟩ [Key signature], page ⟨undefined⟩, gives a general introduction of the use of key signatures.
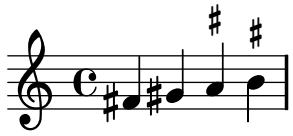
Internals Reference: Section “KeySignature” dans *Référence des propriétés internes*.

Annotational accidentals (*musica ficta*)

In European music from before about 1600, singers were expected to chromatically alter notes at their own initiative according to certain rules. This is called *musica ficta*. In modern transcriptions, these accidentals are usually printed over the note.

Support for such suggested accidentals is included, and can be switched on by setting `suggestAccidentals` to true.
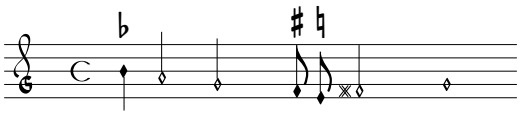
```
fis gis
\set suggestAccidentals = ##t
ais bis
```



This will treat *every* subsequent accidental as *musica ficta* until it is unset with `\set suggestAccidentals = ##f`. A more practical way is to use `\once \set suggestAccidentals = ##t`, which can even be defined as a convenient shorthand:

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative c''
  \new MensuralVoice  {
        \once \set suggestAccidentals = ##t
  bes4 a2 g2 \ficta fis8 \ficta e! fis2 g1
```

```
  }
}
```



Voir aussi

Internals Reference: Section “Accidental_engraver” dans *Référence des propriétés internes*, Section “AccidentalSuggestion” dans *Référence des propriétés internes*.

White mensural ligatures

There is limited support for white mensural ligatures.

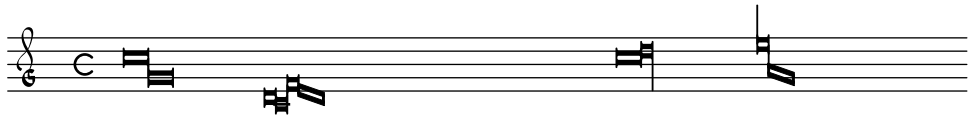
To engrave white mensural ligatures, in the layout block, replace the `Ligature_bracket_engraver` with the `Mensural_ligature_engraver` in the `Voice` context:

```
\layout {
  \context {
    \Voice
    \remove Ligature_bracket_engraver
    \consists Mensural_ligature_engraver
  }
}
```
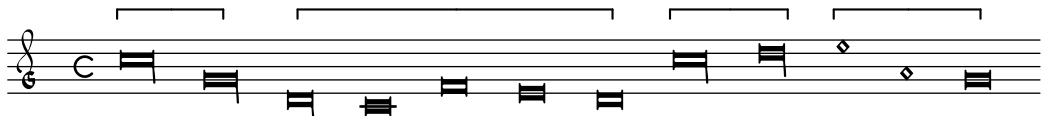
There is no additional input language to describe the shape of a white mensural ligature. The shape is rather determined solely from the pitch and duration of the enclosed notes. While this approach may take a new user a while to get accustomed to, it has the great advantage that the full musical information of the ligature is known internally. This is not only required for correct MIDI output, but also allows for automatic transcription of the ligatures.

For example,

```
\score {
  \transpose c c' {
    \set Score.timing = ##f
    \set Score.defaultBarType = "empty"
    \override NoteHead #'style = #'neomensural
    \override Staff.TimeSignature #'style = #'neomensural
    \clef "petrucci-g"
    \[ c'\maxima g \]
    \[ d\longa c\breve f e d \]
    \[ c'\maxima d'\longa \]
    \[ e'1 a g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove Ligature_bracket_engraver
      \consists Mensural_ligature_engraver
    }
  }
}
```



Without replacing `Ligature_bracket_engraver` with `Mensural_ligature_engraver`, the same music transcribes to the following



Problèmes connus et avertissements

Horizontal spacing of ligatures is poor.

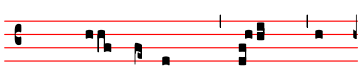
2.8.4 Typesetting Gregorian chant

When typesetting a piece in Gregorian chant notation, the `Vaticana_ligature_engraver` automatically selects the proper note heads, so there is no need to explicitly set the note head style. Still, the note head style can be set, e.g., to `vaticana_punctum` to produce punctum neumes. Similarly, the `Mensural_ligature_engraver` automatically assembles mensural ligatures. See [Ligatures], page 328, for how ligature engravers work.

Gregorian chant contexts

The predefined `VaticanaVoiceContext` and `VaticanaStaffContext` can be used to engrave a piece of Gregorian chant in the style of the Editio Vaticana. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

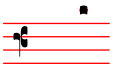
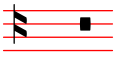
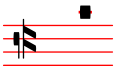
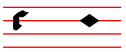
```
\include "gregorian.ly"
\score {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
  >>
}
```





Gregorian clefs

The following table shows all Gregorian clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. Still, you can manually force a clef glyph to be typeset on an arbitrary line, as described in ⟨undefined⟩ [Clef], page ⟨undefined⟩. The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Description	Supported Clefs	Example
Editio Vaticana style do clef	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Editio Vaticana style fa clef	`vaticana-fa1`, `vaticana-fa2`	
Editio Medicaea style do clef	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	
Editio Medicaea style fa clef	`medicaea-fa1`, `medicaea-fa2`	
hufnagel style do clef	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
hufnagel style fa clef	`hufnagel-fa1`, `hufnagel-fa2`	
hufnagel style combined do/fa clef	`hufnagel-do-fa`	

Voir aussi

Notation Reference: ⟨undefined⟩ [Clef], page ⟨undefined⟩.

Gregorian accidentals and key signatures

Accidentals for the three different Gregorian styles are available:

vaticana medicaea hufnagel



As shown, not all accidentals are supported by each style. When trying to access an unsupported accidental, LilyPond will switch to a different style.

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental #'glyph-name-alist = #alteration-mensural-glyph-
name-alist
```

Voir aussi

Notation Reference: ⟨undefined⟩ [Pitches], page ⟨undefined⟩, ⟨undefined⟩ [Accidentals], page ⟨undefined⟩, and ⟨undefined⟩ [Automatic accidentals], page ⟨undefined⟩, give a general introduction of the use of accidentals. ⟨undefined⟩ [Key signature], page ⟨undefined⟩, gives a general introduction of the use of key signatures.

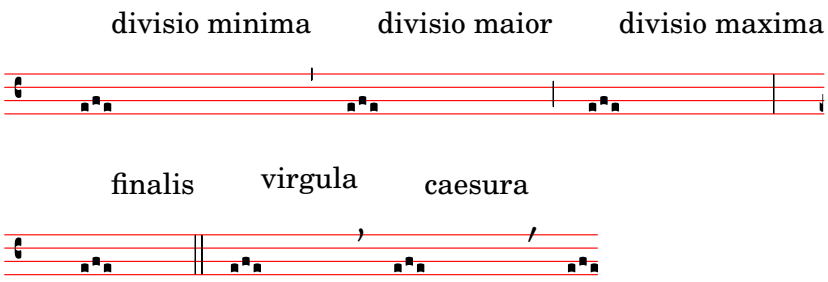
Internals Reference: Section “KeySignature” dans *Référence des propriétés internes*.

Divisiones

There are no rests in Gregorian chant notation; instead, it uses [Divisiones], page 338.

A *divisio* (plural: *divisiones*; Latin word for ‘division’) is a staff context symbol that is used to indicate the phrase and section structure of Gregorian music. The musical meaning of *divisio minima*, *divisio maior*, and *divisio maxima* can be characterized as short, medium, and long pause, somewhat like the breathmarks from ⟨undefined⟩ [Breath marks], page ⟨undefined⟩. The *finalis* sign not only marks the end of a chant, but is also frequently used within a single antiphonal/responsorial chant to mark the end of each section.

To use divisiones, include the file ‘`gregorian.ly`’. It contains definitions that you can apply by just inserting `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, and `\finalis` at proper places in the input. Some editions use *virgula* or *caesura* instead of divisio minima. Therefore, ‘`gregorian.ly`’ also defines `\virgula` and `\caesura`



Commandes prédéfinies

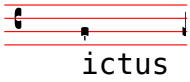
`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

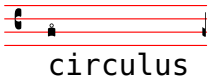
Gregorian articulation signs

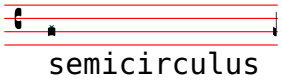
In addition to the standard articulation signs described in section ⟨undefined⟩ [Articulations and ornamentations], page ⟨undefined⟩, articulation signs specifically designed for use with notation in *Editio Vaticana* style are provided.

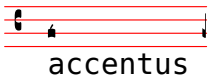
```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript #'font-family = #'typewriter
    \override TextScript #'font-shape = #'upright
    \override Script #'padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
```

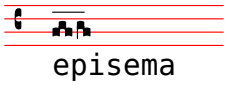
```
  }
}
```











Snippets: Section “Ancient notation” dans *Morceaux choisis*.

Internals Reference: Section “Episema” dans *Référence des propriétés internes*, Section “EpisemaEvent” dans *Référence des propriétés internes*, Section “Episema_engraver” dans *Référence des propriétés internes*, Section “Script” dans *Référence des propriétés internes*, Section “ScriptEvent” dans *Référence des propriétés internes*, Section “Script_engraver” dans *Référence des propriétés internes*.
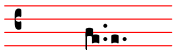
Problèmes connus et avertissements

Some articulations are vertically placed too closely to the corresponding note heads.

Augmentum dots (*morae*)

Augmentum dots, also called *morae*, are added with the music function `\augmentum`. Note that `\augmentum` is implemented as a unary music function rather than as head prefix. It applies to the immediately following music expression only. That is, `\augmentum \virga c` will have no visible effect. Instead, say `\virga \augmentum c` or `\augmentum {\virga c}`. Also note that you can say `\augmentum {a g}` as a shortcut for `\augmentum a \augmentum g`.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Voir aussi

Notation Reference: ⟨undefined⟩ [Breath marks], page ⟨undefined⟩.

Internals Reference: Section “BreathingSign” dans *Référence des propriétés internes*.

Snippets: Section “Ancient notation” dans *Morceaux choisis*.

Gregorian square neume ligatures

There is limited support for Gregorian square neumes notation (following the style of the Editio Vaticana). Core ligatures can already be typeset, but essential issues for serious typesetting are still lacking, such as (among others) horizontal alignment of multiple ligatures, lyrics alignment, and proper handling of accidentals.

The support for Gregorian neumes is enabled by `\include`ing "gregorian.ly" at the beginning of the file. This makes available a number of extra commands to produce the neume symbols used in plainchant notation.

Note heads can be *modified* and/or *joined*.

- The shape of the note head can be modified by *prefixing* the note name with any of the following commands: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Ligatures, properly speaking (i.e. notes joined together), are produced by placing one of the joining commands `\pes` or `\flexa`, for upwards and downwards movement, respectively, *between* the notes to be joined.

A note name without any qualifiers will produce a *punctum*. All other neumes, including the single-note neumes with a different shape such as the *virga*, are in principle considered as ligatures and should therefore be placed between `\[...\]`.

Single-note neumes:

- The *punctum* is the basic note shape (in the *Vaticana* style: a square with some curvation for typographical finesse). In addition to the regular *punctum*, there is also the oblique *punctum inclinatum*, produced with the prefix `\inclinatum`. The regular *punctum* can be modified with `\cavum`, which produces a hollow note, and `\linea`, which draws vertical lines on either side of the note.
- The *virga* has a descending stem on the right side. It is produced by the modifier `\virga`.

Ligatures

Unlike most other neumes notation systems, the typographical appearance of ligatures is not directly dictated by the input commands, but follows certain conventions dependent on musical meaning. For example, a three-note ligature with the musical shape low-high-low, such as `\[ a \pes b \flexa g \]`, produces a Torculus consisting of three Punctum heads, while the shape high-low-high, such as `\[ a \flexa g \pes b \]`, produces a Porrectus with a curved flexa shape and only a single Punctum head. There is no command to explicitly typeset the curved flexa shape; the decision of when to typeset a curved flexa shape is based on the musical input. The idea of this approach is to separate the musical aspects of the input from the notation style of the output. This way, the same input can be reused to typeset the same music in a different style of Gregorian chant notation.

Liquescent neumes

Another main category of notes in Gregorian chant is the so-called liquescent neumes. They are used under certain circumstances at the end of a syllable which ends in a 'liquescent' letter, i.e. the sounding consonants that can hold a tone (the nasals, l, r, v, j, and their diphtong equivalents). Thus, the liquescent neumes are never used alone (although some of them can be produced), and they always fall at the end of a ligature.

Liquescent neumes are represented graphically in two different, more or less interchangeable ways: with a smaller note or by 'twisting' the main note upwards or downwards. The first is produced by making a regular `pes` or `flexa` and modifying the shape of the second note: `\[ a \pes \deminutum b \]` , the second by modifying the shape of a single-note neume with `\auctum` and one of the direction markers `\descendens` or `\ascendens`, e.g., `\[ \auctum \descendens a \]` .

Special signs

A third category of signs is made up of a small number of signs with a special meaning (which, incidentally, in most cases is only vaguely known): the *quilisma*, the *oriscus*, and the *strophicus*. These are all produced by prefixing a note name with the corresponding modifier, `\quilisma`, `\oriscus`, or `\stropha`.

Virtually, within the ligature delimiters `\[` and `\]`, any number of heads may be accumulated to form a single ligature, and head prefixes like `\pes`, `\flexa`, `\virga`, `\inclinatum`, etc. may be mixed in as desired. The use of the set of rules that underlies the construction of the ligatures in the above table is accordingly extrapolated. This way, infinitely many different ligatures can be created.

Note that the use of these signs in the music itself follows certain rules, which are not checked by LilyPond. E.g., the *quilisma* is always the middle note of an ascending ligature, and usually falls on a half-tone step, but it is perfectly possible, although incorrect, to make a single-note quilisma.

In addition to the note signs, gregorian.ly also defines the commands `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ`, and `\IIJ`, that will produce the corresponding characters, e.g., for use in lyrics, as section markers, etc. These commands use special unicode characters and will only work if a font is used which supports them.

The following table shows a limited, but still representative pool of Gregorian ligatures, together with the code fragments that produce the ligatures. The table is based on the extended neumes table of the 2nd volume of the Antiphonale Romanum (*Liber Hymnarius*), published 1983 by the monks of Solesmes. The first column gives the name of the ligature, with the main form in boldface and the liquescent forms in italics. The third column shows the code fragment that produces this ligature, using `g`, `a`, and `b` as example pitches.

Single-note neums

Basic and *Liquescent* forms	**Output**	**LilyPond code**
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



Two-note ligatures

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

```
\[ \deminutum g \pes \auctum
\descendens b \]
```



Multi-note ligatures

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

```
\[ g \pes a \pes \deminutum b \]
```



Special Signs

Quilisma

```
\[ g \pes \quilisma a \pes b \]
```



Quilisma Pes Auctus Descendens

```
\[ \quilisma g \pes \auctum
\descendens b \]
```



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

The following head prefixes are supported: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Head prefixes can be accumulated, though restrictions apply. For example, either `\descendens` or `\ascendens` can be applied to a head, but not both to the same head.

Two adjacent heads can be tied together with the `\pes` and `\flexa` infix commands for a rising and falling line of melody, respectively.

Use the unary music function `\augmentum` to add augmentum dots.

Problèmes connus et avertissements

When an `\augmentum` dot appears at the end of the last staff within a ligature, it is sometimes vertically placed wrong. As a workaround, add an additional skip note (e.g., `s8`) as last note of the staff.

`\augmentum` should be implemented as a head prefix rather than a unary music function, such that `\augmentum` can be intermixed with head prefixes in arbitrary order.

2.8.5 Working with ancient music—scenarios and solutions

Working with ancient music frequently involves particular tasks which differ considerably from the modern notation for which LilyPond is designed. In the rest of this section, a number of typical scenarios are outlined, with suggestions of solutions. These involve:

- how to make incipits (i.e. prefatory material to indicate what the original has looked like) to modern transcriptions of mensural music;
- how to achieve the *Mensurstriche* layout frequently used for modern transcriptions of polyphonic music;
- how to transcribe Gregorian chant in modern notation;

- how to generate both ancient and modern notation from the same source.

Incipits

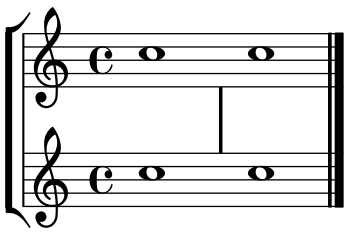
TBC

Mensurstriche layout

Mensurstriche ('mensuration lines') is the accepted term for bar lines that are drawn between the staves of a system but not through the staves themselves. It is a common way to preserve the rhythmic appearance of the original, i.e. not having to break syncopated notes at bar lines, while still providing the orientation aids that bar lines give.

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a `StaffGroup` instead of a `ChoirStaff`. The bar line on staves is blanked out by setting the `transparent` property.

```
global = {
  \override Staff.BarLine #'transparent = ##t
  s1 s
  % the final bar line is not interrupted
  \revert Staff.BarLine #'transparent
  \bar "|."
}
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Transcribing Gregorian chant

Gregorian chant can be transcribed into modern notation with a number of simple tweaks.

Stems. Stems can be left out altogether by `\remove`-ing the `Stem_engraver` from the Voice context:

```
\layout {
  ...
  \context {
    \Voice
      \remove "Stem_engraver"
  }
}
```

However, in some transcription styles, stems are used occasionally, for example to indicate the transition from a single-tone recitative to a fixed melodic gesture. In these cases, one can use either `\override Stem #'transparent = ##t` or `\override Stem #'length = #0` instead, and

restore the stem when needed with the corresponding `\once \override Stem #'transparent = ##f` (see example below).

Timing. For unmetered chant, there are several alternatives.

The Time_signature_engraver can be removed from the Staff context without any negative side effects. The alternative, to make it transparent, will leave an empty space in the score, since the invisible signature will still take up space.
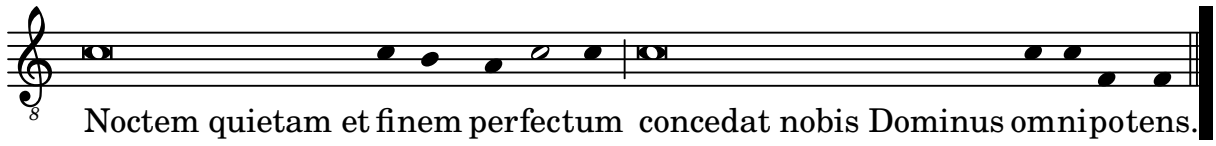
In many cases, `\set Score.timing = ##f` will give good results. Another alternative is to use `\CadenzaOn` and `\CadenzaOff`.

To remove the barlines, the radical approach is to `\remove` the Bar_engraver from the Staff context. Again, one may want to use `\override BarLine #'transparent = ##t` instead, if an occasional barline is wanted.

A common type of transcription is recitativic chant where the repeated notes are indicated with a single breve. The text to the recitation tone can be dealt with in two different ways: either set as a single, left-aligned syllable:

```
\include "gregorian.ly"
chant = \relative c' {
  \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"
      \override Stem #'transparent = ##t
    }
  }
}
```
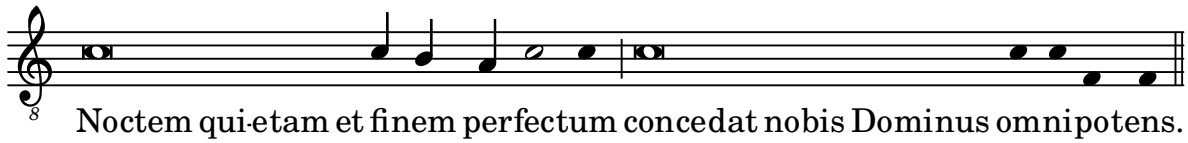


This works fine, as long as the text doesn't span a line break. If that is the case, an alternative is to add hidden notes to the score, here in combination with changing stem visibility:

```
\include "gregorian.ly"
chant = \relative c' {
```

```
  \clef "G_8"
  \set Score.timing = ##f
  c\breve \override NoteHead #'transparent = ##t  c c c c c
  \revert NoteHead #'transparent
  \override Stem #'transparent = ##f \stemUp c4 b4 a
  \override Stem #'transparent = ##t c2 c4  \divisioMaior
  c\breve \override NoteHead #'transparent = ##t c c c c c c c
  \revert NoteHead #'transparent 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"
      \override BarLine #'transparent = ##t
      \override Stem #'transparent = ##t
    }
  }
}
```



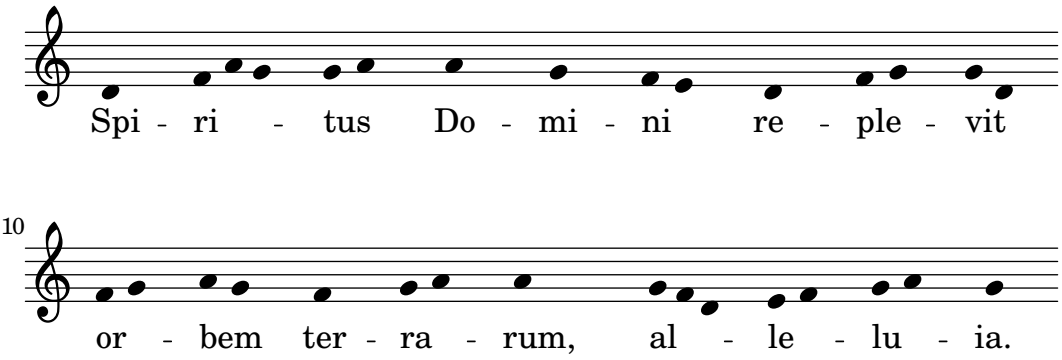
Another common situation is transcription of neumatic or melismatic chants, i.e. chants with a varying number of notes to each syllable. In this case, one would want to set the syllable groups clearly apart, usually also the subdivisions of a longer melisma. One way to achieve this is to use a fixed `\time`, e.g., 1/4, and let each syllable or note group fill one of these measures, with the help of tuplets or shorter durations. If the barlines and all other rhythmical indications are made transparent, and the space around the barlines is increased, this will give a fairly good representation in modern notation of the original.

To avoid that syllables of different width (such as « -ri » and « -rum ») spread the syllable note groups unevenly apart, the `#'X-extent` property of the `LyricText` object may be set to a fixed value. Another, more cumbersome way would be to add the syllables as `\markup` elements. If further adjustments are necessary, this can be easily done with `s` 'notes'.

```
spiritus = \relative c' {
  \time 1/4
  \override Lyrics.LyricText #'X-extent  = #'(0 . 3)
  d4 \times 2/3 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4  s
  \times 2/3 { 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)
      \override Stem #'transparent = ##t
      \override Beam #'transparent = ##t
      \override BarLine #'transparent = ##t
      \override TupletNumber #'transparent = ##t
    }
  }
}
```



Ancient and modern from one source

TBC

Editorial markings

TBC

2.9 World music

The purpose of this section is to highlight musical notation issues that are relevant to traditions outside the Western tradition.

2.9.1 Common notation for non-Western music

This section discusses how to enter and print music scores that do not belong to the Western classical tradition, also referred to as *Common Practice Period*.

Extending notation and tuning systems

Standard classical notation (also known as *Common Practice Period* notation) is commonly used in all sorts of music, not limited to 'classical' Western music. This notation is discussed in ⟨undefined⟩ [Writing pitches], page ⟨undefined⟩, and the various note names that may be used are explained in ⟨undefined⟩ [Note names in other languages], page ⟨undefined⟩.

However, many types of non-Western music (and some types of Western folk and traditional music) employ alternative or extended tuning systems that do not fit readily into standard classical notation.

In some cases standard notation is still used, with the pitch differences being implicit. For example, *Arabic music* is notated with standard semitone and quarter-tone accidentals, with the precise pitch alterations being determined by context. Italian note names are typically used, while the init file '`arabic.ly`' provides a suitable set of macros and definitions extending the standard notation. For more details, see Section 2.9.2 [Arabic music], page 352.

Other types of music require extended or unique notations. *Turkish classical music* or Ottoman music, for example, employs melodic forms known as *makamlar*, whose intervals are based on 1/9 divisions of the whole tone. Standard Western staff notes are still used, but with special accidentals unique to Turkish music, that are defined in the file '`makam.ly`'. For further information on Turkish classical music and makamlar, see Section 2.9.3 [Turkish classical music], page 357.

To locate init files such as '`arabic.ly`' or '`makam.ly`' on your system, see Section "Other sources of information" dans *Manuel d'initiation*.
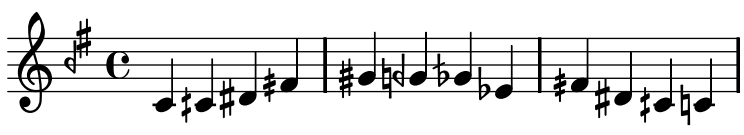
Morceaux choisis

Makam example

Makam is a type of melody from Turkey using 1/9th-tone microtonal alterations. Consult the initialization file '`ly/makam.ly`' for details of pitch names and alterations.

```
% Initialize makam settings
\include "makam.ly"

\relative c' {
  \set Staff.keySignature = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



Voir aussi

Music Glossary: Section "Common Practice Period" dans *Glossaire*, Section "makamlar" dans *Glossaire*.

Learning Manual: Section "Other sources of information" dans *Manuel d'initiation*.

Notation Reference: ⟨undefined⟩ [Writing pitches], page ⟨undefined⟩, ⟨undefined⟩ [Note names in other languages], page ⟨undefined⟩, Section 2.9.2 [Arabic music], page 352, Section 2.9.3 [Turkish classical music], page 357.

2.9.2 Arabic music

This section highlights issues that are relevant to notating Arabic music.

References for Arabic music

Arabic music so far has been mainly an oral tradition. When music is transcribed, it is usually in a sketch format, on which performers are expected to improvise significantly. Increasingly, Western notation, with a few variations, is adopted in order to communicate and preserve Arabic music.

Some elements of Western musical notation such as the transcription of chords or independent parts, are not required to typeset the more traditional Arabic pieces. There are however some different issues, such as the need to indicate medium intervals that are somewhere between a semi-tone and a tone, in addition to the minor and major intervals that are used in Western music. There is also the need to group and indicate a large number of different maqams (modes) that are part of Arabic music.

In general, Arabic music notation does not attempt to precisely indicate microtonal elements that are present in musical practice.

Several issues that are relevant to Arabic music are covered elsewhere:

- Note names and accidentals (including quarter tones) can be tailored as discussed in Section 2.9.1 [Common notation for non-Western music], page 350.
- Additional key signatures can also be tailored as described in ⟨undefined⟩ [Key signature], page ⟨undefined⟩.
- Complex time signatures may require that notes be grouped manually as described in ⟨undefined⟩ [Manual beams], page ⟨undefined⟩.
- *Takasim* which are rhythmically free improvisations may be written down omitting bar lines as described in ⟨undefined⟩ [Unmetered music], page ⟨undefined⟩.

Voir aussi

Notation Reference: Section 2.9.1 [Common notation for non-Western music], page 350, ⟨undefined⟩ [Key signature], page ⟨undefined⟩, ⟨undefined⟩ [Manual beams], page ⟨undefined⟩.
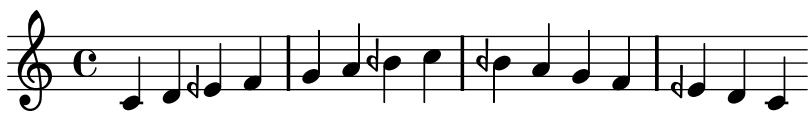
Snippets: Section “World music” dans *Morceaux choisis*.

Arabic note names

The more traditional Arabic note names can be quite long and are not suitable for the purpose of music writing, so they are not used. English note names are not very familiar in Arabic music education, so Italian or Solfege note names (`do, re, mi, fa, sol, la, si`) are used instead; modifiers (accidentals) can also be used. Italian note names and accidentals are explained in ⟨undefined⟩ [Note names in other languages], page ⟨undefined⟩; the use of standard Western notation to notate non-Western music is discussed in Section 2.9.1 [Common notation for non-Western music], page 350.
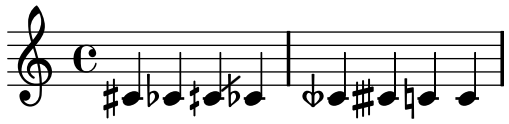
For example, this is how the Arabic *rast* scale can be notated:

```
\include "arabic.ly"
\relative do' {
  do re misb fa sol la sisb do sisb la sol fa misb re do
}
```



The symbol for semi-flat does not match the symbol which is used in Arabic notation. The `\dwn` symbol defined in `arabic.ly` may be used preceding a flat symbol as a work around if it is important to use the specific Arabic semi-flat symbol. The appearance of the semi-flat symbol in the key signature cannot be altered by using this method.

```
\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
  dod dob dosd \dwn dob dobsb dodsd do do
}
```



Voir aussi

Notation Reference: ⟨undefined⟩ [Note names in other languages], page ⟨undefined⟩, Section 2.9.1 [Common notation for non-Western music], page 350.

Snippets: Section “World music” dans *Morceaux choisis*.

Arabic key signatures

In addition to the minor and major key signatures, the following key signatures are defined in `arabic.ly`: *bayati*, *rast*, *sikah*, *iraq*, and *kurd*. These key signatures define a small number of maqam groups rather than the large number of maqams that are in common use.

In general, a maqam uses the key signature of its group, or a neighbouring group, and varying accidentals are marked throughout the music.

For example to indicate the key signature of a maqam muhayer piece:

```
\key re \bayati
```

Here *re* is the default pitch of the muhayer maqam, and *bayati* is the name of the base maqam in the group.

While the key signature indicates the group, it is common for the title to indicate the more specific maqam, so in this example, the name of maqam muhayer should appear in the title.

Other maqams in the same bayati group, as shown in the table below: (bayati, hussaini, saba, and ushaq) can be indicated in the same way. These are all variations of the base and most common maqam in the group, which is bayati. They usually differ from the base maqam in their upper tetrachords, or certain flow details that don’t change their fundamental nature, as siblings.

The other maqam in the same group (Nawa) is related to bayati by modulation which is indicated in the table in parenthesis for those maqams that are modulations of their base maqam. Arabic maqams admit of only limited modulations, due to the nature of Arabic musical instruments. Nawa can be indicated as follows:

```
\key sol \bayati
```

In Arabic music, the same term such as bayati that is used to indicate a maqam group, is also a maqam which is usually the most important in the group, and can also be thought of as a base maqam.

Here is one suggested grouping that maps the more common maqams to key signatures:

maqam group **key** **finalis** **Other maqmas in group (finalis)**

ajam	major	sib	jaharka (fa)
bayati	bayati	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	kurd	re	shahnaz, shad arban (sol), hijazkar (do)
iraq	iraq	sisb	-
kurd	kurd	re	hijazkar kurd (do)
nahawand	minor	do	busalik (re), farah faza (sol)
nakriz	minor	do	nawa athar, hisar (re)
rast	rast	do	mahur, yakah (sol)
sikah	sikah	misb	huzam

Morceaux choisis

Non-traditional key signatures

The commonly used `\key` command sets the `keySignature` property, in the `Staff` context.
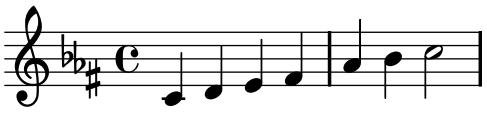
To create non-standard key signatures, set this property directly. The format of this command is a list:

`\set Staff.keySignature = #`(((octave . step) . alter) ((octave . step) . alter) ...)` where, for each element in the list, `octave` specifies the octave (0 being the octave from middle C to the B above), `step` specifies the note within the octave (0 means C and 6 means B), and `alter` is `,SHARP ,FLAT ,DOUBLE-SHARP` etc. (Note the leading comma.)

Alternatively, for each item in the list, using the more concise format `(step . alter)` specifies that the same alteration should hold in all octaves.

Here is an example of a possible key signature for generating a whole-tone scale:

```
\relative c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
                              ((0 . 5) . ,FLAT)
                              ((0 . 3) . ,SHARP))
  c4 d e fis
  aes4 bes c2
}
```



Voir aussi

Music Glossary: Section “maqam” dans *Glossaire*, Section “bayati” dans *Glossaire*, Section “rast” dans *Glossaire*, Section “sikah” dans *Glossaire*, Section “iraq” dans *Glossaire*, Section “kurd” dans *Glossaire*.

Notation Reference: ⟨undefined⟩ [Key signature], page ⟨undefined⟩.

Learning Manual: Section “Accidentals and key signatures” dans *Manuel d’initiation*.

Internals Reference: Section “KeySignature” dans *Référence des propriétés internes*.

Snippets: Section “World music” dans *Morceaux choisis*, Section “Pitches” dans *Morceaux choisis*.

Arabic time signatures

Some Arabic and Turkish music classical forms such as *Semai* use unusual time signatures such as 10/8. This may lead to an automatic grouping of notes that is quite different from existing typeset music, where notes may not be grouped on the beat, but in a manner that is difficult to

match by adjusting automatic beaming. The alternative is to switch off automatic beaming and beam the notes manually. Even if a match to existing typeset music is not required, it may still be desirable to adjust the automatic beaming behaviour and/or use compound time signatures.
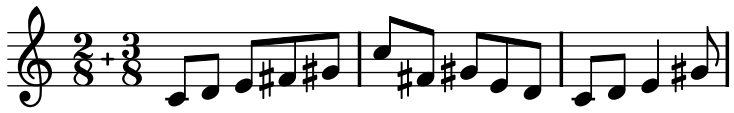
Morceaux choisis

Compound time signatures

Odd 20th century time signatures (such as "5/8") can often be played as compound time signatures (e.g. "3/8 + 2/8"), which combine two or more inequal metrics. LilyPond can make such music quite easy to read and play, by explicitly printing the compound time signatures and adapting the automatic beaming behavior. (Graphic measure grouping indications can also be added; see the appropriate snippet in this database.)

```
#(define ((compound-time one two num) grob)
   (grob-interpret-markup grob
                          (markup #:override '(baseline-skip . 0) #:number
                                  (#:line ((#:column (one num))
                                           #:vcenter "+"
                                           (#:column (two num)))))))

\relative c' {
  \override Staff.TimeSignature #'stencil = #(compound-time "2" "3" "8")
  \time 5/8
  \set Staff.beatStructure = #'(2 3)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```

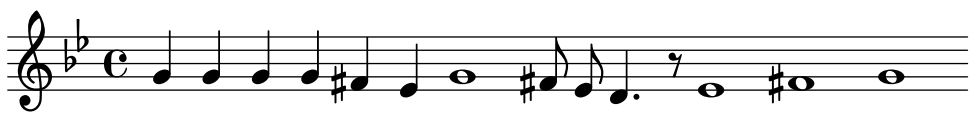


Arabic improvisation

For improvisations or taqasim which are temporarily free, the time signature can be omitted and `\cadenzaOn` can be used. Adjusting the accidental style might be required, since the absence of bar lines will cause the accidental to be marked only once. Here is an example of what could be the start of a hijaz improvisation:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  #(set-accidental-style 'forget)
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Voir aussi

Music Glossary: Section “semai” dans *Glossaire*, Section “taqasim” dans *Glossaire*.

Notation Reference: ⟨undefined⟩ [Manual beams], page ⟨undefined⟩, ⟨undefined⟩ [Automatic beams], page ⟨undefined⟩, ⟨undefined⟩ [Unmetered music], page ⟨undefined⟩, ⟨undefined⟩ [Automatic accidentals], page ⟨undefined⟩, ⟨undefined⟩ [Setting automatic beam behavior], page ⟨undefined⟩, ⟨undefined⟩ [Time signature], page ⟨undefined⟩.
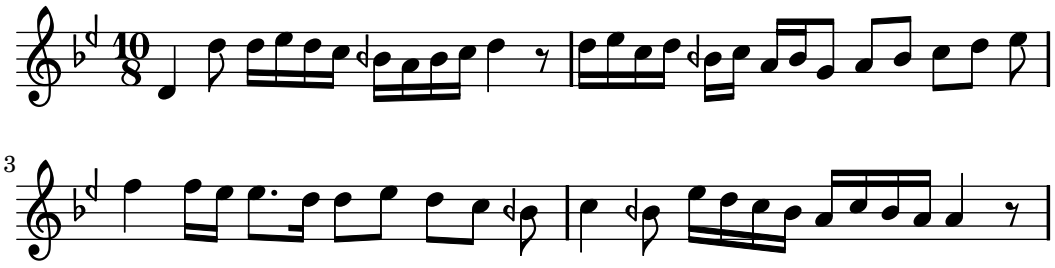
Snippets: Section “World music” dans *Morceaux choisis*.

Arabic music example

Here is a template that also uses the start of a Turkish *Semai* that is familiar in Arabic music education in order to illustrate some of the peculiarities of Arabic music notation, such as medium intervals and unusual modes that are discussed in this section.

```
\include "arabic.ly"
\score {
  \relative re' {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re4 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
  }
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
}
```



Voir aussi

Snippets: Section “World music” dans *Morceaux choisis*.

Further reading for Arabic music

1. *The music of the Arabs* by Habib Hassan Touma [Amadeus Press, 1996], contains a discussion of maqams and their method of groupings.

   There are also various web sites that explain maqams and some provide audio examples such as :

   - http://www.maqamworld.com/
   - http://www.turath.org/

There are some variations in the details of how maqams are grouped, despite agreement on the criteria of grouping maqams that are related through common lower tetra chords, or through modulation.

2. There is not a complete consistency, sometimes even in the same text on how key signatures for particular maqams should be specified. It is common, however, to use a key signature per group, rather than a different key signature for each different maqam.

   Method books by the following authors for the *Oud*, the Arabic lute, contain examples of mainly Turkish and Arabic compositions.

   - Charbel Rouhana
   - George Farah
   - Ibrahim Ali Darwish Al-masri

2.9.3 Turkish classical music

This section highlights issues that are relevant to notating Turkish classical music.

References for Turkish classical music

Turkish classical music developed in the Ottoman Empire in a period roughly contemporaneous with classical music in Europe, and has continued on into the 20th and 21st centuries as a vibrant and distinct tradition with its own compositional forms, theory and performance styles. Among its striking features is the use of microtonal intervals based on 'commas' of 1/9 of a tone, from which are constructed the melodic forms known as *makam* (plural *makamlar*).

Some issues relevant to Turkish classical music are covered elsewhere:

- Special note names and accidentals are explained in Section 2.9.1 [Common notation for non-Western music], page 350.

Turkish note names

Pitches in Turkish classical music traditionally have unique names, and the basis of pitch on 1/9-tone divisions means makamlar employ a completely different set of intervals from Western scales and modes: *koma* (1/9 of a tone), *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (a whole tone) and *artık ikili* (12/9 or 13/9 of a tone).

From a modern notational point of view it is convenient to use the standard Western staff notes (c, d, e, ...) with special accidentals that raise or lower notes by intervals of 1/9, 4/9, 5/9 and 8/9 of a tone. These accidentals are defined in the file '`makam.ly`'.

The following table lists:

- the name of these special accidentals,
- the accidental suffix that must be added to notes,
- and their pitch alteration as a fraction of one whole tone.

Accidental name	**suffix**	**pitch alteration**
büyük mücenneb (sharp)	-bm	+8/9
kücük mücenneb (sharp)	-k	+5/9
bakiye (sharp)	-b	+4/9
koma (sharp)	-c	+1/9
koma (flat)	-fc	-1/9
bakiye (flat)	-fb	-4/9
kücük mücenneb (flat)	-fk	-5/9

büyük mücenneb (flat) -fbm -8/9

For a more general explanation of non-Western music notation, see Section 2.9.1 [Common notation for non-Western music], page 350.

Voir aussi

Music Glossary: Section “makam” dans *Glossaire*, Section “makamlar” dans *Glossaire*.

Notation Reference: Section 2.9.1 [Common notation for non-Western music], page 350.

3 General input and output

This section deals with general LilyPond input and output issues, rather than specific notation.

3.1 Input structure

The main format of input for LilyPond are text files. By convention, these files end with `.ly`.

3.1.1 Structure of a score

A `\score` block must contain a single music expression delimited by curly brackets:
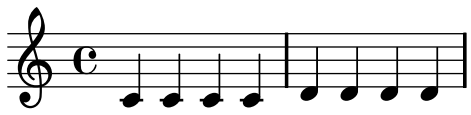
```
\score {
...
}
```

Note : There must be **only one** outer music expression in a `\score` block, and it **must** be surrounded by curly brackets.

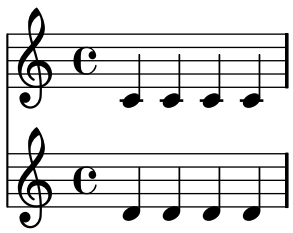
This single music expression may be of any size, and may contain other music expressions to any complexity. All of these examples are music expressions:

```
{ 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 { \oboe }
    >>
    \new StaffGroup <<
      \new Staff { \violinI }
      \new Staff { \violinII }
    >>
  >>
```

```
}
```

Comments are one exception to this general rule. (For others see Section 3.1.5 [File structure], page 363.) Both single-line comments and comments delimited by `%{ .. %}` may be placed anywhere within an input file. They may be placed inside or outside a `\score` block, and inside or outside the single music expression within a `\score` block.

Remember that even in a file containing only a `\score` block, it is implicitly enclosed in a \book block. A \book block in a source file produces at least one output file, and by default the name of the output file produced is derived from the name of the input file, so '`fandangoforelephants.ly`' will produce '`fandangoforelephants.pdf`'.

(For more details about `\book` blocks, see Section 3.1.2 [Multiple scores in a book], page 360, Section 3.1.3 [Multiple output files from one input file], page 361 Section 3.1.5 [File structure], page 363.)

Voir aussi

Learning Manual: Section "Working on input files" dans *Manuel d'initiation*, Section "Music expressions explained" dans *Manuel d'initiation*, Section "Score is a (single) compound musical expression" dans *Manuel d'initiation*.

3.1.2 Multiple scores in a book

A document may contain multiple pieces of music and text. Examples of these are an etude book, or an orchestral part with multiple movements. Each movement is entered with a `\score` block,

```
\score {
  ..music..
}
```

and texts are entered with a `\markup` block,

```
\markup {
  ..text..
}
```

All the movements and texts which appear in the same `.ly` file will normally be typeset in the form of a single output file.

```
\score {
  ..
}
\markup {
  ..
}
\score {
  ..
}
```

One important exception is within lilypond-book documents, where you explicitly have to add a `\book` block, otherwise only the first `\score` or `\markup` will appear in the output.

The header for each piece of music can be put inside the `\score` block. The `piece` name from the header will be printed before each movement. The title for the entire book can be put inside the `\book`, but if it is not present, the `\header` which is at the top of the file is inserted.

```
\header {
  title = "Eight miniatures"
  composer = "Igor Stravinsky"
}
```

```
\score {
  ...
  \header { piece = "Romanze" }
}
\markup {
   ..text of second verse..
}
\markup {
   ..text of third verse..
}
\score {
  ...
  \header { piece = "Menuetto" }
}
```

Pieces of music may be grouped into book parts using `\bookpart` blocks. Book parts are separated by a page break, and can start with a title, like the book itself, by specifying a `\header` block.

```
\bookpart {
  \header {
    title = "Book title"
    subtitle = "First part"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Second part"
  }
  \score { ... }
  ...
}
```

3.1.3 Multiple output files from one input file

If you want multiple output files from the same .ly file, then you can add multiple `\book` blocks, where each such \book block will result in a separate output file. If you do not specify any `\book` block in the input file, LilyPond will implicitly treat the whole file as a single \book block, see Section 3.1.5 [File structure], page 363.

When producing multiple files from a single source file, Lilypond ensures that none of the output files from any `\book` block overwrites the output file produced by a preceding `\book` from the same input file.

It does this by adding a suffix to the output name for each `\book` which uses the default output file name derived from the input source file.

The default behaviour is to append a version-number suffix for each name which may clash, so

```
\book {
  \score { ... }
  \layout { ... }
}
\book {
```

```
  \score { ... }
  \layout { ... }
}
\book {
  \score { ... }
  \layout { ... }
}
```

in source file ‘`eightminiatures.ly`’ will produce

- ‘`eightminiatures.pdf`’,
- ‘`eightminiatures-1.pdf`’ and
- ‘`eightminiatures-2.pdf`’.

3.1.4 Output file names

Lilypond provides facilities to allow you to control what file names are used by the various back-ends when producing output files.

In the previous section, we saw how Lilypond prevents name-clashes when producing several ouputs from a single source file. You also have the ability to specify your own suffixes for each `\book` block, so for example you can produce files called ‘`eightminiatures-Romanze.pdf`’, ‘`eightminiatures-Menuetto.pdf`’ and ‘`eightminiatures-Nocturne.pdf`’ by adding a `\bookOutputSuffix` declaration inside each `\book` block.

```
\book {
  \bookOutputSuffix "Romanze"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputSuffix "Menuetto"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \layout { ... }
}
```

You can also specify a different output filename for `book` block, by using `\bookOutputName` declarations

```
\book {
  \bookOutputName "Romanze"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputName "Menuetto"
  \score { ... }
  \layout { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
```

```
  \layout { ... }
}
```

The file above will produce these output files:

- 'Romanze.pdf',
- 'Menuetto.pdf' and
- 'Nocturne.pdf'.

3.1.5 File structure

A .ly file may contain any number of toplevel expressions, where a toplevel expression is one of the following:

- An output definition, such as \paper, \midi, and \layout. Such a definition at the toplevel changes the default book-wide settings. If more than one such definition of the same type is entered at the top level any definitions in the later expressions have precedence.
- A direct scheme expression, such as #(set-default-paper-size "a7" 'landscape) or #(ly:set-option 'point-and-click #f).
- A \header block. This sets the global header block. This is the block containing the definitions for book-wide settings, like composer, title, etc.
- A \score block. This score will be collected with other toplevel scores, and combined as a single \book. This behavior can be changed by setting the variable toplevel-score-handler at toplevel. The default handler is defined in the init file '../scm/lily.scm'.
- A \book block logically combines multiple movements (i.e., multiple \score blocks) in one document. If there are a number of \scores, one output file will be created for each \book block, in which all corresponding movements are concatenated. The only reason to explicitly specify \book blocks in a .ly file is if you wish to create multiple output files from a single input file. One exception is within lilypond-book documents, where you explicitly have to add a \book block if you want more than a single \score or \markup in the same example. This behavior can be changed by setting the variable toplevel-book-handler at toplevel. The default handler is defined in the init file '../scm/lily.scm'.
- A \bookpart block. A book may be divided into several parts, using \bookpart blocks, in order to ease the page breaking, or to use different \paper settings in different parts.
- A compound music expression, such as

  ```
  { c'4 d' e'2 }
  ```

  This will add the piece in a \score and format it in a single book together with all other toplevel \scores and music expressions. In other words, a file containing only the above music expression will be translated into

  ```
  \book {
    \score {
      \new Staff {
        \new Voice {
          { c'4 d' e'2 }
        }
      }
    }
  \layout { }
  \header { }
  }
  ```

  This behavior can be changed by setting the variable toplevel-music-handler at toplevel. The default handler is defined in the init file '../scm/lily.scm'.

- A markup text, a verse for example

  ```
  \markup {
     2.  The first line verse two.
  }
  ```

  Markup texts are rendered above, between or below the scores or music expressions, wherever they appear.

- A variable, such as

  ```
  foo = { c4 d e d }
  ```

  This can be used later on in the file by entering `\foo`. The name of a variable should have alphabetic characters only; no numbers, underscores or dashes.

The following example shows three things that may be entered at toplevel

```
\layout {
  % Don't justify the output
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

At any point in a file, any of the following lexical instructions can be entered:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- A single-line comment, introduced by a leading `%` sign.
- A multi-line comment delimited by `%{ .. %}`.

Whitespace between items in the input stream is generally ignored, and may be freely omitted or extended to enhance readability. However, whitespace should always be used in the following circumstances to avoid errors:

- Around every opening and closing curly bracket.
- After every command or variable, i.e. every item that begins with a `\` sign.
- After every item that is to be interpreted as a Scheme expression, i.e. every item that begins with a `#` sign.
- To separate all elements of a Scheme expression.
- In `lyricmode` to separate all the terms in both `\override` and `\set` commands. In particular, spaces must be used around the dot and the equals sign in commands like `\override Score . LyricText #'font-size = #5` and before and after the entire command.

Voir aussi

Learning Manual: Section “How LilyPond input files work” dans *Manuel d'initiation*.

3.2 Titles and headers

Almost all printed music includes a title and the composer's name; some pieces include a lot more information.

3.2.1 Creating titles

Titles are created for each `\score` block, as well as for the full input file (or `\book` block) and book parts (created by `\bookpart` blocks).

The contents of the titles are taken from the `\header` blocks. The header block for a book supports the following

`dedication`
: The dedicatee of the music, centered at the top of the first page.

`title` The title of the music, centered just below the dedication.

`subtitle` Subtitle, centered below the title.

`subsubtitle`
: Subsubtitle, centered below the subtitle.

`poet` Name of the poet, flush-left below the subsubtitle.

`instrument`
: Name of the instrument, centered below the subsubtitle. Also centered at the top of pages (other than the first page).

`composer` Name of the composer, flush-right below the subsubtitle.

`meter` Meter string, flush-left below the poet.

`arranger` Name of the arranger, flush-right below the composer.

`piece` Name of the piece, flush-left below the meter.

`opus` Name of the opus, flush-right below the arranger.

`breakbefore`
: This forces the title to start on a new page (set to ##t or ##f).

`copyright`
: Copyright notice, centered at the bottom of the first page. To insert the copyright symbol, see Section 3.3.3 [Text encoding], page 376.

`tagline` Centered at the bottom of the last page.

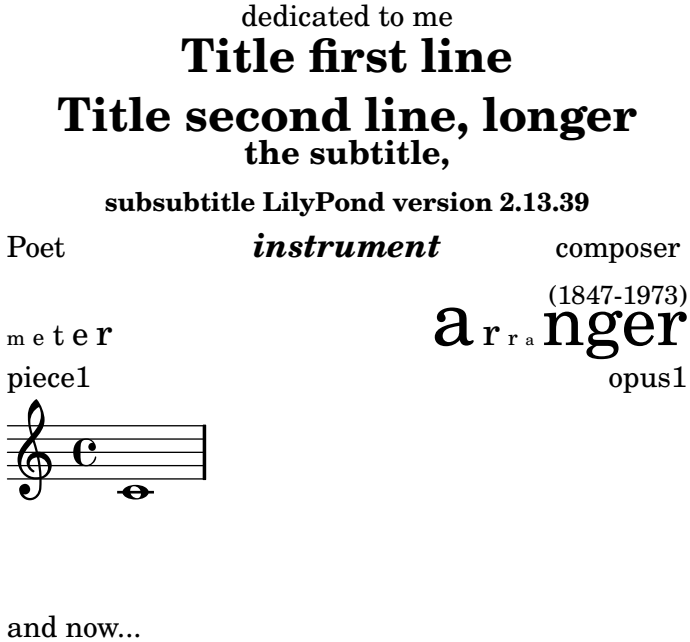
Here is a demonstration of the fields available. Note that you may use any ⟨undefined⟩ [Formatting text], page ⟨undefined⟩, commands in the header.

```
\paper {
  line-width = 9.0\cm
  paper-height = 10.0\cm
}

\book {
  \header {
    dedication = "dedicated to me"
    title = \markup \center-column { "Title first line" "Title second line,
longer" }
    subtitle = "the subtitle,"
    subsubtitle = #(string-append "subsubtitle LilyPond version "
(lilypond-version))
    poet = "Poet"
    composer =  \markup \center-column { "composer" \small "(1847-1973)" }
    texttranslator = "Text Translator"
```
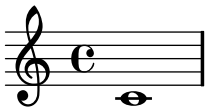
```
    meter = \markup { \teeny "m" \tiny "e" \normalsize "t" \large "e" \huge
"r" }
    arranger = \markup { \fontsize #8.5 "a" \fontsize #2.5 "r" \fontsize
#-2.5 "r" \fontsize #-5.3 "a" \fontsize #7.5 "nger" }
    instrument = \markup \bold \italic "instrument"
    piece = "Piece"
  }

  \score {
    { c'1 }
    \header {
      piece = "piece1"
      opus = "opus1"
    }
  }
  \markup {
      and now...
  }
  \score {
    { c'1 }
    \header {
      piece = "piece2"
      opus = "opus2"
    }
  }
}
```



2 ***instrument***

piece2 opus2



Music engraving by LilyPond 2.13.39—www.lilypond.org

As demonstrated before, you can use multiple `\header` blocks. When same fields appear in different blocks, the latter is used. Here is a short example.

```
\header {
  composer = "Composer"
}
\header {
  piece = "Piece"
}
\score {
  \new Staff { c'4 }
  \header {
    piece = "New piece"  % overwrite previous one
  }
}
```

If you define the `\header` inside the `\score` block, then normally only the `piece` and `opus` headers will be printed. Note that the music expression must come before the `\header`.

```
\score {
  { c'4 }
  \header {
    title = "title"  % not printed
    piece = "piece"
    opus = "opus"
  }
}
```

piece opus



You may change this behavior (and print all the headers when defining `\header` inside `\score`) by using

```
\paper{
  print-all-headers = ##t
}
```

The default footer is empty, except for the first page, where the `copyright` field from `\header` is inserted, and the last page, where `tagline` from `\header` is added. The default tagline is « Music engraving by LilyPond (*version*) ».[1]

Headers may be completely removed by setting them to false.

```
\header {
  tagline = ##f
  composer = ##f
}
```

3.2.2 Custom titles

A more advanced option is to change the definitions of the following variables in the `\paper` block. The init file '`../ly/titling-init.ly`' lists the default layout.

`bookTitleMarkup`
: This is the title added at the top of the entire output document. Typically, it has the composer and the title of the piece

`scoreTitleMarkup`
: This is the title put over a `\score` block. Typically, it has the name of the movement (`piece` field).

`oddHeaderMarkup`
: This is the page header for odd-numbered pages.

`evenHeaderMarkup`
: This is the page header for even-numbered pages. If unspecified, the odd header is used instead.

  By default, headers are defined such that the page number is on the outside edge, and the instrument is centered.

`oddFooterMarkup`
: This is the page footer for odd-numbered pages.

`evenFooterMarkup`
: This is the page footer for even-numbered pages. If unspecified, the odd header is used instead.

  By default, the footer has the copyright notice on the first, and the tagline on the last page.

The following definition will put the title flush left, and the composer flush right on a single line.

```
\paper {
  bookTitleMarkup = \markup {
   \fill-line {
     \fromproperty #'header:title
     \fromproperty #'header:composer
   }
  }
}
```

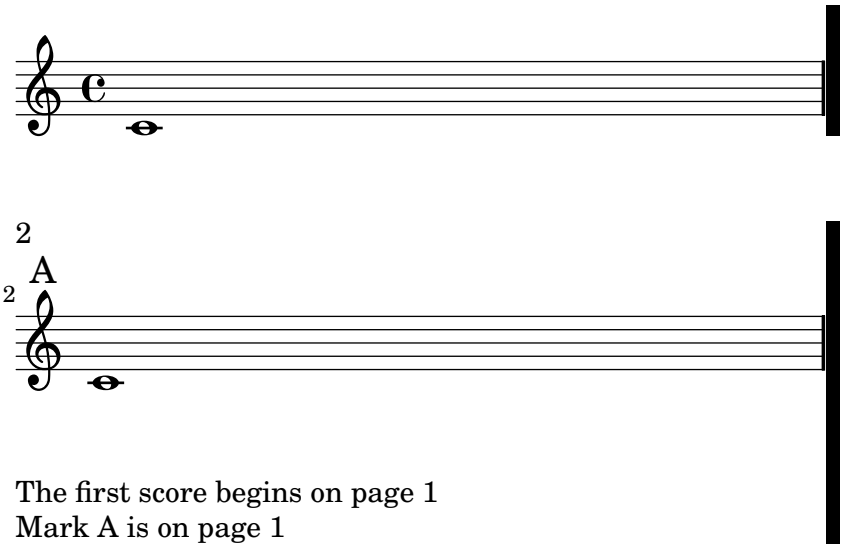
[1] Nicely printed parts are good PR for us, so please leave the tagline if you can.

3.2.3 Reference to page numbers

A particular place of a score can be marked using the `\label` command, either at top-level or inside music. This label can then be referred to in a markup, to get the number of the page where the marked point is placed, using the `\page-ref` markup command.

```
\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    {
      c'1
      \pageBreak \mark A \label #'markA
      c'1
    }
  }

  \markup { The first score begins on page \page-ref #'firstScore "0" "?" }
  \markup { Mark A is on page \page-ref #'markA "0" "?" }
}
```



The `\page-ref` markup command takes three arguments:

1. the label, a scheme symbol, eg. `#'firstScore`;
2. a markup that will be used as a gauge to estimate the dimensions of the markup;
3. a markup that will be used in place of the page number if the label is not known;

The reason why a gauge is needed is that, at the time markups are interpreted, the page breaking has not yet occurred, so the page numbers are not yet known. To work around this issue, the actual markup interpretation is delayed to a later time; however, the dimensions of the markup have to be known before, so a gauge is used to decide these dimensions. If the book has between 10 and 99 pages, it may be `"00"`, ie. a two digit number.

Commandes prédéfinies

`\label`, `\page-ref`.

3.2.4 Table of contents

A table of contents is included using the `\markuplines \table-of-contents` command. The elements which should appear in the table of contents are entered with the `\tocItem` command, which may be used either at top-level, or inside a music expression.

```
\markuplines \table-of-contents
\pageBreak

\tocItem \markup "First score"
\score {
  {
    c'4  % ...
    \tocItem \markup "Some particular point in the first score"
    d'4  % ...
  }
}

\tocItem \markup "Second score"
\score {
  {
    e'4 % ...
  }
}
```

The markups which are used to format the table of contents are defined in the `\paper` block. The default ones are `tocTitleMarkup`, for formatting the title of the table, and `tocItemMarkup`, for formatting the toc elements, composed of the element title and page number. These variables may be changed by the user:

```
\paper {
  %% Translate the toc title into French:
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
  %% use larger font size
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Note how the toc element text and page number are referred to in the `tocItemMarkup` definition.

New commands and markups may also be defined to build more elaborated table of contents:

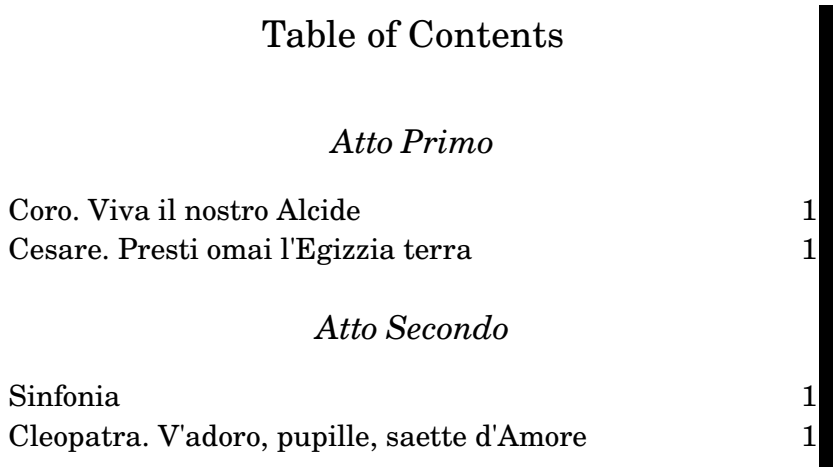
- first, define a new markup variable in the `\paper` block
- then, define a music function which aims at adding a toc element using this markup paper variable.

In the following example, a new style is defined for entering act names in the table of contents of an opera:

```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
```

```
    \hspace #1
  }
}

tocAct =
#(define-music-function (parser location text) (markup?)
   (add-toc-item! 'tocActMarkup text))
```



Voir aussi

Init files: '`../ly/toc-init.ly`'.

Commandes prédéfinies

`\table-of-contents`, `\tocItem`.

3.3 Working with input files

3.3.1 Including LilyPond files

A large project may be split up into separate files. To refer to another file, use

```
\include "otherfile.ly"
```

The line `\include "otherfile.ly"` is equivalent to pasting the contents of '`otherfile.ly`' into the current file at the place where the `\include` appears. For example, in a large project you might write separate files for each instrument part and create a « full score » file which brings together the individual instrument files. Normally the included file will define a number of variables which then become available for use in the full score file. Tagged sections can be marked in included files to assist in making them usable in different places in a score, see Section 3.3.2 [Different editions from one source], page 372.

Files in the current working directory may be referenced by specifying just the file name after the `\include` command. Files in other locations may be included by giving either a full path reference or a relative path reference (but use the UNIX forward slash, /, rather than the DOS/Windows back slash, \, as the directory separator.) For example, if '`stuff.ly`' is located one directory higher than the current working directory, use

```
\include "../stuff.ly"
```

or if the included orchestral parts files are all located in a subdirectory called '`parts`' within the current directory, use

```
\include "parts/VI.ly"
\include "parts/VII.ly"
... etc
```

Files which are to be included can also contain `\include` statements of their own. By default, these second-level `\include` statements are not interpreted until they have been brought into the main file, so the file names they specify must all be relative to the directory containing the main file, not the directory containing the included file. However, this behavior can be changed by passing the option `-drelative-includes` option at the command line (or by adding `#(ly:set-option 'relative-includes #t)` at the top of the main input file). With `relative-includes` set, the path for each `\include` command will be taken relative to the file containing that command. This behavior is recommended and it will become the default behavior in a future version of lilypond.

Files can also be included from a directory in a search path specified as an option when invoking LilyPond from the command line. The included files are then specified using just their file name. For example, to compile '`main.ly`' which includes files located in a subdirectory called '`parts`' by this method, cd to the directory containing '`main.ly`' and enter

```
lilypond --include=parts main.ly
```

and in main.ly write

```
\include "VI.ly"
\include "VII.ly"
... etc
```

Files which are to be included in many scores may be placed in the LilyPond directory '`../ly`'. (The location of this directory is installation-dependent - see Section "Other sources of information" dans *Manuel d'initiation*). These files can then be included simply by naming them on an `\include` statement. This is how the language-dependent files like '`english.ly`' are included.

LilyPond includes a number of files by default when you start the program. These includes are not apparent to the user, but the files may be identified by running `lilypond --verbose` from the command line. This will display a list of paths and files that LilyPond uses, along with much other information. Alternatively, the more important of these files are discussed in Section "Other sources of information" dans *Manuel d'initiation*. These files may be edited, but changes to them will be lost on installing a new version of LilyPond.

Some simple examples of using `\include` are shown in Section "Scores and parts" dans *Manuel d'initiation*.

Voir aussi

Learning Manual: Section "Other sources of information" dans *Manuel d'initiation*, Section "Scores and parts" dans *Manuel d'initiation*.

Problèmes connus et avertissements

If an included file is given a name which is the same as one in LilyPond's installation files, LilyPond's file from the installation files takes precedence.
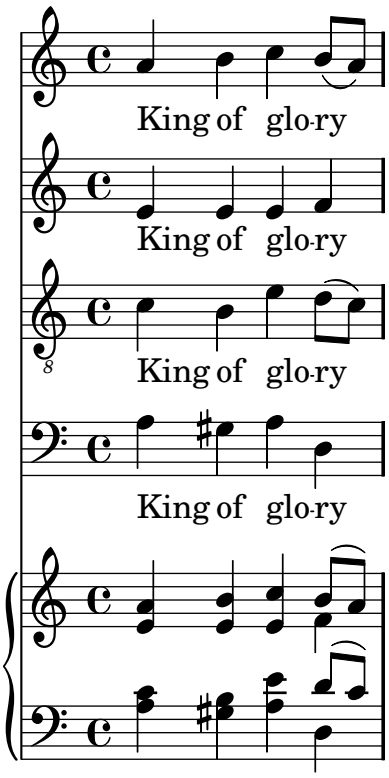
3.3.2 Different editions from one source

Several mechanisms are available to facilitate the generation of different versions of a score from the same music source. Variables are perhaps most useful for combining lengthy sections of music and/or annotation in various ways, while tags are more useful for selecting one from several alternative shorter sections of music. Whichever method is used, separating the notation from the structure of the score will make it easier to change the structure while leaving the notation untouched.

Using variables

If sections of the music are defined in variables they can be reused in different parts of the score, see Section “Organizing pieces with variables” dans *Manuel d'initiation*. For example, an *a cappella* vocal score frequently includes a piano reduction of the parts for rehearsal purposes which is identical to the vocal music, so the music need be entered only once. Music from two variables may be combined on one staff, see ⟨undefined⟩ [Automatic part combining], page ⟨undefined⟩. Here is an example:

```
sopranoMusic = \relative c'' { a4 b c b8( a) }
altoMusic = \relative g' { e4 e e f }
tenorMusic = \relative c' { c4 b e d8( c) }
bassMusic = \relative c' { 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
    }
  >>
>>
```



Separate scores showing just the vocal parts or just the piano part can be produced by changing just the structural statements, leaving the musical notation unchanged.

For lengthy scores, the variable definitions may be placed in separate files which are then included, see Section 3.3.1 [Including LilyPond files], page 371.

Using tags

The `\tag #'partA` command marks a music expression with the name *partA*. Expressions tagged in this way can be selected or filtered out by name later, using either `\keepWithTag #'name` or `\removeWithTag #'name`. The result of applying these filters to tagged music is as follows:

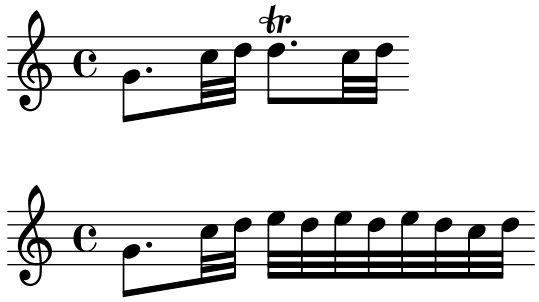
Filter	Result
Tagged music preceded by `\keepWithTag #'name`	Untagged music and music tagged with *name* is included; music tagged with any other tag name is excluded.
Tagged music preceded by `\removeWithTag #'name`	Untagged music and music tagged with any tag name other than *name* is included; music tagged with *name* is excluded.
Tagged music not preceded by either `\keepWithTag` or `\removeWithTag`	All tagged and untagged music is included.

The arguments of the `\tag`, `\keepWithTag` and `\removeWithTag` commands should be a symbol (such as `#'score` or `#'part`), followed by a music expression.

In the following example, we see two versions of a piece of music, one showing trills with the usual notation, and one with trills explicitly expanded:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }
```

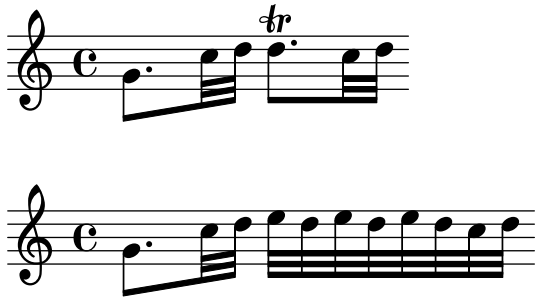
```
\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}
```



Alternatively, it is sometimes easier to exclude sections of music:

```
music = \relative g' {
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \removeWithTag #'expand
  \music
}
\score {
  \removeWithTag #'trills
  \music
}
```



Tagged filtering can be applied to articulations, texts, etc. by prepending

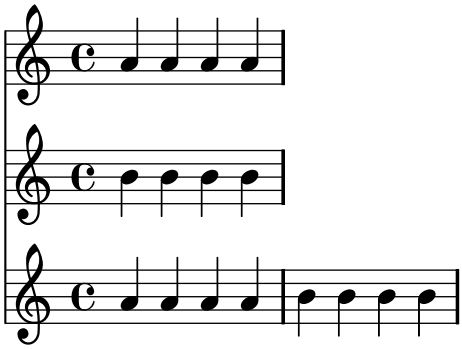
`-\tag #'`*your-tag*

to an articulation. For example, this would define a note with a conditional fingering indication and a note with a conditional annotation:

```
c1-\tag #'finger ^4
c1-\tag #'warn ^"Watch!"
```
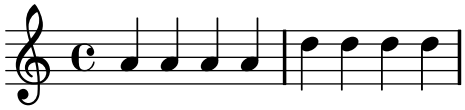
Multiple tags may be placed on expressions with multiple `\tag` entries:

```
music = \relative c'' {
  \tag #'a \tag #'both { a4 a a a }
  \tag #'b \tag #'both { b4 b b b }
}
<<
\keepWithTag #'a \music
\keepWithTag #'b \music
\keepWithTag #'both \music
>>
```



Multiple `\removeWithTag` filters may be applied to a single music expression to remove several differently named tagged sections:

```
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 }
}
{
\removeWithTag #'B
\removeWithTag #'C
\music
}
```



Two or more `\keepWithTag` filters applied to a single music expression will cause *all* tagged sections to be removed, as the first filter will remove all tagged sections except the one named, and the second filter will remove even that tagged section.

Voir aussi

Learning Manual: Section “Organizing pieces with variables” dans *Manuel d'initiation*.

Notation Reference: ⟨undefined⟩ [Automatic part combining], page ⟨undefined⟩, Section 3.3.1 [Including LilyPond files], page 371.

3.3.3 Text encoding

LilyPond uses the character repertoire defined by the Unicode consortium and ISO/IEC 10646. This defines a unique name and code point for the character sets used in virtually all modern languages and many others too. Unicode can be implemented using several different encodings.

LilyPond uses the UTF-8 encoding (UTF stands for Unicode Transformation Format) which represents all common Latin characters in one byte, and represents other characters using a variable length format of up to four bytes.

The actual appearance of the characters is determined by the glyphs defined in the particular fonts available - a font defines the mapping of a subset of the Unicode code points to glyphs. LilyPond uses the Pango library to layout and render multi-lingual texts.
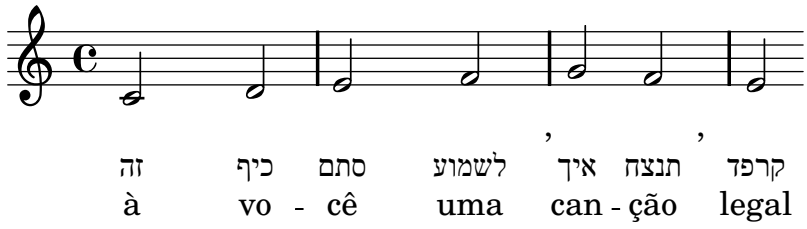
LilyPond does not perform any input-encoding conversions. This means that any text, be it title, lyric text, or musical instruction containing non-ASCII characters, must be encoded in UTF-8. The easiest way to enter such text is by using a Unicode-aware editor and saving the file with UTF-8 encoding. Most popular modern editors have UTF-8 support, for example, vim, Emacs, jEdit, and GEdit do. All MS Windows systems later than NT use Unicode as their native character encoding, so even Notepad can edit and save a file in UTF-8 format. A more functional alternative for Windows is BabelPad.

If a LilyPond input file containing a non-ASCII character is not saved in UTF-8 format the error message

```
FT_Get_Glyph_Name () error: invalid argument
```

will be generated.

Here is an example showing Cyrillic, Hebrew and Portuguese text:



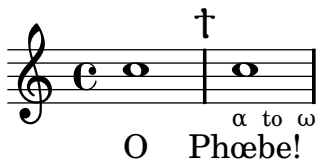
To enter a single character for which the Unicode code point is known but which is not available in the editor being used, use either `\char ##xhhhh` or `\char #dddd` within a `\markup` block, where `hhhh` is the hexadecimal code for the character required and `dddd` is the corresponding decimal value. Leading zeroes may be omitted, but it is usual to specify all four characters in the hexadecimal representation. (Note that the UTF-8 encoding of the code point should *not* be used after `\char`, as UTF-8 encodings contain extra bits indicating the number of octets.) Unicode code charts and a character name index giving the code point in hexadecimal for any character can be found on the Unicode Consortium website, http://www.unicode.org/.

For example, `\char ##x03BE` and `\char #958` would both enter the Unicode U+03BE character, which has the Unicode name « Greek Small Letter Xi ».

Any Unicode code point may be entered in this way and if all special characters are entered in this format it is not necessary to save the input file in UTF-8 format. Of course, a font containing all such encoded characters must be installed and available to LilyPond.

The following example shows Unicode hexadecimal values being entered in four places – in a rehearsal mark, as articulation text, in lyrics and as stand-alone text below the score:

```
\score {
  \relative c'' {
    c1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2010" \char ##x00A9 }
```



Copyright 2008--2010 ©

To enter the copyright sign in the copyright notice use:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

3.3.4 Displaying LilyPond notation

Displaying a music expression in LilyPond notation can be done with the music function `\displayLilyMusic` but only when using the command line. For example,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

will display

```
{ a,4 cis e fis g }
```

By default, LilyPond will print these messages to the console along with all the other LilyPond compilation messages. To split up these messages and save the results of `\display{STUFF}`, redirect the output to a file.

```
lilypond file.ly >display.txt
```

3.4 Controlling output

3.4.1 Extracting fragments of music

It is possible to quote small fragments of a large score directly from the output. This can be compared to clipping a piece of a paper score with scissors.

This is done by defining the measures that need to be cut out separately. For example, including the following definition

```
\layout {
  clip-regions
  = #(list
      (cons
       (make-rhythmic-location 5 1 2)
       (make-rhythmic-location 7 3 4)))
}
```

will extract a fragment starting halfway the fifth measure, ending in the seventh measure. The meaning of `5 1 2` is: after a 1/2 note in measure 5, and `7 3 4` after 3 quarter notes in measure 7.

More clip regions can be defined by adding more pairs of rhythmic-locations to the list.

In order to use this feature, LilyPond must be invoked with `-dclip-systems`. The clips are output as EPS files, and are converted to PDF and PNG if these formats are switched on as well.

For more information on output formats, see Section “Invoking lilypond” dans *Utilisation des programmes*.

3.4.2 Skipping corrected music

When entering or copying music, usually only the music near the end (where you are adding notes) is interesting to view and correct. To speed up this correction process, it is possible to skip typesetting of all but the last few measures. This is achieved by putting

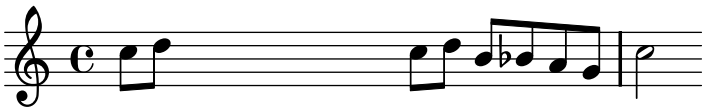
```
showLastLength = R1*5
\score { ... }
```

in your source file. This will render only the last 5 measures (assuming 4/4 time signature) of every `\score` in the input file. For longer pieces, rendering only a small part is often an order of magnitude quicker than rendering it completely. When working on the beginning of a score you have already typeset (e.g. to add a new part), the `showFirstLength` property may be useful as well.

Skipping parts of a score can be controlled in a more fine-grained fashion with the property `Score.skipTypesetting`. When it is set, no typesetting is performed at all.

This property is also used to control output to the MIDI file. Note that it skips all events, including tempo and instrument changes. You have been warned.

```
c8 d
\set Score.skipTypesetting = ##t
e8 e e e e e e e
\set Score.skipTypesetting = ##f
c8 d b bes a g c2
```



In polyphonic music, `Score.skipTypesetting` will affect all voices and staves, saving even more time.

3.4.3 Alternative output formats
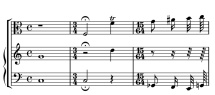
The default output formats for the printed score are Portable Document Format (PDF) and PostScript (PS). Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) and Portable Network Graphics (PNG) output formats are also available through command line options, see Section “Command line options for lilypond” dans *Utilisation des programmes*.

3.4.4 Replacing the notation font
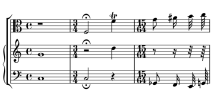
Gonville is an alternative to the Feta font used in LilyPond and can be downloaded from:

http://www.chiark.greenend.org.uk/~sgtatham/gonville/

Here are a few sample bars of music set in Gonville:



Here are a few sample bars of music set in LilyPond’s Feta font:



Installation Instructions for MacOS

Download and extract the zip file. Copy the `lilyfonts` directory to '*SHARE_DIR*`/lilypond/current`'; for more information, see Section "Other sources of information" dans *Manuel d'initiation*. Move the existing `fonts` directory to `fonts_orig` and move the `lilyfonts` directory to `fonts`. Simply move `fonts_orig` back to `fonts` to revert back to Feta.

Voir aussi

Learning Manual: Section "Other sources of information" dans *Manuel d'initiation*.

Problèmes connus et avertissements

Gonville cannot be used to typeset 'Ancient Music' notation. Please refer to the author's website for more information on this and other specifics including licensing of Gonville.

3.5 MIDI output

MIDI (Musical Instrument Digital Interface) is a standard for connecting and controlling digital instruments. A MIDI file is a series of notes in a number of tracks. It is not an actual sound file; you need special software to translate between the series of notes and actual sounds.

Pieces of music can be converted to MIDI files, so you can listen to what was entered. This is convenient for checking the music; octaves that are off or accidentals that were mistyped stand out very much when listening to the MIDI output.

The midi output allocates a channel for each staff, and one for global settings. Therefore the midi file should not have more than 15 staves (or 14 if you do not use drums). Other staves will remain silent.

3.5.1 Creating MIDI files

To create a MIDI output file from a LilyPond input file, add a `\midi` block to a score, for example,

```
\score {
  ...music...
  \midi { }
}
```

If there is a `\midi` block in a `\score` with no `\layout` block, only MIDI output will be produced. When notation is needed too, a `\layout` block must also be present.

```
\score {
  ...music...
  \midi { }
  \layout { }
}
```

Pitches, rhythms, ties, dynamics, and tempo changes are interpreted and translated correctly to the MIDI output. Dynamic marks, crescendi and decrescendi translate into MIDI volume levels. Dynamic marks translate to a fixed fraction of the available MIDI volume range. Crescendi and decrescendi make the volume vary linearly between their two extremes. The effect of dynamic markings on the MIDI output can be removed completely, see Section 3.5.2 [MIDI block], page 382.

The initial tempo and later tempo changes can be specified with the `\tempo` command within the music notation. These are reflected in tempo changes in the MIDI output. This command will normally result in the metronome mark being printed, but this can be suppressed, see ⟨undefined⟩ [Metronome marks], page ⟨undefined⟩. An alternative way of specifying the initial or overall MIDI tempo is described below, see Section 3.5.2 [MIDI block], page 382.

Due to some limitations on Windows, the default extension for MIDI files on Windows is `.mid`. Other operating systems still use the extension `.midi`. If a different extension is preferred, insert the following line at the top-level of the input file, before the start of any `\book`, `\bookpart` or `\score` blocks:

```
#(ly:set-option 'midi-extension "midi")
```

The line above will set the default extension for MIDI files to `.midi`.

Alternatively, this option can also be supplied on the command line:

```
lilypond ... -dmidi-extension=midi lilyFile.ly
```

Instrument names

The MIDI instrument to be used is specified by setting the `Staff.midiInstrument` property to the instrument name. The name should be chosen from the list in Section A.4 [MIDI instruments], page 494.

```
\new Staff {
  \set Staff.midiInstrument = #"glockenspiel"
  ...notes...
}
\new Staff \with {midiInstrument = #"cello"} {
  ...notes...
}
```

If the selected instrument does not exactly match an instrument from the list of MIDI instruments, the Grand Piano (`"acoustic grand"`) instrument is used.

Morceaux choisis
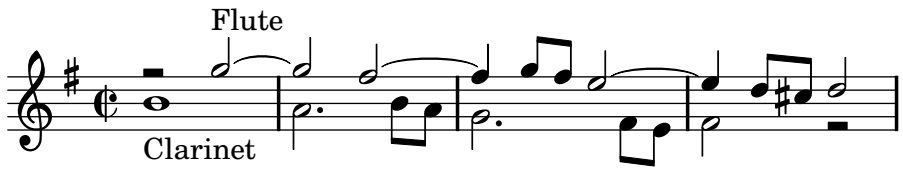
Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `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"
    }
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



Problèmes connus et avertissements

Changes in the MIDI volume take place only on starting a note, so crescendi and decrescendi cannot affect the volume of a single note.

Not all midi players correctly handle tempo changes in the midi output. Players that are known to work include MS Windows Media Player and timidity.

3.5.2 MIDI block

A `\midi` block must appear within a score block if MIDI output is required. It is analogous to the layout block, but somewhat simpler. Often, the `\midi` block is left empty, but it can contain context rearrangements, new context definitions or code to set the values of properties. For example, the following will set the initial tempo exported to a MIDI file without causing a tempo indication to be printed:

```
\score {
  ...music...
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 4)
    }
  }
}
```

In this example the tempo is set to 72 quarter note beats per minute. This kind of tempo specification cannot take a dotted note length as an argument. If one is required, break the

dotted note into smaller units. For example, a tempo of 90 dotted quarter notes per minute can be specified as 270 eighth notes per minute:

```
tempoWholesPerMinute = #(ly:make-moment 270 8)
```

Context definitions follow precisely the same syntax as those within a `\layout` block. Translation modules for sound are called performers. The contexts for MIDI output are defined in '`../ly/performer-init.ly`', see Section "Other sources of information" dans *Manuel d'initiation*. For example, to remove the effect of dynamics from the MIDI output, insert the following lines in the `\midi{ }` block.

```
\midi {
  ...
  \context {
    \Voice
    \remove "Dynamic_performer"
  }
}
```

MIDI output is created only when a `\midi` block is included within a score block defined with a `\score` command.

```
\score {
  { ...notes... }
  \midi { }
}
```

3.5.3 What goes into the MIDI output?

Supported in MIDI

The following items of notation are reflected in the MIDI output:

- Pitches
- Microtones (See ⟨undefined⟩ [Accidentals], page ⟨undefined⟩. Rendering needs a player that supports pitch bend.)
- Chords entered as chord names
- Rhythms entered as note durations, including tuplets
- Tremolos entered without '`:`[*number*]'
- Ties
- Dynamic marks
- Crescendi, decrescendi over multiple notes
- Tempo changes entered with a tempo marking
- Lyrics

Unsupported in MIDI

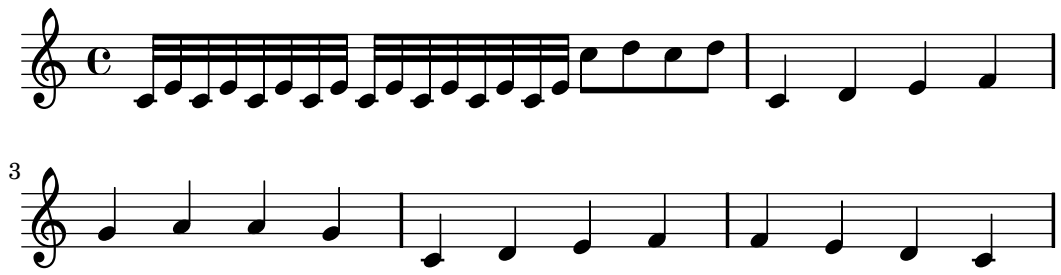
The following items of notation have no effect on the MIDI output:

- Rhythms entered as annotations, e.g. swing
- Tempo changes entered as annotations with no tempo marking
- Staccato and other articulations and ornamentations
- Slurs and Phrasing slurs
- Crescendi, decrescendi over a single note
- Tremolos entered with '`:`[*number*]'
- Figured bass
- Microtonal chords

3.5.4 Repeats in MIDI

With a few minor additions, all types of repeats can be represented in the MIDI output. This is achieved by applying the `\unfoldRepeats` music function. This function changes all repeats to unfold repeats.

```
\unfoldRepeats {
  \repeat tremolo 8 { c'32 e' }
  \repeat percent 2 { c''8 d'' }
  \repeat volta 2 { c'4 d' e' f' }
  \alternative {
    { g' a' a' g' }
    { f' e' d' c' }
  }
}
\bar "|."
```



When creating a score file using `\unfoldRepeats` for MIDI, it is necessary to make two `\score` blocks: one for MIDI (with unfolded repeats) and one for notation (with volta, tremolo, and percent repeats). For example,

```
\score {
  ..music..
  \layout { .. }
}
\score {
  \unfoldRepeats ..music..
  \midi { .. }
}
```

3.5.5 Controlling MIDI dynamics

MIDI dynamics are implemented by the Dynamic_performer which lives by default in the Voice context. It is possible to control the overall MIDI volume, the relative volume of dynamic markings and the relative volume of different instruments.

Dynamic marks

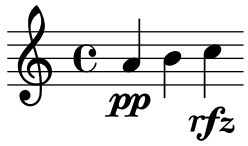
Dynamic marks are translated to a fixed fraction of the available MIDI volume range. The default fractions range from 0.25 for *ppppp* to 0.95 for *fffff*. The set of dynamic marks and the associated fractions can be seen in '`../scm/midi.scm`', see Section "Other sources of information" dans *Manuel d'initiation*. This set of fractions may be changed or extended by providing a function which takes a dynamic mark as its argument and returns the required fraction, and setting `Score.dynamicAbsoluteVolumeFunction` to this function.

For example, if a *rinforzando* dynamic marking, `\rfz`, is required, this will not by default have any effect on the MIDI volume, as this dynamic marking is not included in the default set. Similarly, if a new dynamic marking has been defined with `make-dynamic-script` that too will

not be included in the default set. The following example shows how the MIDI volume for such dynamic markings might be added. The Scheme function sets the fraction to 0.9 if a dynamic mark of rfz is found, or calls the default function otherwise.

```
#(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 c'' {
        a4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Alternatively, if the whole table of fractions needs to be redefined, it would be better to use the *default-dynamic-absolute-volume* procedure in '`../scm/midi.scm`' and the associated table as a model. The final example in this section shows how this might be done.

Overall MIDI volume
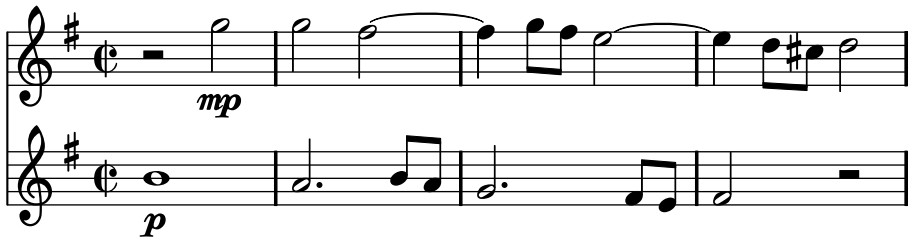
The minimum and maximum overall volume of MIDI dynamic markings is controlled by setting the properties `midiMinimumVolume` and `midiMaximumVolume` at the `Score` level. These properties have an effect only on dynamic marks, so if they are to apply from the start of the score a dynamic mark must be placed there. The fraction corresponding to each dynamic mark is modified with this formula

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * fraction
```

In the following example the dynamic range of the overall MIDI volume is limited to the range 0.2 - 0.5.

```
\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
```

```
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```
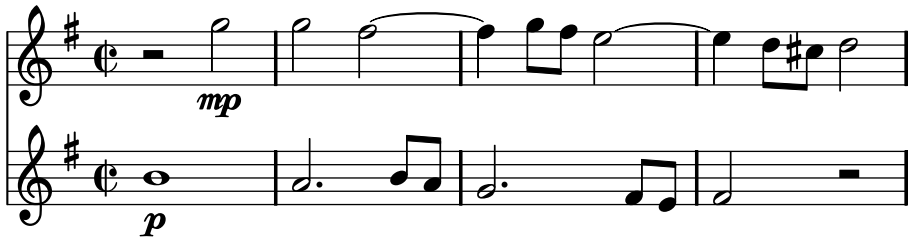


Equalizing different instruments (i)

If the minimum and maximum MIDI volume properties are set in the `Staff` context the relative volumes of the MIDI instruments can be controlled. This gives a basic instrument equalizer, which can enhance the quality of the MIDI output remarkably.

In this example the volume of the clarinet is reduced relative to the volume of the flute. There must be a dynamic mark on the first note of each instrument for this to work correctly.

```
\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Staff.midiInstrument = #"flute"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      \new Voice \relative c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
```

```
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout {}
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



Equalizing different instruments (ii)

If the MIDI minimum and maximum volume properties are not set LilyPond will, by default, apply a small degree of equalization to a few instruments. The instruments and the equalization applied are shown in the table *instrument-equalizer-alist* in '`../scm/midi.scm`'.

This basic default equalizer can be replaced by setting `instrumentEqualizer` in the `Score` context to a new Scheme procedure which accepts a MIDI instrument name as its only argument and returns a pair of fractions giving the minimum and maximum volumes to be applied to that instrument. This replacement is done in the same way as shown for resetting the `dynamicAbsoluteVolumeFunction` at the start of this section. The default equalizer, *default-instrument-equalizer*, in '`../scm/midi.scm`' shows how such a procedure might be written.

The following example sets the relative flute and clarinet volumes to the same values as the previous example.
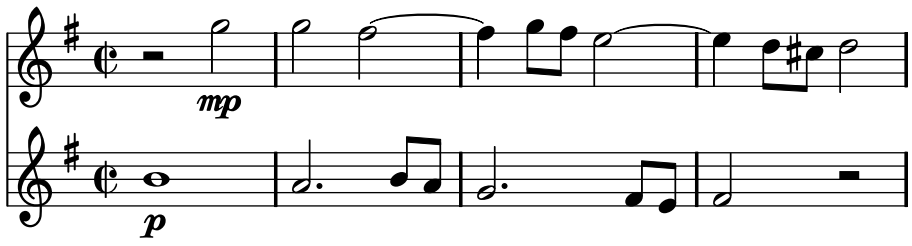
```
#(define my-instrument-equalizer-alist '())

#(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 c''' {
        r2 g\mp g fis~
        fis4 g8 fis e2~
        e4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = #"clarinet"
      \new Voice \relative c'' {
        b1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {
    \context {
      \Score
      tempoWholesPerMinute = #(ly:make-moment 72 2)
    }
  }
}
```



3.5.6 Percussion in MIDI

Percussion instruments are generally notated in a `DrumStaff` context and when notated in this way they are outputted correctly to MIDI channel 10, but some pitched percussion instruments, like the xylophone, marimba, vibraphone, timpani, etc., are treated like « normal » instruments and music for these instruments should be entered in a normal `Staff` context, not a `DrumStaff` context, to obtain the correct MIDI output.

Some non-pitched percussion sounds included in the general MIDI standard, like melodic tom, taiko drum, synth drum, etc., cannot be reached via MIDI channel 10, so the notation

for such instruments should also be entered in a normal `Staff` context, using suitable normal pitches.

Many percussion instruments are not included in the general MIDI standard, e.g. castanets. The easiest, although unsatisfactory, method of producing some MIDI output when writing for such instruments is to substitute the nearest sound from the standard set.

Problèmes connus et avertissements

Because the general MIDI standard does not contain rim shots, the sidestick is used for this purpose instead.

4 Spacing issues

The global paper layout is determined by three factors: the page layout, the line breaks, and the spacing. These all influence each other. The choice of spacing determines how densely each system of music is set. This influences where line breaks are chosen, and thus ultimately, how many pages a piece of music takes.

Globally speaking, this procedure happens in four steps: first, flexible distances ('springs') are chosen, based on durations. All possible line breaking combinations are tried, and a 'badness' score is calculated for each. Then the height of each possible system is estimated. Finally, a page breaking and line breaking combination is chosen so that neither the horizontal nor the vertical spacing is too cramped or stretched.

Settings which influence layout may be placed in two blocks. The `\paper {...}` block is placed outside any `\score {...}` blocks and contains settings that relate to the entire document. The `\layout {...}` block is placed within a `\score {...}` block and contains settings for that particular score. If you have only one `\score {...}` block the two have the same effect. In general the commands shown in this chapter can be placed in either.

4.1 Paper and pages

This section deals with the boundaries that define the area within which music can be printed.

4.1.1 Paper size

Two functions are available for changing the paper size: `set-default-paper-size` and `set-paper-size`. `set-default-paper-size` must be placed in the toplevel scope, and `set-paper-size` must be placed in a `\paper` block:

```
#(set-default-paper-size "a4")

\paper {
  #(set-paper-size "a4")
}
```

`set-default-paper-size` sets the size of all pages, whereas `set-paper-size` only sets the size of the pages that the `\paper` block applies to. For example, if the `\paper` block is at the top of the file, then it will apply the paper size to all pages. If the `\paper` block is inside a `\book`, then the paper size will only apply to that book.

Common paper sizes are available, including `a4`, `letter`, `legal`, and `11x17` (also known as tabloid). Many more paper sizes are supported by default. For details, see '`scm/paper.scm`', and search for the definition of `paper-alist`.

Note : The default paper size is `a4`.

Extra sizes may be added by editing the definition of `paper-alist` in the initialization file '`scm/paper.scm`', however they will be overridden on a subsequent install.

If the symbol `'landscape` is supplied as an argument to `set-default-paper-size`, pages will be rotated by 90 degrees, and wider line widths will be set accordingly.

```
#(set-default-paper-size "a6" 'landscape)
```

Setting the paper size will adjust a number of `\paper` variables, such as margins. To use a particular paper size with altered `\paper` variables, set the paper size before setting the variables.

Voir aussi

Installed Files: '`scm/paper.scm`'.

Snippets: Section "Spacing" dans *Morceaux choisis*.

4.1.2 Page formatting

Margins, headers, and footers and other layout variables are automatically set according to the paper size.

Default margin values are accessible in '`ly/paper-defaults-init.ly`'. They apply to the default paper size (a4, unless specified differently) and are scaled accordingly for other paper sizes.

This section lists and describes a number of paper variables that may be altered.

Fixed vertical dimensions

`paper-height`
: The height of the page. Default: the height of the current paper size. For details, see Section 4.1.1 [Paper size], page 390.

`top-margin`
: The margin between the top of the page and the top of the printable area. Default: `5\mm`.

`bottom-margin`
: The margin between the bottom of the printable area and the bottom of the page. Default: `6\mm`.

Flexible vertical dimensions

In most cases, it is preferable for the vertical distances between certain items (such as margins, titles, systems, and separate scores) to be flexible, so that they stretch and compress nicely according to each situation. A number of `\paper` variables (listed below) are available to fine-tune the stretching behavior of these dimensions.

Note that the `\paper` variables discussed in this section do not control the spacing of staves within individual systems. Within-system spacing is controlled by grob properties, with settings typically entered inside a `\score` or `layout` block, and not inside a `\paper` block. See Section 4.4.1 [Flexible vertical spacing within systems], page 407.

Structure of spacing alists for `\paper` variables

Each of these variables is an alist (association list) containing four *keys*:

- `padding` – the minimum required amount of unobstructed vertical whitespace between two items, measured in staff-spaces. This can be thought of as the minimum height of an unobstructed (invisible) rectangle that extends from the leftmost to the rightmost point of the combined items.
- `space` – the default 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. The reference point of a (title or top-level) markup is its highest point, and the reference point of a system is the vertical center of the nearest `StaffSymbol` – even if a non-staff line (such as a `Lyrics` context) is in the way. Values for `space` that are less than either `padding` or `minimum-distance` are not meaningful, since the resulting distance will never be less than either `padding` or `minimum-distance`.
- `minimum-distance` – the minimum required vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect. Values for `minimum-distance` that are less than `padding` are not meaningful, since the resulting distance will never be less than `padding`.
- `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result). When positive, the

significance of a particular dimension's `stretchability` value lies only in its relation to the `stretchability` values of the other dimensions. For example, if one dimension has twice the `stretchability` of another, it will stretch twice as easily. Values should be non-negative and finite. The value `+inf.0` triggers a `programming_error` and is ignored, but `1.0e7` can be used for an almost infinitely stretchable spring. If unset, the default value is set to `space`. Note that the dimension's propensity to *compress* cannot be directly set by the user and is equal to (`space` − `minimum-distance`).

If a page has a ragged bottom, the resulting distance is the largest of:

- `space`,
- `minimum-distance`, and
- `padding` plus the smallest distance necessary to eliminate collisions.

Specific methods for modifying alists are discussed in Section 5.3.6 [Modifying alists], page 455. The flexible vertical `\paper` dimensions variables can only be set within a `\paper` block.

The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second complete re-defines the variable:

```
\paper {
  system-system-spacing #'space = #8

  score-system-spacing =
    #'((padding . 1)
       (space . 12)
       (minimum-distance . 6)
       (stretchability . 12))
}
```

The initialized default settings for these variables are defined in '`ly/paper-defaults-init.ly`'.

Flexible vertical dimension \paper variables

The names of these variables follow the format *`upper-lower-`*`spacing`, where *`upper`* and *`lower`* are the items to be spaced. Each distance is measured between the reference points of the two items (see the description of the alist structure above). Note that in these variable names, the term '`markup`' refers to both *title markups* (`bookTitleMarkup` or `scoreTitleMarkup`) and *top-level markups* (see Section 3.1.5 [File structure], page 363).

`markup-system-spacing`
: the distance between a (title or top-level) markup and the system that follows it.

`score-markup-spacing`
: the distance between the last system of a score and the (title or top-level) markup that follows it.

`score-system-spacing`
: the distance between the last system of a score and the first system of the score that follows it, when no (title or top-level) markup exists between them.

`system-system-spacing`
: the distance between two systems in the same score.

`markup-markup-spacing`
: the distance between two (title or top-level) markups.

`last-bottom-spacing`
: the distance from the last system or top-level markup on a page to the bottom of the printable area (i.e. the top of the bottom margin).

`top-system-spacing`
: the distance from the top of the printable area (i.e. the bottom of the top margin) to the first system on a page, when there is no (title or top-level) markup between the two.

`top-markup-spacing`
: the distance from the top of the printable area (i.e. the bottom of the top margin) to the first (title or top-level) markup on a page, when there is no system between the two.

Morceaux choisis

The header and footer are created by the functions `make-footer` and `make-header`, defined in `\paper`. The default implementations are in '`ly/paper-defaults.ly`' and '`ly/titling-init.ly`'.

The page layout itself is done by two functions in the `\paper` block, `page-music-height` and `page-make-stencil`. The former tells the line-breaking algorithm how much space can be spent on a page, the latter creates the actual page given the system to put on it.

You can define `\paper` block values in Scheme. In that case `mm`, `in`, `pt`, and `cm` are variables defined in '`paper-defaults.ly`' with values in millimeters. That is why the value `2 cm` must be multiplied in the example:

```
\paper {
 #(define bottom-margin (* 2 cm))
}
```

Example:

```
\paper {
  paper-width = 2\cm
  top-margin = 3\cm
  bottom-margin = 3\cm
  ragged-last-bottom = ##t
}
```

This second example centers page numbers at the bottom of every page.

```
\paper {
  print-page-number = ##t
  print-first-page-number = ##t
  oddHeaderMarkup = \markup \fill-line { " " }
  evenHeaderMarkup = \markup \fill-line { " " }
  oddFooterMarkup = \markup { \fill-line {
     \bold \fontsize #3 \on-the-fly #print-page-number-check-first
     \fromproperty #'page:page-number-string } }
  evenFooterMarkup = \markup { \fill-line {
     \bold \fontsize #3 \on-the-fly #print-page-number-check-first
     \fromproperty #'page:page-number-string } }
}
```

Voir aussi

Notation Reference: Section 4.4.1 [Flexible vertical spacing within systems], page 407.

Snippets: Section "Spacing" dans *Morceaux choisis*.

Horizontal dimensions

There are a few variables that determine the horizontal dimensions on a page:

`binding-offset`
: The amount `inner-margin` is increased to make sure nothing will be hidden by the binding. Works only with `two-sided` set to true. Default: `0`.

`horizontal-shift`
: The amount that all systems (including titles and system separators) are shifted to the right. Default: `0.0`.

`indent`
: The level of indentation for the first system in a score. Default: `15\mm`.

`inner-margin`
: The margin all pages have at the inner side if they are part of a book. Works only with `two-sided` set to true. Default: `10\mm`.

`left-margin`
: The margin between the left edge of the page and the beginning of each system. Default: `10\mm`.

`line-width`
: The width of music systems. Default: `paper-width` minus `left-margin` and `right-margin`.

`outer-margin`
: The margin all pages have at the outer side if they are part of a book. Works only with `two-sided` set to true. Default: `20\mm`.

`paper-width`
: The width of the page. Default: the width of the current paper size. For details, see Section 4.1.1 [Paper size], page 390.

`right-margin`
: The margin between the right edge of the page and the end of each system. Default: `10\mm`.

`short-indent`
: The level of indentation for all systems in a score besides the first system. Default: `0`.

If some values are not set, defaults will be taken. Their exact value is adjusted, depending on the paper size specified. Currently, the following values are affected by this scaling:

- *left-margin*
- *right-margin*
- *top-margin*
- *bottom-margin*
- *head-separation*
- *foot-separation*
- *indent*
- *short-indent*

The settings for `line-width`, `left-margin`, `right-margin` and `paper-width` depend on each other, but they do not have to be specified completely.

```
\paper {
  left-margin = 30\mm
}
```

In this example, only `left-margin` is set. The value for `right-margin` will remain default, `line-width` is calculated automatically.

```
\paper {
  line-width = 150\mm
}
```

Here `left-margin` and `right-margin` will be set to the same value. Therefore, `line-width` is subtracted from `paper-width` and divided by two. That means systems are centered on the page, if only `line-width` is specified.

Some checks occur to ensure the values are set correctly. If the values do not match or systems would run off the page, a warning is printed and default values are set.

```
\paper {
  paper-width = 210\mm
  left-margin = 20\mm
  right-margin = 30\mm
  line-width = 100\mm
}
```

These checks can be avoided by setting `check-consistency` to false.

```
\paper {
  paper-width = 210\mm
  left-margin = 20\mm
  line-width = 200\mm
  check-consistency = ##f
}
```

Note : If `paper-width` is manually set, `line-width`, `left-margin`, `indent`, and `short-indent` may have to be adjusted as well.

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

Other layout variables

These variables can be used to adjust page layout in general.

`auto-first-page-number`
: The page breaking algorithm is affected by the first page number being odd or even. If set to true, the page breaking algorithm will decide whether to start with an odd or even number. This will result in the first page number remaining as is or being increased by one. Default: `##f`.

`blank-last-page-force`
: The penalty for ending the score on an odd-numbered page. Default: `0`.

`blank-page-force`
: The penalty for having a blank page in the middle of a score. This is not used by `ly:optimal-breaking` since it will never consider blank pages in the middle of a score. Default: `5`.

`check-consistency`
: If set to true, check whether `left-margin`, `right-margin` and `line-width` fit each other. Also make sure that their combination does not exceed the available `paper-width`. Default: `##t`.

`first-page-number`
: The value of the page number on the first page. Default: `#1`.

`max-systems-per-page`
: The maximum number of systems that will be placed on a page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`min-systems-per-page`
: The minimum number of systems that will be placed on a page. This may cause pages to be overfilled if it is made too large. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`page-breaking-system-system-spacing`
: Tricks the page breaker into thinking that `system-system-spacing` is set to something different than it really is. For example, if `page-breaking-system-system-spacing #'padding` is set to something substantially larger than `system-system-spacing #'padding`, then the page-breaker will put fewer systems on each page. Default: unset.

`page-count`
: The number of pages to be used for a score. Default: unset.

`page-spacing-weight`
: The relative importance of page (vertical) spacing and line (horizontal) spacing. High values will make page spacing more important. Default: `#10`.

`print-all-headers`
: If set to true, this will print all headers for each \score in the output. Normally only the piece and opus header variables are printed. Default: `##f`.

`print-first-page-number`
: If set to true, a page number is printed on the first page. Default: `##f`.

`print-page-number`
: If set to false, page numbers are not printed. Default: `##t`.

`ragged-bottom`
: If set to true, systems will not spread vertically across the page. This does not affect the last page. Default: `##f`.

  This should be set to true for pieces that have only two or three systems per page, for example orchestral scores.

`ragged-last`
: If set to true, the last system in the score will not fill the line width. Instead the last system ends at its natural horizontal length. Default: `##f`.

`ragged-last-bottom`
: If set to false, systems will spread vertically across the last page. Default: `##t`.

  Pieces that amply fill two pages or more should have this set to true.

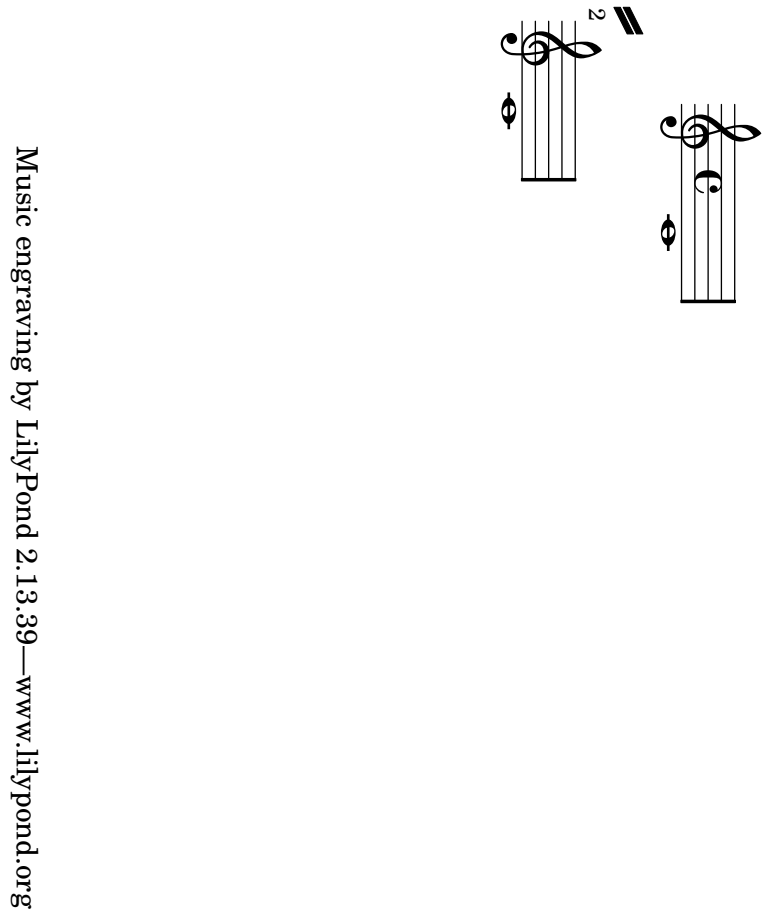
  It also affects the last page of book parts, ie parts of a book created with `\bookpart` blocks.

`ragged-right`
: If set to true, systems will not fill the line width. Instead, systems end at their natural horizontal length. Default: `##f`.

  If the score has only one system, the default value is `##t`.

`system-separator-markup`
: A markup object that is inserted between systems. This is often used for orchestral scores. Default: unset.

  The markup command `\slashSeparator` is provided as a sensible default, for example



`system-count`
: The number of systems to be used for a score. Default: unset.

`systems-per-page`
: The number of systems that should be placed on each page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`two-sided`
: If set to true, use `inner-margin`, `outer-margin` and `binding-offset` to determine margins depending on whether the page number is odd or even. This overrides `left-margin` and `right-margin`. Default: `##f`.

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

Problèmes connus et avertissements

The default page header puts the page number and the `instrument` field from the `\header` block on a line.

The titles (from the `\header{}` section) are treated as a system, so `ragged-bottom` and `ragged-last-bottom` will add space between the titles and the first system of the score.

4.2 Music layout

4.2.1 Setting the staff size

The default **staff size** is set to 20 points. This may be changed in two ways:

To set the staff size globally for all scores in a file (or in a `book` block, to be precise), use `set-global-staff-size`.

```
#(set-global-staff-size 14)
```

This sets the global default size to 14pt staff height and scales all fonts accordingly.

To set the staff size individually for each score, use

```
\score{
  ...
  \layout{
  #(layout-set-staff-size 15)
  }
}
```

The Feta font provides musical symbols at eight different sizes. Each font is tuned for a different staff size: at a smaller size the font becomes heavier, to match the relatively heavier staff lines. The recommended font sizes are listed in the following table:

font name	staff height (pt)	staff height (mm)	use
feta11	11.22	3.9	pocket scores
feta13	12.60	4.4	
feta14	14.14	5.0	
feta16	15.87	5.6	
feta18	17.82	6.3	song books
feta20	20	7.0	standard parts
feta23	22.45	7.9	
feta26	25.2	8.9	

These fonts are available in any sizes. The context property `fontSize` and the layout property `staff-space` (in Section “StaffSymbol” dans *Référence des propriétés internes*) can be used to tune the size for individual staves. The sizes of individual staves are relative to the global size.

Voir aussi

Notation Reference: ⟨undefined⟩ [Selecting notation font size], page ⟨undefined⟩.

Snippets: Section “Spacing” dans *Morceaux choisis*.

Problèmes connus et avertissements

`layout-set-staff-size` does not change the distance between the staff lines.

4.2.2 Score layout

While `\paper` contains settings that relate to the page formatting of the whole document, `\layout` contains settings for score-specific layout.

```
\layout {
  indent = 2.0\cm
  \context { \Staff
    \override VerticalAxisGroup #'minimum-Y-extent = #'(-6 . 6)
  }
  \context { \Voice
    \override TextScript #'padding = #1.0
    \override Glissando #'thickness = #3
  }
}
```

Voir aussi

Notation Reference: Section 5.1.5 [Changing context default settings], page 442.

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.3 Breaks

4.3.1 Line breaking

Line breaks are normally determined automatically. They are chosen so that lines look neither cramped nor loose, and consecutive lines have similar density. Occasionally you might want to override the automatic breaks; you can do this by specifying `\break`. This will force a line break at this point. However, line breaks can only occur at the end of ‘complete’ bars, i.e., where there are no notes or tuplets left ‘hanging’ over the bar line. If you want to have a line break where there is no bar line, you can force an invisible bar line by entering `\bar ""`, although again there must be no notes left hanging over in any of the staves at this point, or it will be ignored.

The opposite command, `\noBreak`, forbids a line break at the bar line where it is inserted.

The most basic settings influencing line spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems end at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is.

The option `ragged-last` is similar to `ragged-right`, but affects only the last line of the piece.

```
\layout {
indent = #0
line-width = #150
ragged-last = ##t
}
```

For line breaks at regular intervals use `\break` separated by skips and repeated with `\repeat`. For example, this would cause the following 28 measures (assuming 4/4 time) to be broken every 4 measures, and only there:

```
<< \repeat unfold 7 {
         s1 \noBreak s1 \noBreak
         s1 \noBreak s1 \break }
   the real music
>>
```

A linebreaking configuration can be saved as a `.ly` file automatically. This allows vertical alignments to be stretched to fit pages in a second formatting run. This is fairly new and complicated. More details are available in Section “Spacing” dans *Morceaux choisis*.

Commandes prédéfinies

`\break`, `\noBreak`.

Voir aussi
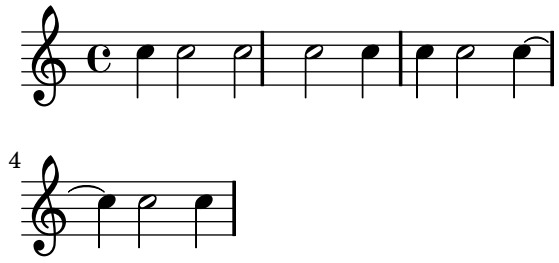
Internals Reference: Section “LineBreakEvent” dans *Référence des propriétés internes*.

Snippets: Section “Spacing” dans *Morceaux choisis*.

Problèmes connus et avertissements

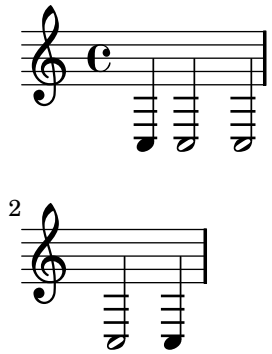
Line breaks can only occur if there is a ‘proper’ bar line. A note which is hanging over a bar line is not proper, such as

```
c4 c2 << c2 {s4 \break } >>  % this does nothing
c2 c4 |           % a break here would work
c4 c2 c4 ~ \break % as does this break
c4 c2 c4
```



This can be avoided by removing the `Forbid_line_break_engraver`. Note that manually forced line breaks have to be added in parallel with the music.

```
\new Voice \with {
  \remove Forbid_line_break_engraver
} {
  c4 c2 << c2 {s4 \break } >>  % now the break is allowed
  c2 c4
}
```



Similarly, line breaks are normally forbidden when beams cross bar lines. This behavior can be changed by setting `\override Beam #'breakable = ##t`.

4.3.2 Page breaking

The default page breaking may be overridden by inserting `\pageBreak` or `\noPageBreak` commands. These commands are analogous to `\break` and `\noBreak`. They should be inserted at a bar line. These commands force and forbid a page-break from happening. Of course, the `\pageBreak` command also forces a line break.

The `\pageBreak` and `\noPageBreak` commands may also be inserted at top-level, between scores and top-level markups.

There are also analogous settings to `ragged-right` and `ragged-last` which have the same effect on vertical spacing: `ragged-bottom` and `ragged-last-bottom`. If set to `##t` the systems on all pages or just the last page respectively will not be justified vertically.

For more details see Section 4.4 [Vertical spacing], page 407.

Page breaks are computed by the `page-breaking` function. LilyPond provides three algorithms for computing page breaks, `ly:optimal-breaking`, `ly:page-turn-breaking` and `ly:minimal-breaking`. The default is `ly:optimal-breaking`, but the value can be changed in the `\paper` block:

```
\paper{
  #(define page-breaking ly:page-turn-breaking)
}
```

When a book has many scores and pages, the page breaking problem may be difficult to solve, requiring large processing time and memory. To ease the page breaking process, `\bookpart` blocks are used to divide the book into several parts: the page breaking occurs separately on each part. Different page breaking functions may also be used in different book parts.

```
\bookpart {
  \header {
    subtitle = "Preface"
  }
  \paper {
     %% In a part consisting mostly of text,
     %% ly:minimal-breaking may be preferred
     #(define page-breaking ly:minimal-breaking)
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In this part, consisting of music, the default optimal
  %% page breaking function is used.
  \header {
    subtitle = "First movement"
  }
  \score { ... }
  ...
}
```

Commandes prédéfinies

`\pageBreak`, `\noPageBreak`.

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.3.3 Optimal page breaking

The `ly:optimal-breaking` function is LilyPond's default method of determining page breaks. It attempts to find a page breaking that minimizes cramping and stretching, both horizontally and vertically. Unlike `ly:page-turn-breaking`, it has no concept of page turns.

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.3.4 Optimal page turning

Often it is necessary to find a page breaking configuration so that there is a rest at the end of every second page. This way, the musician can turn the page without having to miss notes. The `ly:page-turn-breaking` function attempts to find a page breaking minimizing cramping and stretching, but with the additional restriction that it is only allowed to introduce page turns in specified places.

There are two steps to using this page breaking function. First, you must enable it in the `\paper` block, as explained in Section 4.3.2 [Page breaking], page 401. Then you must tell the function where you would like to allow page breaks.

There are two ways to achieve the second step. First, you can specify each potential page turn manually, by inserting `\allowPageTurn` into your input file at the appropriate places.

If this is too tedious, you can add a `Page_turn_engraver` to a Staff or Voice context. The `Page_turn_engraver` will scan the context for sections without notes (note that it does not scan for rests; it scans for the absence of notes. This is so that single-staff polyphony with rests in one of the parts does not throw off the `Page_turn_engraver`). When it finds a sufficiently long section without notes, the `Page_turn_engraver` will insert an `\allowPageTurn` at the final bar line in that section, unless there is a ‘special’ bar line (such as a double bar), in which case the `\allowPageTurn` will be inserted at the final ‘special’ bar line in the section.

The `Page_turn_engraver` reads the context property `minimumPageTurnLength` to determine how long a note-free section must be before a page turn is considered. The default value for `minimumPageTurnLength` is `#(ly:make-moment 1 1)`. If you want to disable page turns, you can set it to something very large.

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % a page turn will be allowed here
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5 2)
  R1 | % a page turn will not be allowed here
  a4 b r2 |
  R1*2 | % a page turn will be allowed here
  a1
}
```

The `Page_turn_engraver` detects volta repeats. It will only allow a page turn during the repeat if there is enough time at the beginning and end of the repeat to turn the page back. The `Page_turn_engraver` can also disable page turns if the repeat is very short. If you set the context property `minimumRepeatLengthForPageTurn` then the `Page_turn_engraver` will only allow turns in repeats whose duration is longer than this value.

The page turning commands, `\pageTurn`, `\noPageTurn` and `\allowPageTurn`, may also be used at top-level, between scores and top-level markups.

Commandes prédéfinies

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Voir aussi

Snippets: Section "Spacing" dans *Morceaux choisis*.

Problèmes connus et avertissements

There should only be one `Page_turn_engraver` in a score. If there is more than one, they will interfere with each other.

4.3.5 Minimal page breaking

The `ly:minimal-breaking` function performs minimal computations to calculate the page breaking: it fills a page with as many systems as possible before moving to the next one. Thus, it may be preferred for scores with many pages, where the other page breaking functions could be too slow or memory demanding, or a lot of texts. It is enabled using:

```
\paper {
  #(define page-breaking ly:minimal-breaking)
}
```

Voir aussi

Snippets: Section "Spacing" dans *Morceaux choisis*.

4.3.6 Explicit breaks

Lily sometimes rejects explicit `\break` and `\pageBreak` commands. There are two commands to override this behavior:
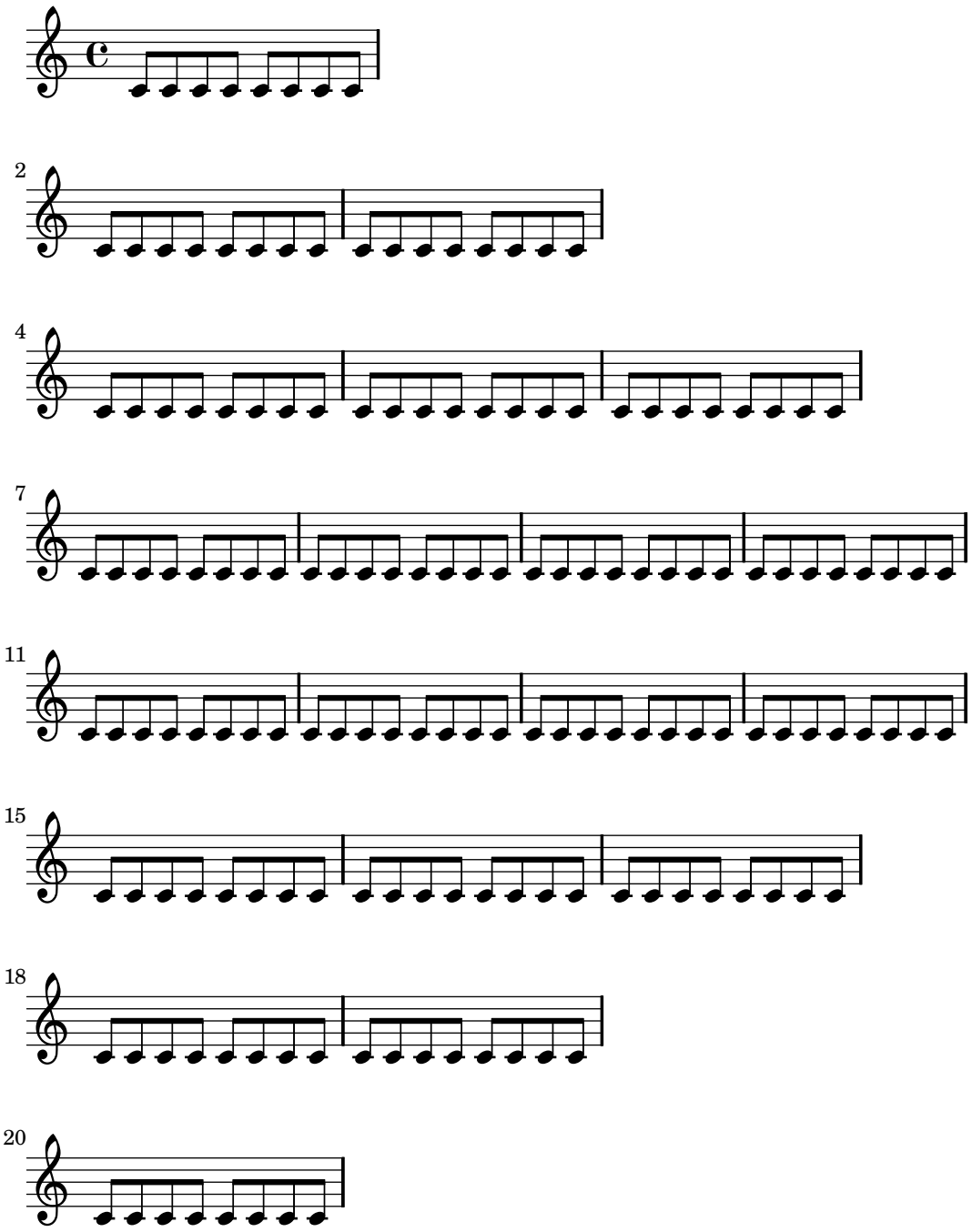
```
\override NonMusicalPaperColumn #'line-break-permission = ##f
\override NonMusicalPaperColumn #'page-break-permission = ##f
```

When `line-break-permission` is overridden to false, Lily will insert line breaks at explicit `\break` commands and nowhere else. When `page-break-permission` is overridden to false, Lily will insert page breaks at explicit `\pageBreak` commands and nowhere else.

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

\score {
  \new Staff {
    \repeat unfold 2 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 4 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 6 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 8 { c'8 c'8 c'8 c'8 } \pageBreak
    \repeat unfold 8 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 6 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 4 { c'8 c'8 c'8 c'8 } \break
    \repeat unfold 2 { c'8 c'8 c'8 c'8 }
    }
```

```
  \layout {
    \context {
      \Score
      \override NonMusicalPaperColumn #'line-break-permission = ##f
      \override NonMusicalPaperColumn #'page-break-permission = ##f
    }
  }
}
```



Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.
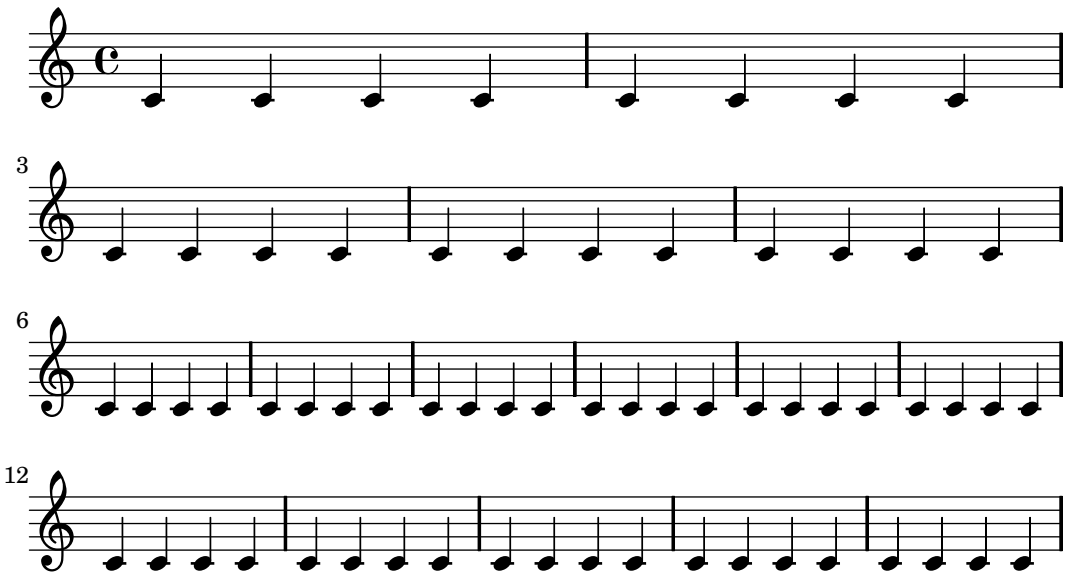
4.3.7 Using an extra voice for breaks

Line- and page-breaking information usually appears within note entry directly.

```
\score {
  \new Staff {
    \repeat unfold 2 { c'4 c'4 c'4 c'4 }
    \break
    \repeat unfold 3 { c'4 c'4 c'4 c'4 }
  }
}
```

This makes `\break` and `\pageBreak` commands easy to enter but mixes music entry with information that specifies how music should lay out on the page. You can keep music entry and line- and page-breaking information in two separate places by introducing an extra voice to contain the breaks. This extra voice contains only skips together with `\break`, `pageBreak` and other breaking layout information.

```
\score {
  \new Staff <<
    \new Voice {
      s1 * 2 \break
      s1 * 3 \break
      s1 * 6 \break
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { c'4 c'4 c'4 c'4 }
      \repeat unfold 3 { c'4 c'4 c'4 c'4 }
      \repeat unfold 6 { c'4 c'4 c'4 c'4 }
      \repeat unfold 5 { c'4 c'4 c'4 c'4 }
    }
  >>
}
```
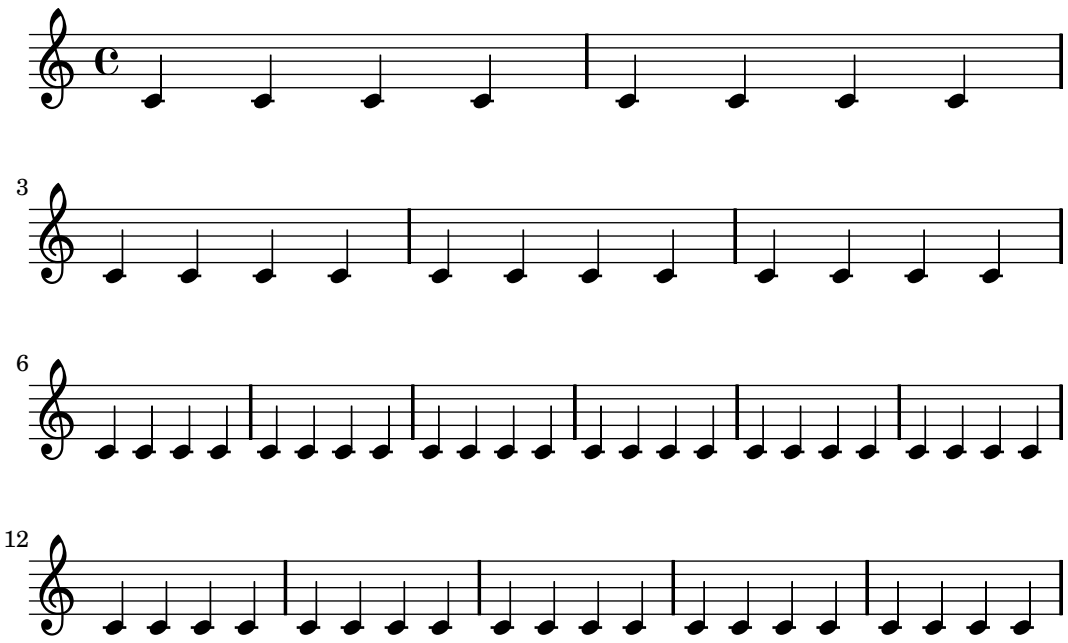


This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of `NonMusicalPaperColumnGrob`, as explained in Section 4.4 [Vertical spacing], page 407.

```
\score {
  \new Staff <<
    \new Voice {
      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 0))
      s1 * 2 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 35))
      s1 * 3 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 70))
      s1 * 6 \break

      \overrideProperty "Score.NonMusicalPaperColumn"
        #'line-break-system-details #'((Y-offset . 105))
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { c'4 c'4 c'4 c'4 }
      \repeat unfold 3 { c'4 c'4 c'4 c'4 }
      \repeat unfold 6 { c'4 c'4 c'4 c'4 }
      \repeat unfold 5 { c'4 c'4 c'4 c'4 }
    }
  >>
}
```



Voir aussi

Notation Reference: Section 4.4 [Vertical spacing], page 407.

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.4 Vertical spacing

Vertical spacing is controlled by three things: the amount of space available (i.e., paper size and margins), the amount of space between systems, and the amount of space between staves inside a system.

4.4.1 Flexible vertical spacing within systems

Three separate mechanisms control the flexible vertical spacing within systems, one for each of the following categories:

- *ungrouped staves*,
- *grouped staves* (staves within a staff-group such as `ChoirStaff`, etc.), and
- *non-staff lines* (such as `Lyrics`, `ChordNames`, etc.).

The height of each system is determined in two steps. First, all of the staves are spaced according to the amount of space available. Then, the non-staff lines are distributed between the staves.

Note that the spacing mechanisms discussed in this section only control the vertical spacing of staves and non-staff lines within individual systems. The vertical spacing between separate systems, scores, markups, and margins is controlled by `\paper` variables, which are discussed in [Flexible vertical dimensions], page 391.

Within-system spacing properties

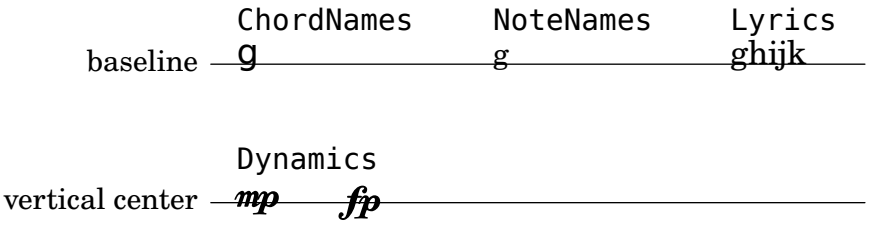
The within-system vertical spacing mechanisms are controlled by two sets of grob properties. The first set is associated with the `VerticalAxisGroup` grob, which is created by all staves and non-staff lines. The second set is associated with the `StaffGrouper` grob, which can be created by staff-groups, but only if explicitly called. These properties are described individually at the end of this section.

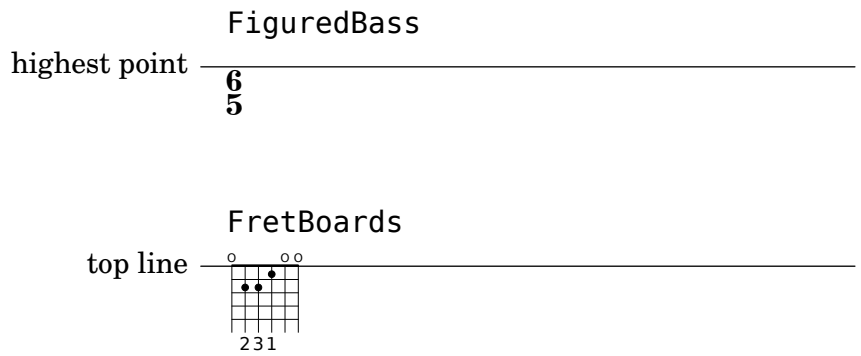
The names of these properties (except for `staff-affinity`) follow the format *`item1`*`-`*`item2`*`-spacing`, where *`item1`* and *`item2`* are the items to be spaced. Note that *`item2`* is not necessarily below *`item1`*; for example, `nonstaff-relatedstaff-spacing` will measure upwards from the non-staff line if `staff-affinity` is `#UP`.

Each distance is measured between the *reference points* of the two items. The reference point for a staff is the vertical center of its `StaffSymbol` (i.e. the middle line if `line-count` is odd; the middle space if `line-count` is even). The reference points for individual non-staff lines are given in the following table:

Non-staff line	**Reference point**
`ChordNames`	baseline
`NoteNames`	baseline
`Lyrics`	baseline
`Dynamics`	vertical center
`FiguredBass`	highest point
`FretBoards`	top line

In the following image, horizontal lines indicate the positions of these reference points:





Each of the vertical spacing grob properties (except `staff-affinity`) is stored as an alist (association list), and each uses the same alist structure as the `\paper` spacing variables discussed in [Flexible vertical dimensions], page 391. Specific methods for modifying alists are discussed in Section 5.3.6 [Modifying alists], page 455. Grob properties should be adjusted with an `\override` inside a `\score` or `\layout` block, and not inside a `\paper` block.

The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely re-defines the property:

```
\new Staff \with {
  \override VerticalAxisGroup #'staff-staff-spacing #'space = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup #'staff-staff-spacing =
    #'((padding . 1)
       (space . 10)
       (minimum-distance . 9)
       (stretchability . 10))
} { ... }
```

To change any spacing settings globally, put them in the `\layout` block:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup #'staff-staff-spacing #'space = #10
  }
}
```

Standard settings for the vertical spacing grob properties are listed in Section "VerticalAxisGroup" dans *Référence des propriétés internes* and Section "StaffGrouper" dans *Référence des propriétés internes*. Default overrides for specific types of non-staff lines are listed in the relevant context descriptions in Section "Contexts" dans *Référence des propriétés internes*.

Properties of the `VerticalAxisGroup` grob

`VerticalAxisGroup` properties are typically adjusted with an `\override` at the `Staff` level (or equivalent).

`staff-staff-spacing`
: The distance between the current staff and the staff just below it in the same system, even if one or more non-staff lines (such as `Lyrics`) are placed between the two staves. Does not apply to the bottom staff of a system. This replaces any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. If this is unset, and there are no `StaffGrouper` properties to inherit, the `default-staff-staff-spacing` property is used.

`default-staff-staff-spacing`
: The settings to use for `staff-staff-spacing` when it is unset. This applies to ungrouped staves and to grouped staves that do not inherit settings from the `StaffGrouper` grob.

`staff-affinity`
: 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. Adjacent non-staff lines should have non-increasing `staff-affinity` from top to bottom, e.g. a non-staff line set to `UP` should not immediately follow one that is set to `DOWN`. Non-staff lines at the top of a system should use `DOWN`; those at the bottom should use `UP`. 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.

`nonstaff-relatedstaff-spacing`
: 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.

`nonstaff-nonstaff-spacing`
: 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`.

`nonstaff-unrelatedstaff-spacing`
: The distance between the current non-staff line and the 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`. This can be used, for example, to require a minimum amount of padding between a `Lyrics` line and the staff to which it does not belong.

Properties of the `StaffGrouper` grob

`StaffGrouper` properties are typically adjusted with an `\override` at the `StaffGroup` level (or equivalent).

`staff-staff-spacing`
: The distance between consecutive staves within the current staff-group. The `staff-staff-spacing` property of an individual staff's `VerticalAxisGroup` grob will be used instead for any staves in the staff-group that have it set. Also see `default-staff-staff-spacing`.

`staffgroup-staff-spacing`
: 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 (such as `Lyrics`) exist between the two staves. Does not apply to the bottom staff of a system. The `staff-staff-spacing` property of an individual staff's `VerticalAxisGroup` grob will be used instead for any staves in the staff-group that have it set. Also see `default-staff-staff-spacing`.

Voir aussi

Installed files: '`ly/engraver-init.ly`', '`scm/define-grobs.scm`'.

Internals Reference: Section “Contexts” dans *Référence des propriétés internes*, Section “VerticalAxisGroup” dans *Référence des propriétés internes*, Section “StaffGrouper” dans *Référence des propriétés internes*.

Spacing of ungrouped staves

Staves (such as `Staff`, `DrumStaff`, `TabStaff`, etc.) are contexts that can contain one or more voice contexts, but cannot contain any other staves.

The following properties affect the spacing of *ungrouped* staves:

- `VerticalAxisGroup` properties:
  - `staff-staff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], page 407.

Additional properties are involved for staves that are part of a staff-group; see [Spacing of grouped staves], page 411.

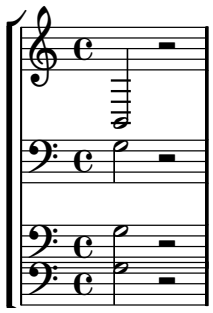
The following example shows how the `staff-staff-spacing` property can affect the spacing of ungrouped staves:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup #'staff-staff-spacing =
      #'((padding . 1)
         (space . 8)
         (minimum-distance . 7))
  }
}

\new StaffGroup <<
  % The very low note here needs more room than 'space can
  % provide, so the distance between this staff and the next is
  % determined by 'padding.
  \new Staff { b,2 r | }

  % Here, 'space 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 'space.
  \new Staff { \clef bass g2 r | }

  % By setting 'padding to a negative value, staves can be made to
  % collide.  The lowest acceptable value for 'space is 0.
  \new Staff \with {
    \override VerticalAxisGroup #'staff-staff-spacing =
      #'((padding . -10)
         (space . 3.5))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



Voir aussi

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Section “Spacing” dans *Morceaux choisis*.

Internals Reference: Section “VerticalAxisGroup” dans *Référence des propriétés internes*.

Spacing of grouped staves

In orchestral and other large scores, it is common to place staves in groups. The space between groups is typically larger than the space between staves of the same group.

Staff-groups (such as `StaffGroup`, `ChoirStaff`, etc.) are contexts that can contain one or more staves simultaneously.

The following properties affect the spacing of staves inside staff-groups:

- `VerticalAxisGroup` properties:
  - `staff-staff-spacing`
  - `default-staff-staff-spacing`
- `StaffGrouper` properties:
  - `staff-staff-spacing`
  - `staffgroup-staff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], page 407.

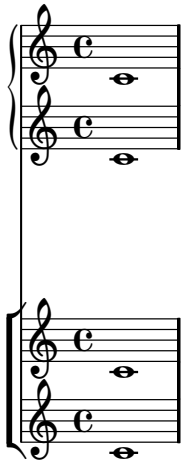
The following example shows how properties of the `StaffGrouper` grob can affect the spacing of grouped staves:

```
\layout {
  \context {
    \Score
    \override StaffGrouper #'staff-staff-spacing #'padding = #0
    \override StaffGrouper #'staff-staff-spacing #'space = #1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper #'staffgroup-staff-spacing #'space = #20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>
```

```
>>
```



Voir aussi

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Section “Spacing” dans *Morceaux choisis*.

Internals Reference: Section “VerticalAxisGroup” dans *Référence des propriétés internes*, Section “StaffGrouper” dans *Référence des propriétés internes*.

Spacing of non-staff lines

Non-staff lines (such as `Lyrics`, `ChordNames`, etc.) are contexts whose layout objects are engraved like staves (i.e. in horizontal lines within systems). Specifically, non-staff lines are non-staff contexts that create the `VerticalAxisGroup` layout object.

The following properties affect the spacing of non-staff lines:

- `VerticalAxisGroup` properties:
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

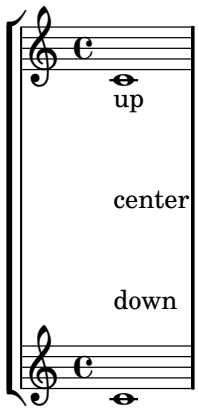
These grob properties are described individually above; see [Within-system spacing properties], page 407.

The following example shows how the `nonstaff-nonstaff-spacing` property can affect the spacing of consecutive non-staff lines. Here, by setting the `stretchability` key to a very high value, the lyrics are able to stretch much more than usual:

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
      #'nonstaff-nonstaff-spacing #'stretchability = #1000
  }
}

\new StaffGroup
<<
  \new Staff \with {
```

```
    \override VerticalAxisGroup #'staff-staff-spacing = #'((space . 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

Installed Files: ‘`ly/engraver-init.ly`’, ‘`scm/define-grobs.scm`’.

Snippets: Section “Spacing” dans *Morceaux choisis*.

Internals Reference: Section “Contexts” dans *Référence des propriétés internes*, Section “VerticalAxisGroup” dans *Référence des propriétés internes*.

4.4.2 Explicit staff and system positioning

One way to understand the `VerticalAxisGroup` and `\paper` settings explained in the previous two sections is as a collection of different settings that primarily concern the amount of vertical padding different staves and systems running down the page.

It is possible to approach vertical spacing in a different way using `NonMusicalPaperColumn #'line-break-system-details`. Where `VerticalAxisGroup` and `\paper` settings specify vertical padding, `NonMusicalPaperColumn #'line-break-system-details` specifies exact vertical positions on the page.

`NonMusicalPaperColumn #'line-break-system-details` accepts an associative list of three different settings:

- `X-offset`
- `Y-offset`
- `alignment-distances`

Grob overrides, including the overrides for `NonMusicalPaperColumn` below, can occur in any of three different places in an input file:

- in the middle of note entry directly

- in a `\context` block
- in the `\with` block

When we override `NonMusicalPaperColumn`, we use the usual `\override` command in `\context` blocks and in the `\with` block. On the other hand, when we override `NonMusicalPaperColumn` in the middle of note entry, use the special `\overrideProperty` command. Here are some example `NonMusicalPaperColumn` overrides with the special `\overrideProperty` command:

```
\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20))
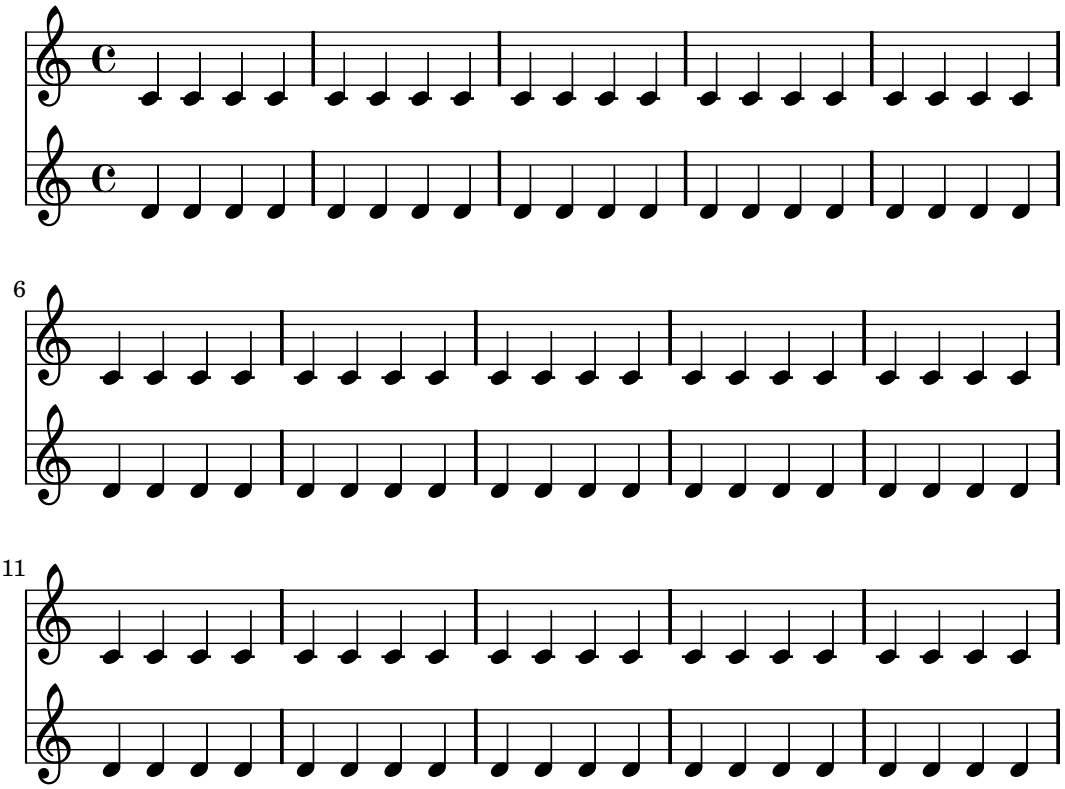
\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((Y-offset . 40))

\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20) (Y-offset . 40))

\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((alignment-distances . (15)))
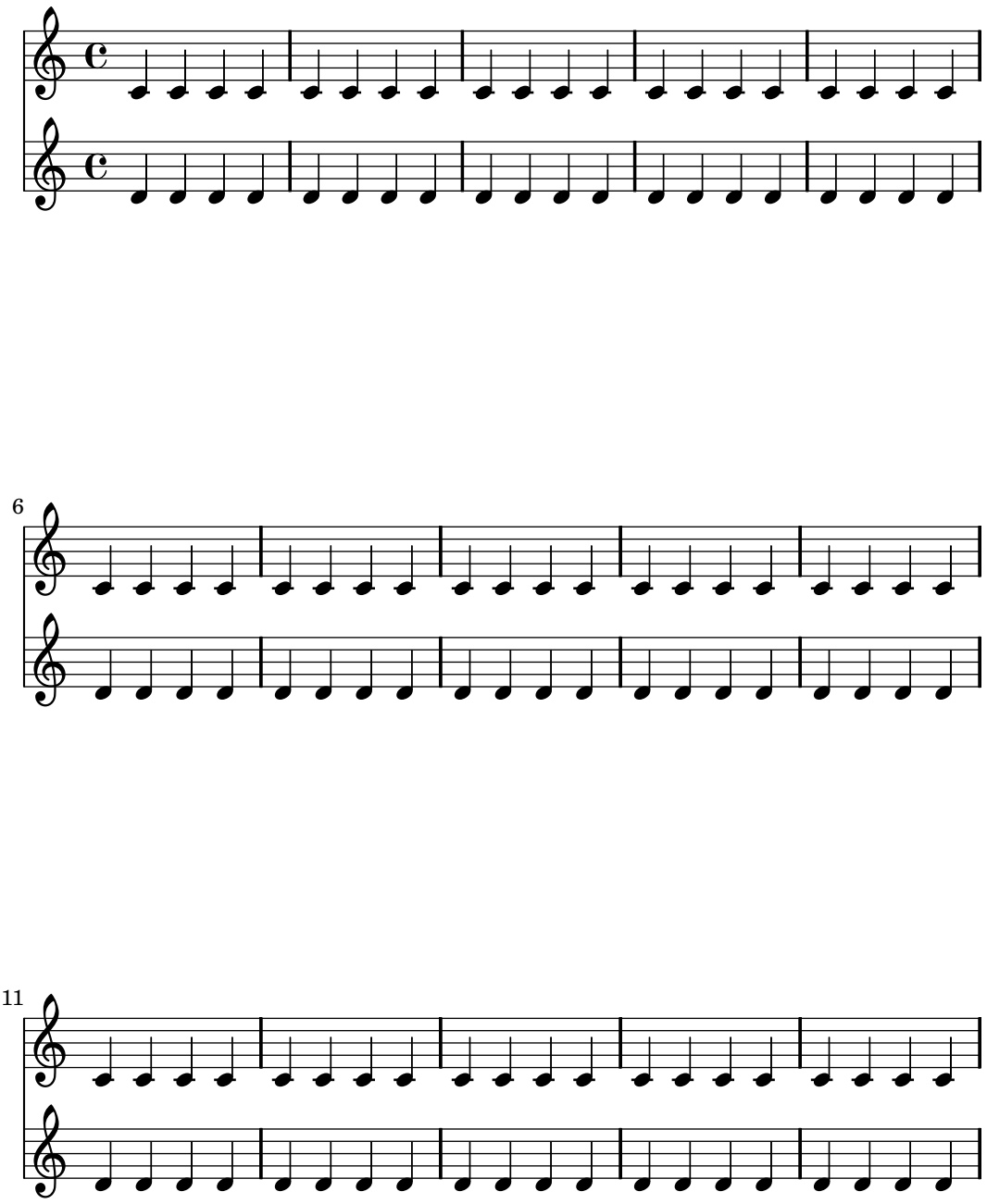
\overrideProperty NonMusicalPaperColumn
  #'line-break-system-details #'((X-offset . 20) (Y-offset . 40)
                                 (alignment-distances . (15)))
```

To understand how each of these different settings work, we begin by looking at an example that includes no overrides at all.
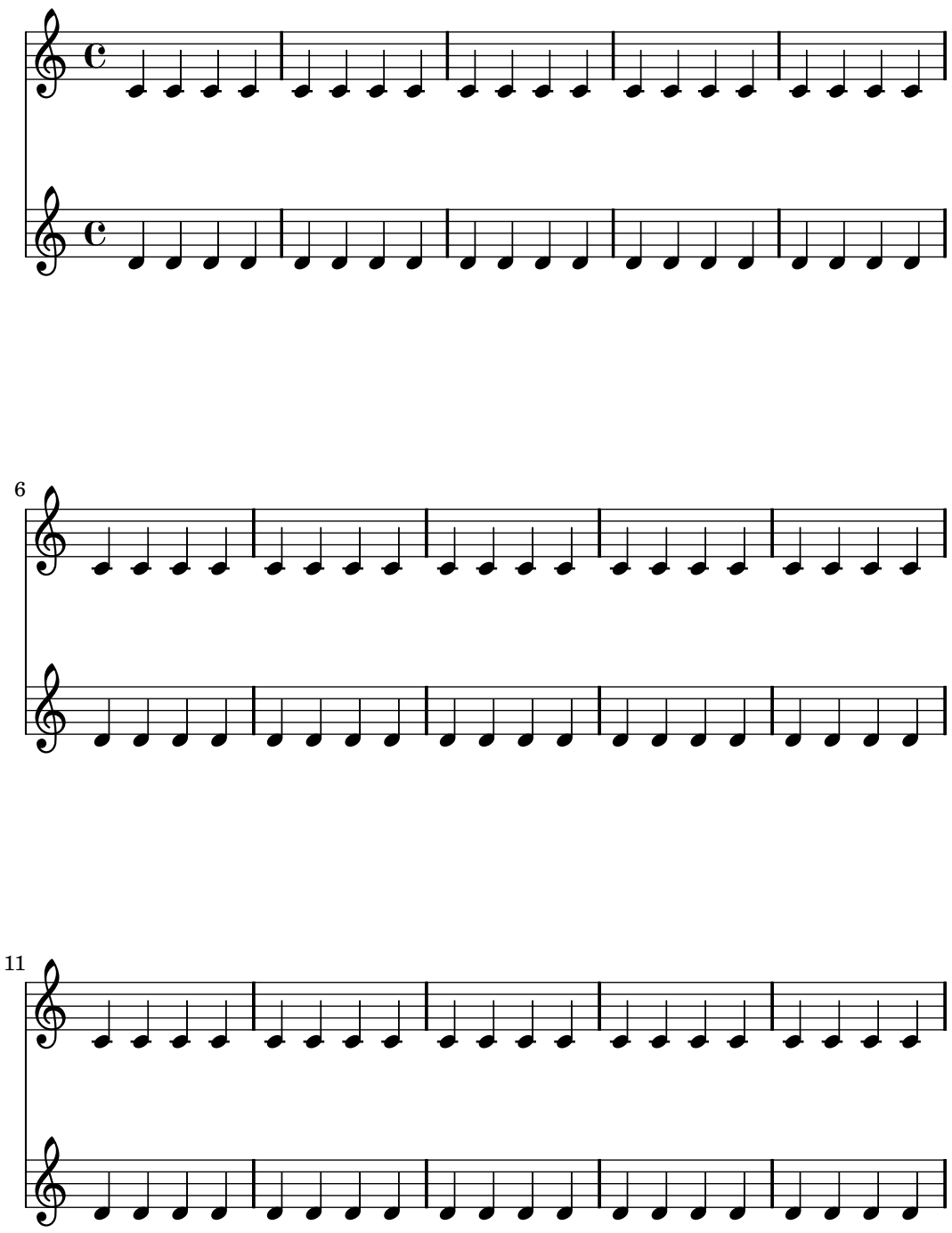


This score isolates line- and page-breaking information in a dedicated voice. This technique of creating a breaks voice will help keep layout separate from music entry as our example becomes more complicated. See Section 4.3.7 [Using an extra voice for breaks], page 405.

Explicit `\breaks` evenly divide the music into six measures per line. Vertical spacing results from LilyPond's defaults. To set the vertical startpoint of each system explicitly, we can set the `Y-offset` pair in the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob:
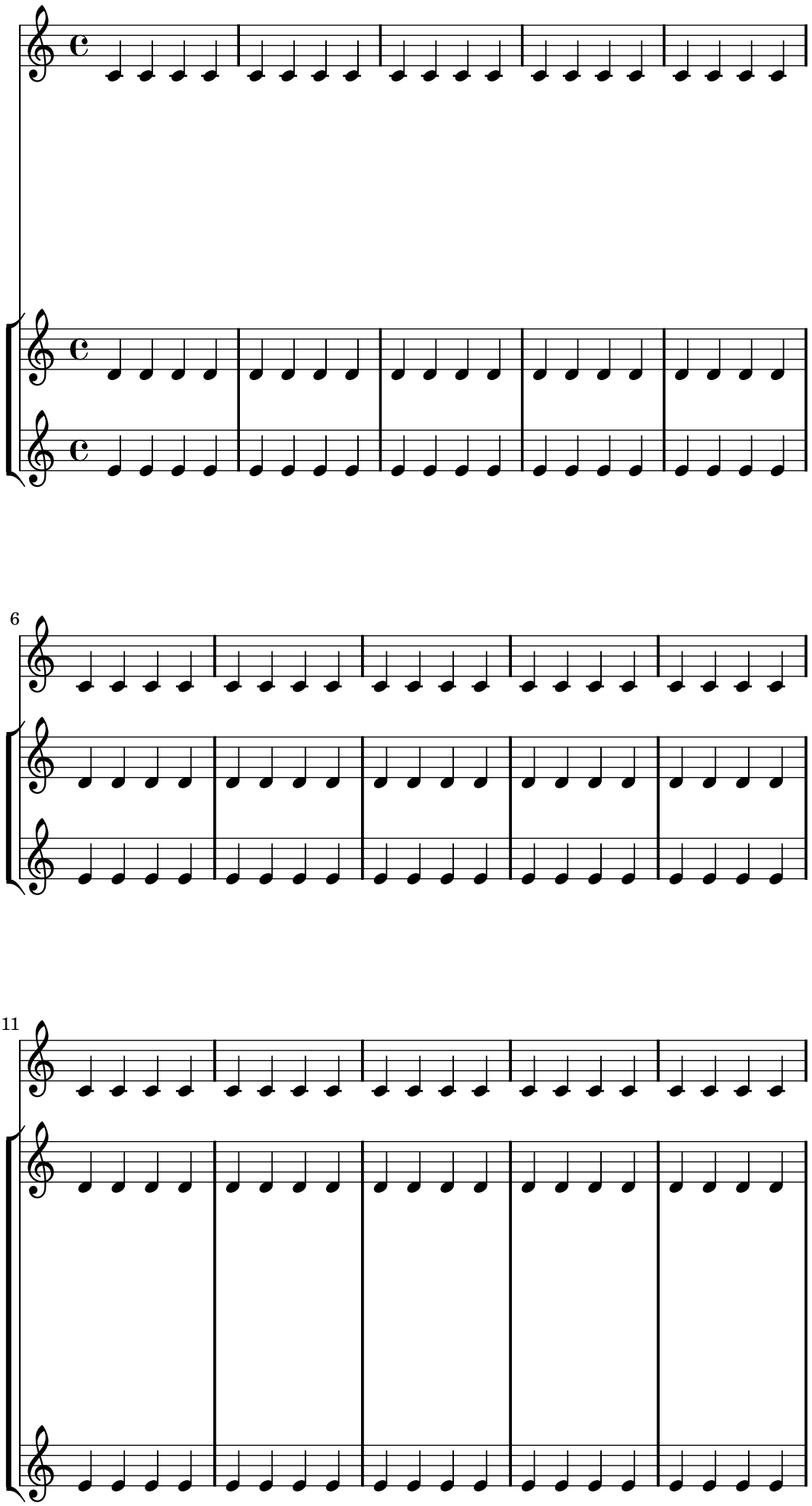


Note that `line-break-system-details` takes an associative list of potentially many values, but that we set only one value here. Note, too, that the `Y-offset` property here determines the exact vertical position on the page at which each new system will render.

Now that we have set the vertical startpoint of each system explicitly, we can also set the vertical distances between staves within each system manually. We do this using the `alignment-distances` subproperty of `line-break-system-details`.



Note that here we assign two different values to the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob. Though the `line-break-system-details` attribute alist accepts many additional spacing parameters (including, for example, a corresponding `X-offset` pair), we need only set the `Y-offset` and `alignment-distances` pairs to control the vertical startpoint of every system and every staff. Finally, note that `alignment-distances` specifies the vertical positioning of staves but not of staff groups.


6
11

Some points to consider:

- When using `alignment-distances`, lyrics and other non-staff lines do not count as a staff.
- The units of the numbers passed to `X-offset`, `Y-offset` and `alignment-distances` are interpreted as multiples of the distance between adjacent staff lines. Positive values move staves and lyrics up, negative values move staves and lyrics down.
- Because the `NonMusicalPaperColumn #'line-break-system-details` settings given here allow the positioning of staves and systems anywhere on the page, it is possible to violate paper or margin boundaries or even to print staves or systems on top of one another. Reasonable values passed to these different settings will avoid this.

Voir aussi

Snippets: Section "Spacing" dans *Morceaux choisis*.

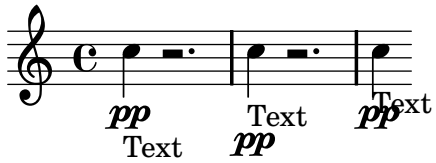
4.4.3 Vertical collision avoidance

Intuitively, there are some objects in musical notation that belong to the staff and there are other objects that should be placed outside the staff. Objects belonging outside the staff include things such as rehearsal marks, text and dynamic markings (from now on, these will be called outside-staff objects). LilyPond's rule for the vertical placement of outside-staff objects is to place them as close to the staff as possible but not so close that they collide with another object.

LilyPond uses the `outside-staff-priority` property to determine whether a grob is an outside-staff object: if `outside-staff-priority` is a number, the grob is an outside-staff object. In addition, `outside-staff-priority` tells LilyPond in which order the objects should be placed.

First, LilyPond places all the objects that do not belong outside the staff. Then it sorts the outside-staff objects according to their `outside-staff-priority` (in increasing order). One by one, LilyPond takes the outside-staff objects and places them so that they do not collide with any objects that have already been placed. That is, if two outside-staff grobs are competing for the same space, the one with the lower `outside-staff-priority` will be placed closer to the staff.

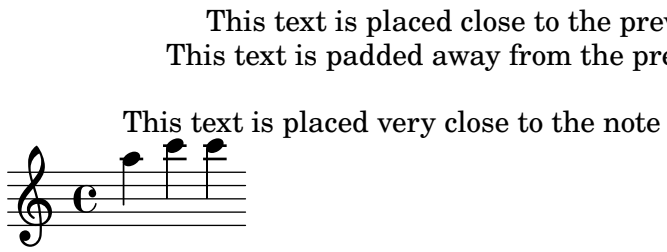
```
c4_"Text"\pp
r2.
\once \override TextScript #'outside-staff-priority = #1
c4_"Text"\pp % this time the text will be closer to the staff
r2.
% by setting outside-staff-priority to a non-number,
% we disable the automatic collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
\once \override DynamicLineSpanner #'outside-staff-priority = ##f
c4_"Text"\pp % now they will collide
```



The vertical padding between an outside-staff object and the previously-positioned grobs can be controlled with `outside-staff-padding`.

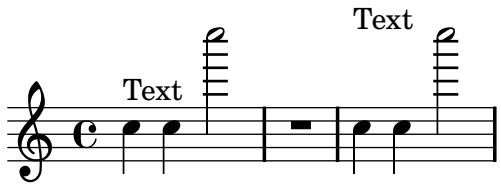
```
\once \override TextScript #'outside-staff-padding = #0
a'^"This text is placed very close to the note"
```

```
\once \override TextScript #'outside-staff-padding = #3
c^"This text is padded away from the previous text"
c^"This text is placed close to the previous text"
```



By default, outside-staff objects are placed only to avoid a horizontal collision with previously-positioned grobs. This can lead to situations in which objects are placed very close to each other horizontally. The vertical spacing between staves can also be set so that outside staff objects are interleaved. Setting `outside-staff-horizontal-padding` causes an object to be offset vertically so that such a situation doesn't occur.

```
% the markup is too close to the following note
c4^"Text"
c4
c''2
% setting outside-staff-horizontal-padding fixes this
R1
\once \override TextScript #'outside-staff-horizontal-padding = #1
c,,4^"Text"
c4
c''2
```



Voir aussi

Snippets: Section "Spacing" dans *Morceaux choisis*.
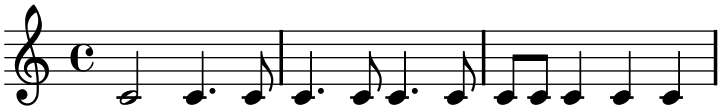
4.5 Horizontal spacing

4.5.1 Horizontal spacing overview

The spacing engine translates differences in durations into stretchable distances ('springs') of differing lengths. Longer durations get more space, shorter durations get less. The shortest durations get a fixed amount of space (which is controlled by `shortest-duration-space` in the Section "SpacingSpanner" dans *Référence des propriétés internes* object). The longer the duration, the more space it gets: doubling a duration adds a fixed amount (this amount is controlled by `spacing-increment`) of space to the note.

For example, the following piece contains lots of half, quarter, and 8th notes; the eighth note is followed by 1 note head width (NHW). The quarter note is followed by 2 NHW, the half by 3 NHW, etc.

```
c2 c4. c8 c4. c8 c4. c8 c8
c8 c4 c4 c4
```



Normally, `spacing-increment` is set to 1.2 staff space, which is approximately the width of a note head, and `shortest-duration-space` is set to 2.0, meaning that the shortest note gets 2.4 staff space (2.0 times the `spacing-increment`) of horizontal space. This space is counted from the left edge of the symbol, so the shortest notes are generally followed by one NHW of space.
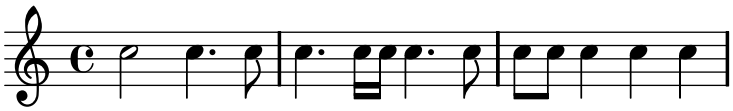
If one would follow the above procedure exactly, then adding a single 32nd note to a score that uses 8th and 16th notes, would widen up the entire score a lot. The shortest note is no longer a 16th, but a 32nd, thus adding 1 NHW to every note. To prevent this, the shortest duration for spacing is not the shortest note in the score, but rather the one which occurs most frequently.

The most common shortest duration is determined as follows: in every measure, the shortest duration is determined. The most common shortest duration is taken as the basis for the spacing, with the stipulation that this shortest duration should always be equal to or shorter than an 8th note. The shortest duration is printed when you run `lilypond` with the `--verbose` option.
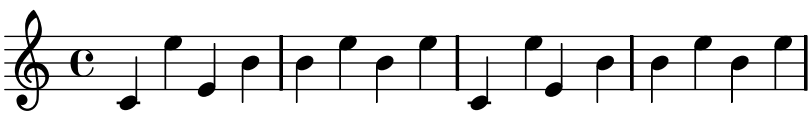
These durations may also be customized. If you set the `common-shortest-duration` in Section “SpacingSpanner” dans *Référence des propriétés internes*, then this sets the base duration for spacing. The maximum duration for this base (normally an 8th), is set through `base-shortest-duration`.

Notes that are even shorter than the common shortest note are followed by a space that is proportional to their duration relative to the common shortest note. So if we were to add only a few 16th notes to the example above, they would be followed by half a NHW:

```
c2 c4. c8 c4. c16[ c] c4. c8 c8 c8 c4 c4 c4
```



In the *Essay on automated music engraving*, it was explained that stem directions influence spacing (see Section “Optical spacing” dans *Essai*). This is controlled with the `stem-spacing-correction` property in the Section “NoteSpacing” dans *Référence des propriétés internes*, object. These are generated for every Section “Voice” dans *Référence des propriétés internes* context. The `StaffSpacing` object (generated in Section “Staff” dans *Référence des propriétés internes* context) contains the same property for controlling the stem/bar line spacing. The following example shows these corrections, once with default settings, and once with exaggerated corrections:



Proportional notation is supported; see Section 4.5.5 [Proportional notation], page 424.

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

Internals Reference: 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

There is no convenient mechanism to manually override spacing. The following work-around may be used to insert extra space into a score, adjusting the padding value as necessary.

```
\override Score.NonMusicalPaperColumn #'padding = #10
```
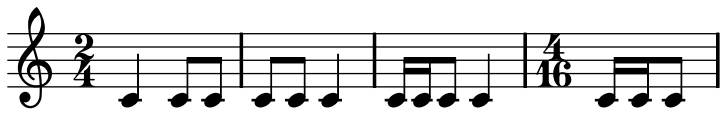
No work-around exists for decreasing the amount of space.

4.5.2 New spacing area

New sections with different spacing parameters can be started with `newSpacingSection`. This is useful when there are sections with a different notions of long and short notes.

In the following example, the time signature change introduces a new section, and hence the 16ths notes are spaced wider.

```
\time 2/4
c4 c8 c
c8 c c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```



The `\newSpacingSection` command creates a new `SpacingSpanner` object, and hence new `\overrides` may be used in that location.

Voir aussi
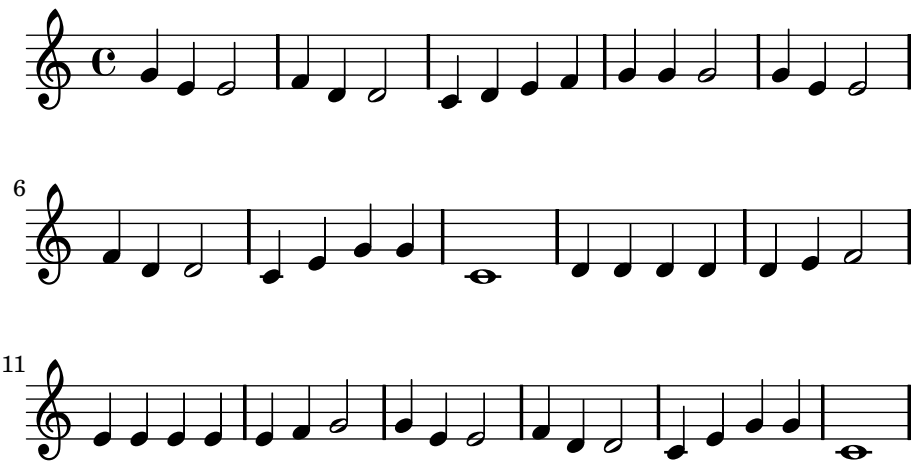
Snippets: Section “Spacing” dans *Morceaux choisis*.

Internals Reference: Section “SpacingSpanner” dans *Référence des propriétés internes*.
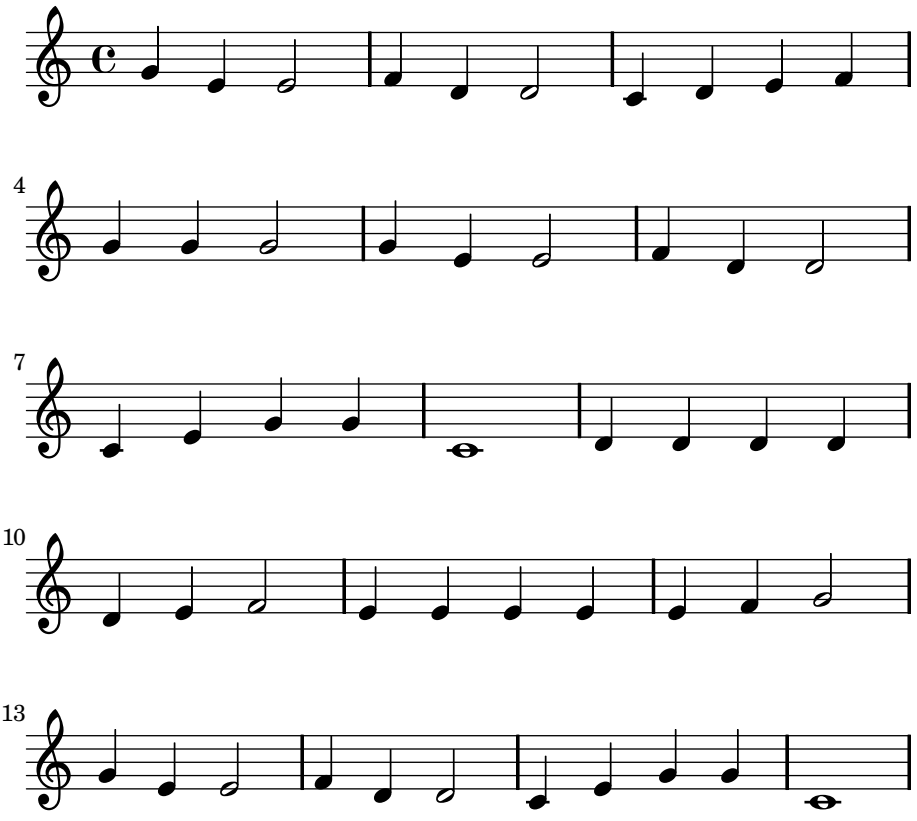
4.5.3 Changing horizontal spacing

Horizontal spacing may be altered with the `base-shortest-duration` property. Here we compare the same music; once without altering the property, and then altered. Larger values of `ly:make-moment` will produce smaller music. Note that `ly:make-moment` constructs a duration, so `1 4` is a longer duration than `1 16`.

```
\score {
  \relative c'' {
    g4 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 |
  }
}
```



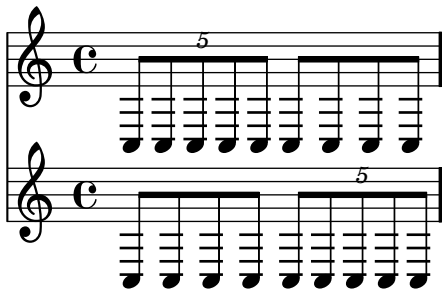
```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner
        #'base-shortest-duration = #(ly:make-moment 1 16)
    }
  }
}
```



Morceaux choisis

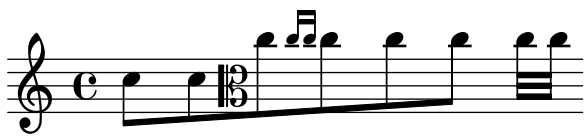
By default, spacing in tuplets depends on various non-duration factors (such as accidentals, clef changes, etc). To disregard such symbols and force uniform equal-duration spacing, use `Score.SpacingSpanner #'uniform-stretching`. This property can only be changed at the beginning of a score,

```
\score {
  <<
    \new Staff {
      \times 4/5 {
        c8 c8 c8 c8 c8
      }
      c8 c8 c8 c8
    }
    \new Staff {
      c8 c8 c8 c8
      \times 4/5 {
        c8 c8 c8 c8 c8
      }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner #'uniform-stretching = ##t
    }
  }
}
```



When `strict-note-spacing` is set, notes are spaced without regard for clefs, bar lines, and grace notes,

```
\override Score.SpacingSpanner #'strict-note-spacing = ##t
\new Staff { c8[ c \clef alto c \grace { c16[ c] } c8 c c]  c32[ c32] }
```



Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.5.4 Line length

The most basic settings influencing the spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems ends at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is. The normal default setting is false, but if the score has only one system the default value is true.

The option `ragged-last` is similar to `ragged-right`, but only affects the last line of the piece. No restrictions are put on that line. The result is similar to formatting text paragraphs. In a paragraph, the last line simply takes its natural horizontal length.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

Voir aussi

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.5.5 Proportional notation

LilyPond supports proportional notation, a type of horizontal spacing in which each note consumes an amount of horizontal space exactly equivalent to its rhythmic duration. This type of proportional spacing is comparable to horizontal spacing on top of graph paper. Some late 20th- and early 21st-century scores use proportional notation to clarify complex rhythmic relationships or to facilitate the placement of timelines or other graphics directly in the score.
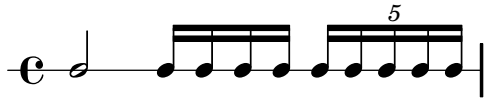
LilyPond supports five different settings for proportional notation, which may be used together or alone:

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove Separating_line_group_engraver`
- `\override PaperColumn #'used = ##t`

In the examples that follow, we explore these five different proportional notation settings and examine how these settings interact.

We start with the following one-measure example, which uses classical spacing with ragged-right turned on.

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
}
```



Notice that the half note which begins the measure takes up far less than half of the horizontal space of the measure. Likewise, the sixteenth notes and sixteenth-note quintuplets (or twentieth notes) which end the measure together take up far more than half the horizontal space of the measure.

In classical engraving, this spacing may be exactly what we want because we can borrow horizontal space from the half note and conserve horizontal space across the measure as a whole.

On the other hand, if we want to insert a measured timeline or other graphic above or below our score, we need proportional notation. We turn proportional notation on with the proportionalNotationDuration setting.

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
 \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
    }
  }
}
```



The half note at the beginning of the measure and the faster notes in the second half of the measure now occupy equal amounts of horizontal space. We could place a measured timeline or graphic above or below this example.

The `proportionalNotationDuration` setting is a context setting that lives in `Score`. Remember that context settings can appear in one of three locations within our input file – in a `\with` block, in a `\context` block, or directly in music entry preceded by the `\set` command. As with all context settings, users can pick which of the three different locations they would like to set `proportionalNotationDuration` in to.

The `proportionalNotationDuration` setting takes a single argument, which is the reference duration against that all music will be spaced. The LilyPond Scheme function `make-moment` takes two arguments – a numerator and denominator which together express some fraction of a whole note. The call `#(ly:make-moment 1 20)` therefore produces a reference duration of a twentieth note. Values such as `#(ly:make-moment 1 16)`, `#(ly:make-moment 1 8)`, and `#(ly:make-moment 3 97)` are all possible as well.

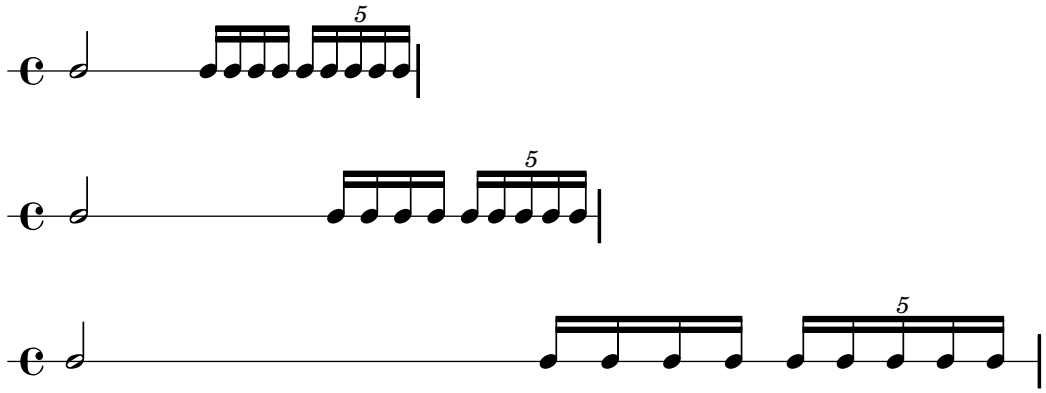
How do we select the right reference duration to pass to `proportionalNotationDuration`? Usually by a process of trial and error, beginning with a duration close to the fastest (or smallest) duration in the piece. Smaller reference durations space music loosely; larger reference durations space music tightly.

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 8)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 16)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 32)
    }
  }
```

```
}
```

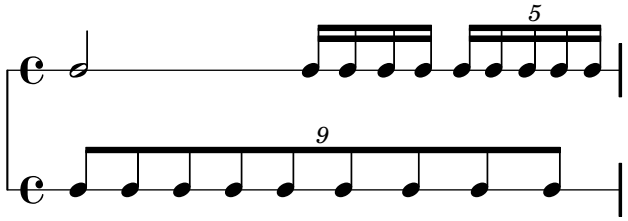


Note that too large a reference duration – such as the eighth note, above – spaces music too tightly and can cause note head collisions. Also that proportional notation in general takes up more horizontal space than classical spacing. Proportional spacing provides rhythmic clarity at the expense of horizontal space.

Next we examine how to optimally space overlapping tuplets.

We start by examining what happens to our original example, with classical spacing, when we add a second staff with a different type of tuplet.

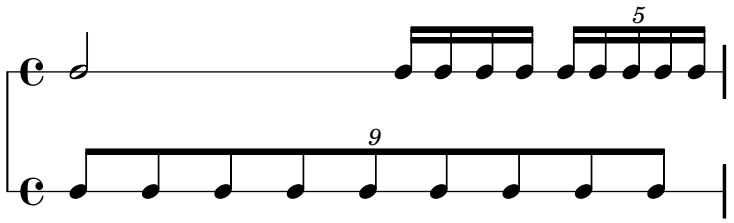
```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
}
```



The spacing is bad because the evenly spaced notes of the bottom staff do not stretch uniformly. Classical engravings include very few complex triplets and so classical engraving rules can generate this type of result. Setting `proportionalNotationDuration` fixes this.

```
\score {
  <<
    \new RhythmicStaff {
```

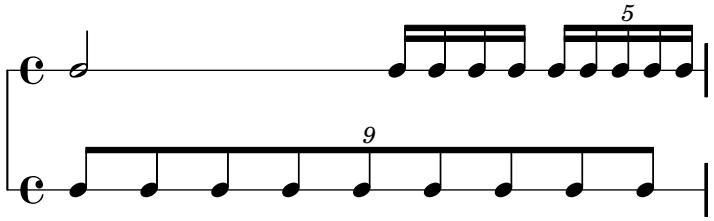
```
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
    }
  }
}
```



But if we look very carefully we can see that notes of the second half of the 9-tuplet space ever so slightly more widely than the notes of the first half of the 9-tuplet. To ensure uniform stretching, we turn on `uniform-stretching`, which is a property of `SpacingSpanner`.

```
\score {
  <<
    \new RhythmicStaff {
      c'2
      c'16 c'16 c'16 c'16
      \times 4/5 {
        c'16 c'16 c'16 c'16 c'16
      }
    }
    \new RhythmicStaff {
      \times 8/9 {
        c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8 c'8
      }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1 20)
      \override SpacingSpanner #'uniform-stretching = ##t
    }
  }
```

```
}
```



Our two-staff example now spaces exactly, our rhythmic relationships are visually clear, and we can include a measured timeline or graphic if we want.

Note that the LilyPond's proportional notation package expects that all proportional scores set the SpacingSpanner's 'uniform-stretching attribute to ##t. Setting proportionalNotation-Duration without also setting the SpacingSpanner's 'uniform-stretching attribute to ##t will, for example, cause Skips to consume an incorrect amount of horizontal space.

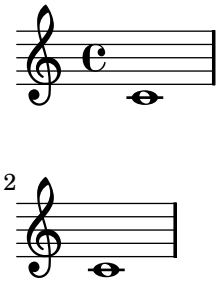
The SpacingSpanner is an abstract grob that lives in the Score context. As with our settings of proportionalNotationDuration, overrides to the SpacingSpanner can occur in any of three different places in our input file – in the Score \with block, in a Score \context block, or in note entry directly.

There is by default only one `SpacingSpanner` per `Score`. This means that, by default, `uniform-stretching` is either turned on for the entire score or turned off for the entire score. We can, however, override this behavior and turn on different spacing features at different places in the score. We do this with the command `\newSpacingSection`. See Section 4.5.2 [New spacing area], page 421, for more info.

Next we examine the effects of the `Separating_line_group_engraver` and see why proportional scores frequently remove this engraver. The following example shows that there is a small amount of « preferatory » space just before the first note in each system.

```
\paper {
  indent = #0
}

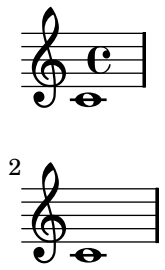
\new Staff {
  c'1
  \break
  c'1
}
```



The amount of this preferatory space is the same whether after a time signature, a key signature or a clef. `Separating_line_group_engraver` is responsible for this space. Removing `Separating_line_group_engraver` reduces this space to zero.

```
\paper {
  indent = #0
}

\new Staff \with {
  \remove Separating_line_group_engraver
} {
  c'1
  \break
  c'1
}
```



Nonmusical elements like time signatures, key signatures, clefs and accidentals are problematic in proportional notation. None of these elements has rhythmic duration. But all of these elements consume horizontal space. Different proportional scores approach these problems differently.

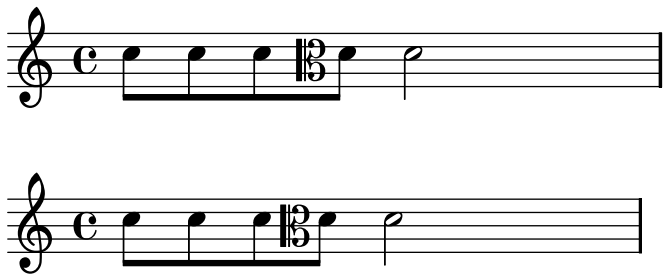
It may be possible to avoid spacing problems with key signatures simply by not having any. This is a valid option since most proportional scores are contemporary music. The same may be true of time signatures, especially for those scores that include a measured timeline or other graphic. But these scores are exceptional and most proportional scores include at least some time signatures. Clefs and accidentals are even more essential.

So what strategies exist for spacing nonmusical elements in a proportional context? One good option is the `strict-note-spacing` property of `SpacingSpanner`. Compare the two scores below:

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1 16)
  c''8
  c''8
  c''8
  \clef alto
  d'8
  d'2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1 16)
  \override Score.SpacingSpanner #'strict-note-spacing = ##t
  c''8
  c''8
  c''8
  \clef alto
  d'8
  d'2
```

```
}
```



Both scores are proportional, but the spacing in the first score is too loose because of the clef change. The spacing of the second score remains strict, however, because strict-note-spacing is turned on. Turning on strict-note-spacing causes the width of time signatures, key signatures, clefs and accidentals to play no part in the spacing algorithm.

In addition to the settings given here, there are other settings that frequently appear in proportional scores. These include:

- `\override SpacingSpanner #'strict-grace-spacing = ##t`
- `tupletFullLength = ##t`
- `\override Beam #'breakable = ##t`
- `\override Glissando #'breakable = ##t`
- `\override TextSpanner #'breakable = ##t`
- `\remove Forbid_line_break_engraver in the Voice context`

These settings space grace notes strictly, extend tuplet brackets to mark both rhythmic start- and stop-points, and allow spanning elements to break across systems and pages. See the respective parts of the manual for these related settings.

Voir aussi

Notation Reference: Section 4.5.2 [New spacing area], page 421.

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.6 Fitting music onto fewer pages

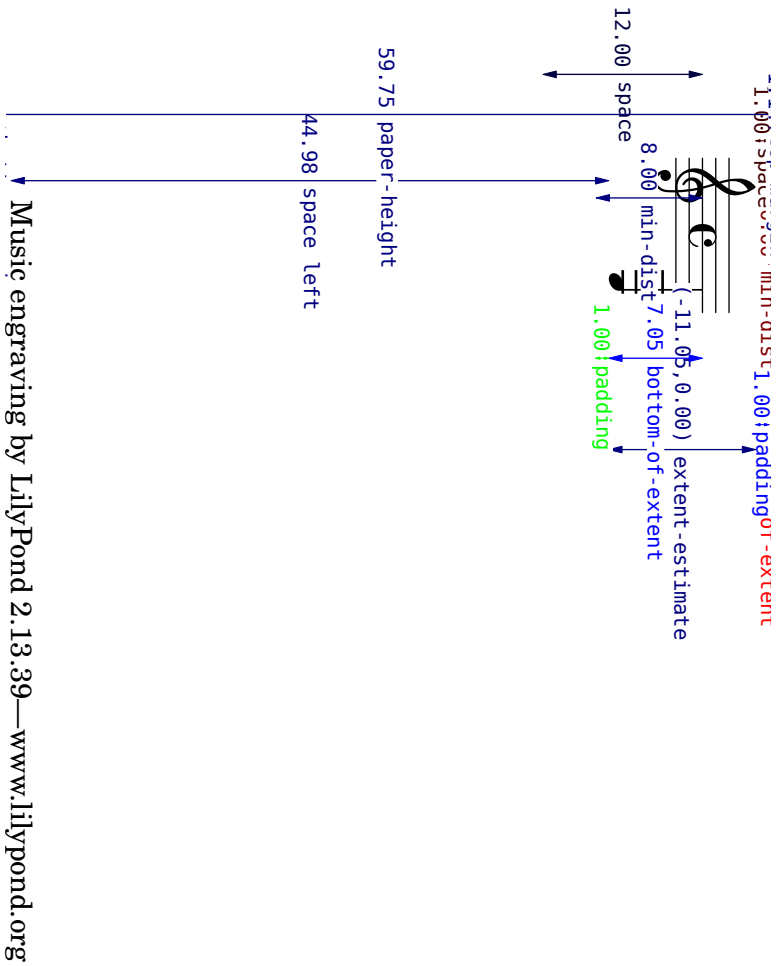
Sometimes you can end up with one or two staves on a second (or third, or fourth...) page. This is annoying, especially if you look at previous pages and it looks like there is plenty of room left on those.

When investigating layout issues, `annotate-spacing` is an invaluable tool. This command prints the values of various layout spacing variables; for more details see the following section, Section 4.6.1 [Displaying spacing], page 431.

4.6.1 Displaying spacing

To graphically display the dimensions of vertical layout variables that may be altered for page formatting, set `annotate-spacing` in the `\paper` block:

```
#(set-default-paper-size "a6" 'landscape)
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



All layout dimensions are displayed in staff-spaces, regardless of the units specified in the `\paper` or `\layout` block. In the above example, `paper-height` has a value of 59.75 `staff-spaces`, and the `staff-size` is 20 points (the default value). Note that:

1 point = (25.4/72.27) mm

1 staff-space = (`staff-size`)/4 pts
= (`staff-size`)/4 * (25.4/72.27) mm

In this case, one `staff-space` is approximately equal to 1.757mm. Thus the `paper-height` measurement of 59.75 `staff-spaces` is equivalent to 105 millimeters, the height of `a6` paper in landscape orientation. The pairs (*a*,*b*) are intervals, where *a* is the lower edge and *b* the upper edge of the interval.

Voir aussi

Notation Reference: Section 4.2.1 [Setting the staff size], page 398.

Snippets: Section “Spacing” dans *Morceaux choisis*.

4.6.2 Changing spacing

The output of `annotate-spacing` reveals vertical dimensions in great detail. For details about modifying margins and other layout variables, see Section 4.1.2 [Page formatting], page 391.

Other than margins, there are a few other options to save space:

- Force systems to move as close together as possible (to fit as many systems as possible onto a page) while being spaced so that there is no blank space at the bottom of the page.
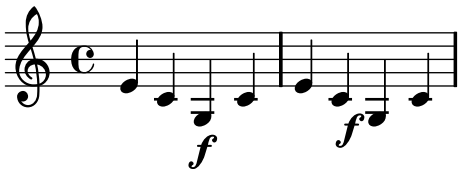
```
\paper {
  system-system-spacing = #'((padding . 0) (space . 0.1))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Force the number of systems. This can help in two ways. Just setting a value, even the same value as the number of systems being typeset by default, will sometimes cause more systems to be fitted onto each page, as an estimation step is then bypassed, giving a more accurate fit to each page. Also, forcing an actual reduction in the number of systems may save a further page. For example, if the default layout has 11 systems, the following assignment will force a layout with 10 systems.

```
\paper {
  system-count = #10
}
```
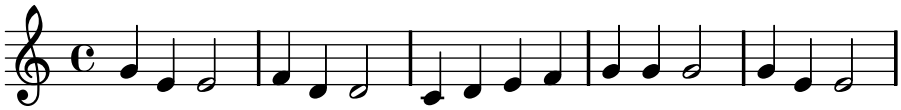
- Avoid (or reduce) objects that increase the vertical size of a system. For example, volta repeats (or alternate repeats) require extra space. If these repeats are spread over two systems, they will take up more space than one system with the volta repeats and another system without. For example, dynamics that 'stick out' of a system can be moved closer to the staff:

```
e4 c g\f c
e4 c g-\tweak #'X-offset #-2.7 -\tweak #'Y-offset #2.5 \f c
```
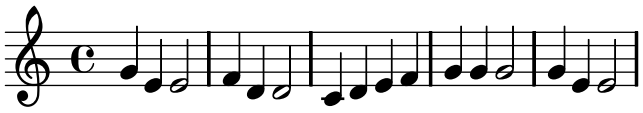


- Alter the horizontal spacing via `SpacingSpanner`. For more details, see Section 4.5.3 [Changing horizontal spacing], page 421. The following example illustrates the default spacing:

```
\score {
  \relative c'' {
    g4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```



The next example modifies `common-shortest-duration` from a value of `1/4` to `1/2`. The quarter note is the most common and shortest duration in this example, so by making this duration longer, a 'squeezing' effect occurs:

```
\score {
  \relative c'' {
    g4 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)
    }
  }
}
```



The `common-shortest-duration` property cannot be modified dynamically, so it must always be placed in a `\context` block so that it applies to the whole score.

Voir aussi

Notation Reference: Section 4.1.2 [Page formatting], page 391, Section 4.5.3 [Changing horizontal spacing], page 421.

Snippets: Section "Spacing" dans *Morceaux choisis*.

5 Changing defaults

The purpose of LilyPond's design is to provide the finest quality output by default. Nevertheless, it may happen that you need to change this default layout. The layout is controlled through a large number of 'knobs and switches' collectively called 'properties'. A tutorial introduction to accessing and modifying these properties can be found in the Learning Manual, see Section "Tweaking output" dans *Manuel d'initiation*. This should be read first. This chapter covers similar ground, but in a style more appropriate to a reference manual.

The definitive description of the controls available for tuning can be found in a separate document: Section "Top" dans *Référence des propriétés internes*. That manual lists all the variables, functions and options available in LilyPond. It is written as a HTML document, which is available on-line, and is also included with the LilyPond documentation package.

Internally, LilyPond uses Scheme (a LISP dialect) to provide infrastructure. Overriding layout decisions in effect accesses the program internals, which requires Scheme input. Scheme elements are introduced in a `.ly` file with the hash mark `#`.[1]

5.1 Interpretation contexts

This section describes what contexts are, and how to modify them.

Voir aussi

Learning Manual: Section "Contexts and engravers" dans *Manuel d'initiation*.

Installed files: '`ly/engraver-init.ly`', '`ly/performer-init.ly`'.

Snippets: Section "Contexts and engravers" dans *Morceaux choisis*.

Internals Reference: Section "Contexts" dans *Référence des propriétés internes*, Section "Engravers and Performers" dans *Référence des propriétés internes*.

5.1.1 Contexts explained

Contexts are arranged hierarchically:

Score - the master of all contexts

This is the top level notation context. No other context can contain a Score context. By default the Score context handles the administration of time signatures and makes sure that items such as clefs, time signatures, and key-signatures are aligned across staves.

A Score context is instantiated implicitly when a `\score {...}` or `\layout {...}` block is processed.

Top-level contexts - staff containers

StaffGroup

Groups staves while adding a bracket on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically. `StaffGroup` only consists of a collection of staves, with a bracket in front and spanning bar lines.

ChoirStaff

Identical to `StaffGroup` except that the bar lines of the contained staves are not connected vertically.

GrandStaff

[1] Section "Scheme tutorial" dans *Extension de LilyPond*, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.

A group of staves, with a brace on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically.

PianoStaff

Just like `GrandStaff`, but with support for instrument names to the left of each system.

Intermediate-level contexts - staves

Staff

Handles clefs, bar lines, keys, accidentals. It can contain `Voice` contexts.

RhythmicStaff

Like `Staff` but for printing rhythms. Pitches are ignored; the notes are printed on one line.

TabStaff

Context for generating tablature. By default lays the music expression out as a guitar tablature, printed on six lines.

DrumStaff

Handles typesetting for percussion. Can contain `DrumVoice`

VaticanaStaff

Same as `Staff`, except that it is designed for typesetting a piece in gregorian style.

MensuralStaff

Same as `Staff`, except that it is designed for typesetting a piece in mensural style.

Bottom-level contexts - voices

Voice-level contexts initialise certain properties and start appropriate engravers. Being bottom-level contexts, they cannot contain other contexts.

Voice

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and sub-scripts, slurs, ties, and rests. You have to instantiate this explicitly if you require multiple voices on the same staff.

VaticanaVoice

Same as `Voice`, except that it is designed for typesetting a piece in gregorian style.

MensuralVoice

Same as `Voice`, with modifications for typesetting a piece in mensural style.

Lyrics

Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

DrumVoice

The voice context used in a percussion staff.

FiguredBass

The context in which `BassFigure` objects are created from input entered in `\figuremode` mode.

TabVoice

The voice context used within a `TabStaff` context. Usually left to be created implicitly.

CueVoice

A voice context used to render notes of a reduced size, intended primarily for adding cue notes to a staff, see ⟨undefined⟩ [Formatting cue notes], page ⟨undefined⟩. Usually left to be created implicitly.

ChordNames

Typesets chord names.

5.1.2 Creating contexts

For scores with only one voice and one staff, contexts are created automatically. For more complex scores, it is necessary to create them by hand. There are three commands that do this.
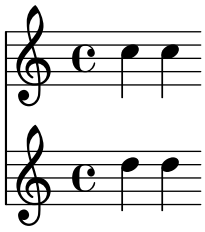
- The easiest command is `\new`, and it also the quickest to type. It is prepended to a music expression, for example

  `\new` *`type music expression`*

  where *type* is a context name (like `Staff` or `Voice`). This command creates a new context, and starts interpreting the *music expression* with that.

  A practical application of `\new` is a score with many staves. Each part that should be on its own staff, is preceded with `\new Staff`.

  ```
  <<
    \new Staff { c4 c }
    \new Staff { d4 d }
  >>
  ```

  The `\new` command may also give a name to the context,

  `\new` *`type = id music`*

  However, this user specified name is only used if there is no other context already earlier with the same name.

- Like `\new`, the `\context` command also directs a music expression to a context object, but gives the context an explicit name. The syntax is

  `\context` *`type = id music`*

  This form will search for an existing context of type *type* called *id*. If that context does not exist yet, a new context with the specified name is created. This is useful if the context is referred to later on. For example, when setting lyrics the melody is in a named context

  `\context Voice = "tenor"` *`music`*

  so the texts can be properly aligned to its notes,
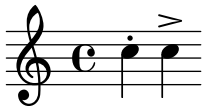
  `\new Lyrics \lyricsto "tenor"` *`lyrics`*

  Another possible use of named contexts is funneling two different music expressions into one context. In the following example, articulations and notes are entered separately,

  ```
  music = { c4 c4 }
  arts = { s4-. s4-> }
  ```

  They are combined by sending both to the same `Voice` context,

  ```
  <<
    \new Staff \context Voice = "A" \music
    \context Voice = "A" \arts
  >>
  ```

With this mechanism, it is possible to define an Urtext (original edition), with the option to put several distinct articulations on the same notes.

- The third command for creating contexts is

  `\context` *`type music`*

  This is similar to `\context` with `= id`, but matches any context of type *type*, regardless of its given name.

  This variant is used with music expressions that can be interpreted at several levels. For example, the `\applyOutput` command (see Section "Running a function on all layout objects" dans *Extension de LilyPond*). Without an explicit `\context`, it is usually applied to `Voice`

  ```
  \applyOutput #'context #function   % apply to Voice
  ```

  To have it interpreted at the `Score` or `Staff` level use these forms

  ```
  \applyOutput #'Score #function
  \applyOutput #'Staff #function
  ```

5.1.3 Keeping contexts alive

Contexts are usually terminated at the first musical moment in which they have nothing to do. So `Voice` contexts die as soon as they contain no events; `Staff` contexts die as soon as all the `Voice` contexts within them contain no events; etc. This can cause difficulties if earlier contexts which have died have to be referenced, for example, when changing staves with `\change` commands, associating lyrics with a voice with `\lyricsto` commands, or when adding further musical events to an earlier context.

There is an exception to this general rule: just one of the `Voice` contexts in a `Staff` context or in a `<<...>>` construct will always persist to the end of the enclosing `Staff` context or `<<...>>` construct, even though there may be periods when it has nothing to do. The context to persist in this way will be the first one encountered in the first enclosed `{...}` construct, ignoring any in enclosed `<<...>>` constructs.

Any context can be kept alive by ensuring it has something to do at every musical moment. `Staff` contexts are kept alive by ensuring one of their voices is kept alive. One way of doing this is to add spacer rests to a voice in parallel with the real music. These need to be added to every `Voice` context which needs to be kept alive. If several voices are to be used sporadically it is safest to keep them all alive rather than attempting to rely on the exceptions mentioned above.

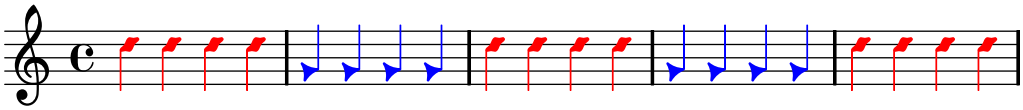
In the following example, both voice A and voice B are kept alive in this way for the duration of the piece:

```
musicA = \relative c'' { d4 d d d }
musicB = \relative c'' { g4 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 }
}

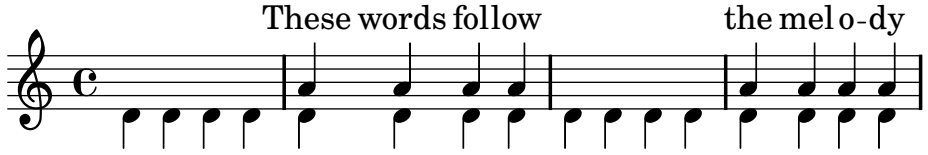
\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



The following example shows how a sporadic melody line with lyrics might be written using this approach. In a real situation the melody and accompaniment would consist of several different sections, of course.

```
melody = \relative c'' { a4 a a a }
accompaniment = \relative c' { d4 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 }
```

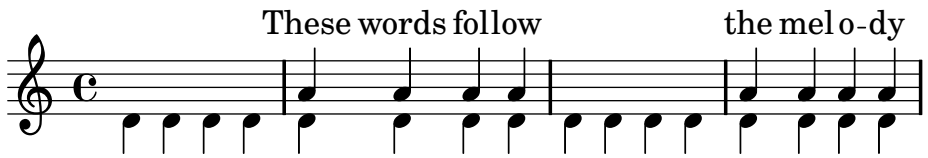
```
  >>
}
```



An alternative way, which may be better in many circumstances, is to keep the melody line alive by simply including spacer notes to line it up correctly with the accompaniment:

```
melody = \relative c'' {
  s1  % skip a bar
  a4 a a a
  s1  % skip a bar
  a4 a a a
}
accompaniment = \relative c' {
  d4 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 Modifying context plug-ins

Notation contexts (like `Score` and `Staff`) not only store properties, they also contain plug-ins called ‘engravers’ that create notation elements. For example, the `Voice` context contains a `Note_head_engraver` and the `Staff` context contains a `Key_signature_engraver`.
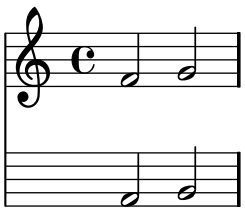
For a full a description of each plug-in, see Internals Reference ↦ Translation ↦ Engravers. Every context described in Internals Reference ↦ Translation ↦ Context. lists the engravers used for that context.

It can be useful to shuffle around these plug-ins. This is done by starting a new context with `\new` or `\context`, and modifying it,

```
\new context \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ..music..
}
```

where the ... should be the name of an engraver. Here is a simple example which removes `Time_signature_engraver` and `Clef_engraver` from a `Staff` context,

```
<<
  \new Staff {
    f2 g
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } {
    f2 g2
  }
>>
```

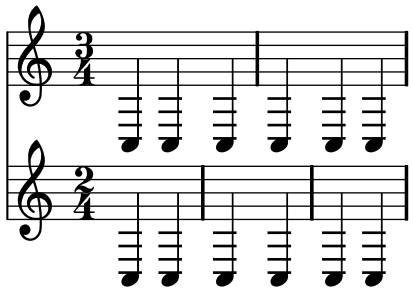


In the second staff there are no time signature or clef symbols. This is a rather crude method of making objects disappear since it will affect the entire staff. This method also influences the spacing, which may or may not be desirable. More sophisticated methods of blanking objects are shown in Section “Visibility and color of objects” dans *Manuel d’initiation*.

The next example shows a practical application. Bar lines and time signatures are normally synchronized across the score. This is done by the `Timing_translator` and `Default_bar_line_engraver`. This plug-in keeps an administration of time signature, location within the measure, etc. By moving these engraver from `Score` to `Staff` context, we can have a score where each staff has its own time signature.

```
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    } {
```

```
        \time 3/4
        c4 c c c c c
    }
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  } {
      \time 2/4
      c4 c c c c c
  }
>>
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    }
  }
}
```



Problèmes connus et avertissements

Usually the order in which the engravers are specified does not matter, but in a few special cases the order is important, for example where one engraver writes a property and another reads it, or where one engraver creates a grob and another must process it. The order in which the engravers are specified is the order in which they are called to carry out their processing.

The following orderings are important: the `Bar_engraver` must normally be first, and the `New_fingering_engraver` must come before the `Script_column_engraver`. There may be others with ordering dependencies.

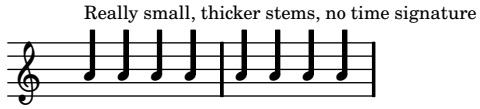
5.1.5 Changing context default settings

The context settings which are to be used by default in `Score`, `Staff` and `Voice` contexts may be specified in a `\layout` block, as illustrated in the following example. The `\layout` block should be placed within the `\score` block to which it is to apply, but outside any music.

Note that the `\set` command itself and the context must be omitted when the context default values are specified in this way:

```
\score {
  \relative c'' {
    a4^"Really small, thicker stems, no time signature" a a a
    a a a a
  }
  \layout {
```

```
    \context {
      \Staff
      fontSize = #-4
      \override Stem #'thickness = #4.0
      \remove "Time_signature_engraver"
    }
  }
}
```



In this example, the `\Staff` command specifies that the subsequent specifications are to be applied to all staves within this score block.

Modifications can be made to the `Score` context or all `Voice` contexts in a similar way.

Problèmes connus et avertissements

It is not possible to collect context changes in a variable and apply them to a `\context` definition by referring to that variable.

The `\Staff \RemoveEmptyStaves` will overwrite your current `\Staff` settings. If you wish to change the defaults for a staff which uses `\Staff \RemoveEmptyStaves`, you must do so after calling `\Staff \RemoveEmptyStaves`, ie

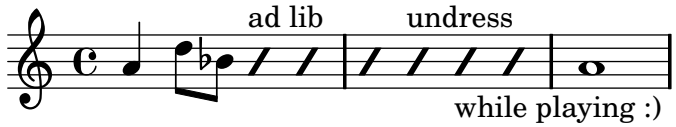
```
\layout {
  \context {
    \Staff \RemoveEmptyStaves

    \override Stem #'thickness = #4.0
  }
}
```

5.1.6 Defining new contexts

Specific contexts, like `Staff` and `Voice`, are made of simple building blocks. It is possible to create new types of contexts with different combinations of engraver plug-ins.

The next example shows how to build a different type of `Voice` context from scratch. It will be similar to `Voice`, but only prints centered slash note heads. It can be used to indicate improvisation in jazz pieces,



These settings are defined within a `\context` block inside a `\layout` block,

```
\layout {
  \context {
    ...
  }
}
```

In the following discussion, the example input shown should go in place of the ... in the previous fragment.

First it is necessary to define a name for the new context:

```
\name ImproVoice
```

Since it is similar to the `Voice`, we want commands that work on (existing) `Voice`s to remain working. This is achieved by giving the new context an alias `Voice`,

```
\alias Voice
```

The context will print notes and instructive texts, so we need to add the engravers which provide this functionality,

```
\consists Note_heads_engraver
\consists Text_engraver
```

but we only need this on the center line,

```
\consists Pitch_squash_engraver
squashedPosition = #0
```

The Section "Pitch_squash_engraver" dans *Référence des propriétés internes* modifies note heads (created by Section "Note_heads_engraver" dans *Référence des propriétés internes*) and sets their vertical position to the value of `squashedPosition`, in this case `0`, the center line.

The notes look like a slash, and have no stem,

```
\override NoteHead #'style = #'slash
\override Stem #'transparent = ##t
```

All these plug-ins have to cooperate, and this is achieved with a special plug-in, which must be marked with the keyword `\type`. This should always be `Engraver_group`.

```
\type "Engraver_group"
```

Put together, we get

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists Pitch_squash_engraver
  squashedPosition = #0
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
  \alias Voice
}
```

Contexts form hierarchies. We want to hang the `ImproVoice` under `Staff`, just like normal `Voice`s. Therefore, we modify the `Staff` definition with the `\accepts` command,

```
\context {
  \Staff
  \accepts ImproVoice
}
```

The opposite of `\accepts` is `\denies`, which is sometimes needed when reusing existing context definitions.

Putting both into a `\layout` block, like

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
```
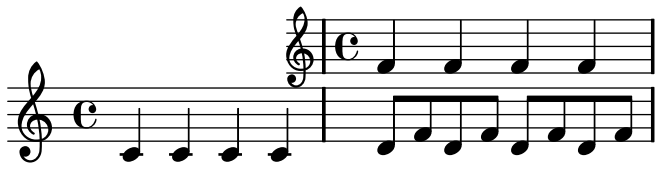
```
    \accepts "ImproVoice"
  }
}
```

Then the output at the start of this subsection can be entered as

```
\relative c'' {
  a4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"undress"
    c c_"while playing :)"
  }
  a1
}
```
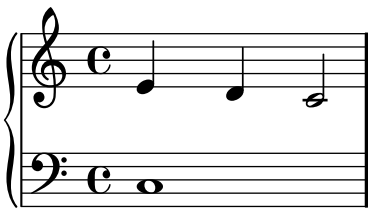
5.1.7 Aligning contexts

New contexts may be aligned above or below existing contexts. This could be useful in setting up a vocal staff (Section "Vocal ensembles" dans *Manuel d'initiation*) and in ossia,



Contexts like `PianoStaff` can contain other contexts nested within them. Contexts which are acceptable for nesting are defined by the « accepts » list of a context. Contexts which are not in this list are placed below the outer context in the printed score. For example, the `PianoStaff` context is defined by default to accept `Staff` and `FiguredBass` contexts within it, but not (for example) a `Lyrics` context. So in the following structure the lyrics are placed below the piano staff rather than between the two staves:
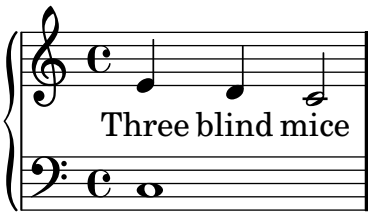
```
\new PianoStaff
<<
  \new Staff { e4 d c2 }
  \addlyrics { Three blind mice }
  \new Staff {
    \clef "bass"
    { c,1 }
  }
>>
```



Three blind mice

The « accepts » list of a context can be modified to include additional nested contexts, so if we wanted the lyrics to appear between the two staves we could use:

```
\new PianoStaff \with { \accepts Lyrics }
<<
  \new Staff { e4 d c2 }
  \addlyrics { Three blind mice }
  \new Staff {
    \clef "bass"
    { c,1 }
  }
>>
```
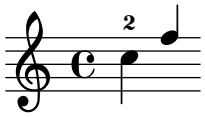


The opposite of `\accepts` is `\denies`; this removes a context from the « accepts » list.

5.2 Explaining the Internals Reference

5.2.1 Navigating the program reference

Suppose we want to move the fingering indication in the fragment below:

```
c-2
\stemUp
f
```



If you visit the documentation on fingering instructions (in ⟨undefined⟩ [Fingering instructions], page ⟨undefined⟩), you will notice:

See also

Internals Reference: Section “Fingering” dans *Référence des propriétés internes*.

The programmer’s reference is available as an HTML document. It is highly recommended that you read it in HTML form, either online or by downloading the HTML documentation. This section will be much more difficult to understand if you are using the PDF manual.

Follow the link to Section “Fingering” dans *Référence des propriétés internes*. At the top of the page, you will see

Fingering objects are created by: Section “Fingering_engraver” dans *Référence des propriétés internes* and Section “New_fingering_engraver” dans *Référence des propriétés internes*.

By following related links inside the program reference, we can follow the flow of information within the program:

- 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*

This path goes against the flow of information in the program: it starts from the output, and ends at the input event. You could also start at an input event, and read with the flow of information, eventually ending up at the output object(s).

The program reference can also be browsed like a normal document. It contains chapters on `Music definitions` on Section "Translation" dans *Référence des propriétés internes*, and the Section "Backend" dans *Référence des propriétés internes*. Every chapter lists all the definitions used and all properties that may be tuned.

5.2.2 Layout interfaces

The HTML page that we found in the previous section describes the layout object called Section "Fingering" dans *Référence des propriétés internes*. Such an object is a symbol within the score. It has properties that store numbers (like thicknesses and directions), but also pointers to related objects. A layout object is also called a *Grob*, which is short for Graphical Object. For more details about Grobs, see Section "grob-interface" dans *Référence des propriétés internes*.

The page for `Fingering` lists the definitions for the `Fingering` object. For example, the page says

`padding` (dimension, in staff space):

`0.5`

which means that the number will be kept at a distance of at least 0.5 of the note head.

Each layout object may have several functions as a notational or typographical element. For example, the Fingering object has the following aspects

- Its size is independent of the horizontal spacing, unlike slurs or beams.
- It is a piece of text. Granted, it is usually a very short text.
- That piece of text is typeset with a font, unlike slurs or beams.
- Horizontally, the center of the symbol should be aligned to the center of the note head.
- Vertically, the symbol is placed next to the note and the staff.
- The vertical position is also coordinated with other superscript and subscript symbols.

Each of these aspects is captured in so-called *interface*s, which are listed on the Section "Fingering" dans *Référence des propriétés internes* page at the bottom

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*.

Clicking any of the links will take you to the page of the respective object interface. Each interface has a number of properties. Some of them are not user-serviceable ('Internal properties'), but others can be modified.

We have been talking of *the* `Fingering` object, but actually it does not amount to much. The initialization file (see Section "Other sources of information" dans *Manuel d'initiation*) '`scm/define-grobs.scm`' shows the soul of the 'object',

```
(Fingering
  . ((padding . 0.5)
```
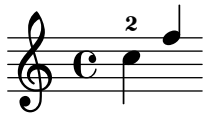
```
(avoid-slur . around)
(slur-padding . 0.2)
(staff-padding . 0.5)
(self-alignment-X . 0)
(self-alignment-Y . 0)
(script-priority . 100)
(stencil . ,ly:text-interface::print)
(direction . ,ly:script-interface::calc-direction)
(font-encoding . fetaText)
(font-size . -5)  ; don't overlap when next to heads.
(meta . ((class . Item)
(interfaces . (finger-interface
               font-interface
               text-script-interface
               text-interface
               side-position-interface
               self-alignment-interface
               item-interface))))))
```

As you can see, the `Fingering` object is nothing more than a bunch of variable settings, and the webpage in the Internals Reference is directly generated from this definition.

5.2.3 Determining the grob property

Recall that we wanted to change the position of the **2** in

```
c-2
\stemUp
f
```



Since the **2** is vertically positioned next to its note, we have to meddle with the interface associated with this positioning. This is done using `side-position-interface`. The page for this interface says

`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?)

Below this description, the variable `padding` is described as

`padding` (dimension, in staff space)
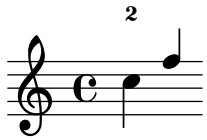
Add this much extra space between objects that are next to each other.

By increasing the value of `padding`, we can move the fingering away from the note head. The following command inserts 3 staff spaces of white between the note and the fingering:

```
\once \override Voice.Fingering #'padding = #3
```

Inserting this command before the Fingering object is created, i.e., before `c2`, yields the following result:

```
\once \override Voice.Fingering #'padding = #3
c-2
\stemUp
f
```



In this case, the context for this tweak is `Voice`. This fact can also be deduced from the program reference, for the page for the Section "Fingering_engraver" dans *Référence des propriétés internes* plug-in says

Fingering_engraver is part of contexts: ... Section "Voice" dans *Référence des propriétés internes*

5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (incl. one-word names)
- scheme functions: ly:plus-scheme-style
- music events, music classes and music properties: as-scheme-functions
- Grob interfaces: scheme-style
- backend properties: scheme-style (but X and Y!)
- contexts (and MusicExpressions and grobs): Capitalized or CamelCase
- context properties: lowercaseFollowedByCamelCase
- engravers: Capitalized_followed_by_lowercase_and_with_underscores

Questions to be answered:

- Which of these are conventions and which are rules?
- Which are rules of the underlying language, and which are LP-specific?

5.3 Modifying properties

5.3.1 Overview of modifying properties

Each context is responsible for creating certain types of graphical objects. The settings used for printing these objects are also stored by context. By changing these settings, the appearance of objects can be altered.

There are two different kinds of properties stored in contexts: context properties and grob properties. Context properties are properties that apply to the context as a whole and control how the context itself is displayed. In contrast, grob properties apply to specific grob types that will be displayed in the context.

The `\set` and `\unset` commands are used to change values for context properties. The `\override` and `\revert` commands are used to change values for grob properties.

Voir aussi

Internals: Section "Backend" dans *Référence des propriétés internes*, Section "All layout objects" dans *Référence des propriétés internes*, Section "OverrideProperty" dans *Référence des propriétés internes*, Section "RevertProperty" dans *Référence des propriétés internes*, Section "PropertySet" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

The back-end is not very strict in type-checking object properties. Cyclic references in Scheme values for properties can cause hangs or crashes, or both.

5.3.2 The \set command

Each context has a set of *properties*, variables contained in that context. Context properties are changed with the `\set` command, which has the following syntax:

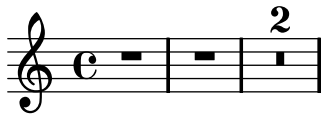
`\set` *`context.property = #value`*

value is a Scheme object, which is why it must be preceded by the `#` character.

Contexts properties are usually named in `studlyCaps`. They mostly control the translation from music to notation, e.g. `localKeySignature` (for determining whether to print accidentals), or `measurePosition` (for determining when to print a bar line). Context properties can change value over time while interpreting a piece of music; `measurePosition` is an obvious example of this. Context properties are modified with `\set`.
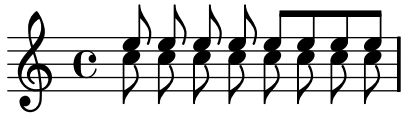
For example, multimeasure rests will be combined into a single bar if the context property `skipBars` is set to `#t`:

```
R1*2
\set Score.skipBars = ##t
R1*2
```



If the *context* argument is left out, then the property will be set in the current bottom context (typically `ChordNames`, `Voice`, `TabVoice`, or `Lyrics`).
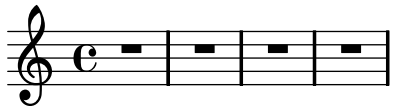
```
\set Score.autoBeaming = ##f
<<
  {
    e8 e e e
    \set autoBeaming = ##t
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```



The change is applied 'on-the-fly', during the music, so that the setting only affects the second group of eighth notes.

Note that the bottom-most context does not always contain the property that you wish to change – for example, attempting to set the `skipBars` property of the default bottom context, in this case `Voice`, will have no effect, because skipBars is a property of the `Score` context.

```
R1*2
\set skipBars = ##t
R1*2
```



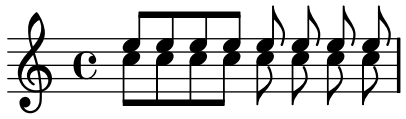
Contexts are hierarchical, so if an enclosing context was specified, for example `Staff`, then the change would also apply to all `Voice`s in the current staff.

The `\unset` command:

`\unset` *`context.property`*

is used to remove the definition of *property* from *context*. This command removes the definition only if it is set in *context*. Properties that have been set in enclosing contexts will not be altered by an unset in an enclosed context:

```
\set Score.autoBeaming = ##t
<<
  {
    \unset autoBeaming
    e8 e e e
    \unset Score.autoBeaming
    e8 e e e
  } \\ {
    c8 c c c c8 c c c
  }
>>
```
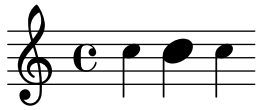


Like `\set`, the *context* argument does not have to be specified for a bottom context, so the two statements

```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

are equivalent if the current bottom context is `Voice`.

Preceding a `\set` command by `\once` makes the setting apply to only a single time-step:

```
c4
\once \set fontSize = #4.7
c4
c4
```



A full description of all available context properties is in the internals reference, see Translation ↦ Tunable context properties.

Voir aussi

Internals Reference:

Section “Tunable context properties” dans *Référence des propriétés internes*.

5.3.3 The \override command

There is a special type of context property: the grob description. Grob descriptions are named in StudlyCaps (starting with capital letters). They contain the 'default settings' for a particular kind of grob as an association list. See 'scm/define-grobs.scm' to see the settings for each grob description. Grob descriptions are modified with \override.

\override is actually a shorthand;

```
\override context.GrobName #'property = #value
```
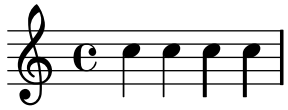
is more or less equivalent to

```
\set context.GrobName  =
  #(cons (cons 'property value)
         <previous value of context.GrobName>)
```

The value of context.GrobName (the alist) is used to initialize the properties of individual grobs. Grobs have properties, named in Scheme style, with dashed-words. The values of grob properties change during the formatting process: formatting basically amounts to computing properties using callback functions.
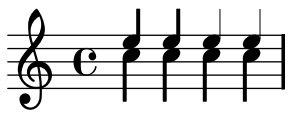
For example, we can increase the thickness of a note stem by overriding the thickness property of the Stem object:

```
c4 c
\override Voice.Stem #'thickness = #3.0
c4 c
```
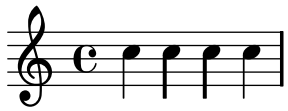


If no context is specified in an \override, the bottom context is used:

```
{ \override Staff.Stem #'thickness = #3.0
  <<
    {
      e4 e
      \override Stem #'thickness = #0.5
      e4 e
    } \\ {
      c4 c c c
    }
  >>
}
```
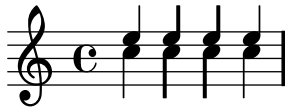


The effects of \override can be undone by \revert:

```
c4
\override Voice.Stem #'thickness = #3.0
c4 c
\revert Voice.Stem #'thickness
c4
```
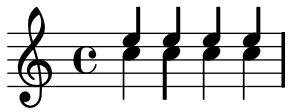


The effects of `\override` and `\revert` apply to all grobs in the affected context from the current time forward:

```
{
  <<
    {
      e4
      \override Staff.Stem #'thickness = #3.0
      e4 e e
    } \\ {
      c4 c c
      \revert Staff.Stem #'thickness
      c4
    }
  >>
}
```



`\once` can be used with `\override` to affect only the current time step:

```
{
  <<
    {
      \override Stem #'thickness = #3.0
      e4 e e e
    } \\ {
      c4
      \once \override Stem #'thickness = #3.0
      c4 c c
    }
  >>
}
```



Voir aussi

Internals Reference: Section “Backend” dans *Référence des propriétés internes*

5.3.4 The `\tweak` command

Changing grob properties with `\override` causes the changes to apply to all of the given grobs in the context at the moment the change applies. Sometimes, however, it is desirable to have changes apply to just one grob, rather than to all grobs in the affected context. This is accomplished with the `\tweak` command, which has the following syntax:

```
\tweak #'grob-property #value
```

The `\tweak` command applies to the object that immediately follows `value` in the music stream.
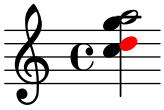
For an introduction to the syntax and uses of the tweak command see Section "Tweaking methods" dans *Manuel d'initiation*.

When several similar items are placed at the same musical moment, the `\override` command cannot be used to modify just one of them – this is where the `\tweak` command must be used. Items which may appear more than once at the same musical moment include the following:

- note heads of notes inside a chord
- articulation signs on a single note
- ties between notes in a chord
- tuplet brackets starting at the same time
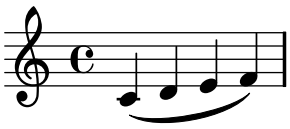
In this example, the color of one note head and the type of another note head are modified within a single chord:

```
< c
  \tweak #'color #red
  d
  g
  \tweak #'duration-log #1
  a
> 4
```
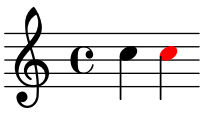


`\tweak` can be used to modify slurs:

```
c-\tweak #'thickness #5 ( d e f)
```



For the `\tweak` command to work, it must remain immediately adjacent to the object to which it is to apply after the input file has been converted to a music stream. At times, LilyPond may insert additional items into the music stream during the parsing process. For example, when a note that is not explicitly part of a chord will be placed in a chord by LilyPond, so notes to be modified with `\tweak` must be placed inside a chord construct:

```
\tweak #'color #red c4
<\tweak #'color #red c>4
```
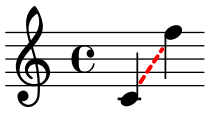


The `\tweak` command cannot be used to modify any item that does not appear explicitly in the input file. In particular it cannot be used to modify stems, beams or accidentals directly, since these are generated later by note heads, rather than by music elements in the input stream. Nor can `\tweak` be used to modify clefs or time signatures, since these become separated from

any preceding `\tweak` command in the input stream by the automatic insertion of extra elements required to specify the context.

Several `\tweak` commands may be placed before a notational element – all affect it:

```
c
-\tweak #'style #'dashed-line
-\tweak #'dash-fraction #0.2
-\tweak #'thickness #3
-\tweak #'color #red
 \glissando
f'
```



The music stream which is generated from a section of an input file, including any automatically inserted elements, may be examined, see Section "Displaying music expressions" dans *Extension de LilyPond*. This may be helpful in determining what may be modified by a `\tweak` command, or in determining how to adjust the input to make a `\tweak` apply.

Voir aussi

Learning Manual: Section "Tweaking methods" dans *Manuel d'initiation*.

Extending: Section "Displaying music expressions" dans *Extension de LilyPond*.

Problèmes connus et avertissements

The `\tweak` command cannot be used inside a variable.

The `\tweak` commands cannot be used in `\lyricmode`.

The `\tweak` command will apply to only the first of several generated ties in a chord.

5.3.5 `\set` vs. `\override`

TODO – This section is probably unnecessary now.

5.3.6 Modifying alists

Some user-configurable properties are internally represented as *alists* (association lists), which store pairs of *keys* and *values*. The structure of an alist is:

```
'((key1 . value1)
  (key2 . value2)
  (key3 . value3)
  ...)
```

If an alist is a grob property or `\paper` variable, its keys can be modified individually without affecting other keys.

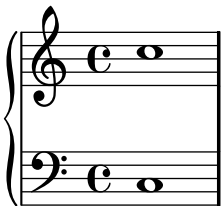
For example, to reduce the space between adjacent staves in a staff-group, use the `staff-staff-spacing` property of the `StaffGrouper` grob. The property is an alist with four keys: `padding`, `space`, `minimum-distance`, and `stretchability`. Three of the four keys have initialized default values; these are listed in the « Backend » section of the Internals Reference (see Section "StaffGrouper" dans *Référence des propriétés internes*):

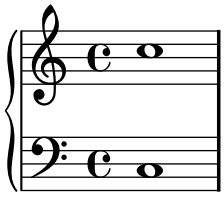
```
'((space . 9) (minimum-distance . 7) (padding . 1))
```

One way to bring the staves closer together is by reducing the value of the `space` key (`9`) to match the value of `minimum-distance` (`7`). To modify a single key individually, use a *nested declaration*:

```
% 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 #'space = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```

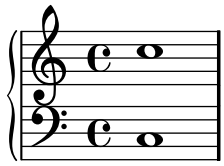




Using a nested declaration will update the specified key (such as `space` in the above example) without altering any other keys already set for the same property.

Now suppose we want the staves to be as close as possible without overlapping. The simplest way to do this is to set all four alist keys to zero. However, it is not necessary to enter four nested declarations, one for each key. Instead, the property can be completely re-defined with one declaration, as an alist:

```
\new PianoStaff \with {
  \override StaffGrouper #'staff-staff-spacing =
    #'((padding . 0)
       (space . 0)
       (minimum-distance . 0)
       (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



Note that any keys not explicitly listed in the alist definition will be reset to their *default-when-unset* values. In the case of `staff-staff-spacing`, any unset key-values would be reset to zero (except `stretchability`, which takes the value of `space` when unset). Thus the following two declarations are equivalent:

```
\override StaffGrouper #'staff-staff-spacing =
  #'((space . 7))

\override StaffGrouper #'staff-staff-spacing =
  #'((padding . 0)
     (space . 7)
     (minimum-distance . 0)
     (stretchability . 7))
```

One (possibly unintended) consequence of this is the removal of any *initialized* default values that are set in an initialization file and loaded each time an input file is compiled. In the above example, the initialized default values for `padding` and `minimum-distance` (defined in '`scm/define-grobs.scm`') are reset to their default-when-unset values (zero for both keys). Defining a property or variable as an alist (of any size) will always reset all unset key-values to their default-when-unset values. Unless this is the intended result, it is safer to update key-values individually with a nested declaration.

Note : Nested declarations will not work for context property alists (such as `beamExceptions`, `keySignature`, `timeSignatureSettings`, etc.). These properties can only be modified by completely re-defining them as alists.

5.4 Useful concepts and properties

5.4.1 Input modes

The way in which the notation contained within an input file is interpreted is determined by the current input mode.

Chord mode

This is activated with the `\chordmode` command, and causes input to be interpreted with the syntax of chord notation, see Section 2.7 [Chord notation], page 307. Chords are rendered as notes on a staff.

Chord mode is also activated with the `\chords` command. This also creates a new `ChordNames` context and causes the following input to be interpreted with the syntax of chord notation and rendered as chord names in the `ChordNames` context, see [Printing chord names], page 313.

Drum mode

This is activated with the `\drummode` command, and causes input to be interpreted with the syntax of drum notation, see [Basic percussion notation], page 286.

Drum mode is also activated with the `\drums` command. This also creates a new `DrumStaff` context and causes the following input to be interpreted with the syntax of drum notation and rendered as drum symbols on a drum staff, see [Basic percussion notation], page 286.

Figure mode

This is activated with the `\figuremode` command, and causes input to be interpreted with the syntax of figured bass, see [Entering figured bass], page 321.

Figure mode is also activated with the `\figures` command. This also creates a new `FiguredBass` context and causes the following input to be interpreted with the figured bass

syntax and rendered as figured bass symbols in the `FiguredBass` context, see [Introduction to figured bass], page 320.

Fret and tab modes

There are no special input modes for entering fret and tab symbols.

To create tab diagrams, enter notes or chords in note mode and render them in a `TabStaff` context, see [Default tablatures], page 250.

To create fret diagrams above a staff, you have two choices. You can either use the `FretBoards` context (see [Automatic fret diagrams], page 277 or you can enter them as a markup above the notes using the `\fret-diagram` command (see [Fret diagram markups], page 259).

Lyrics mode

This is activated with the `\lyricmode` command, and causes input to be interpreted as lyric syllables with optional durations and associated lyric modifiers, see ⟨undefined⟩ [Vocal music], page ⟨undefined⟩.

Lyric mode is also activated with the `\addlyrics` command. This also creates a new `Lyrics` context and an implicit `\lyricsto` command which associates the following lyrics with the preceding music.

Markup mode

This is activated with the `\markup` command, and causes input to be interpreted with the syntax of markup, see Section A.8 [Text markup commands], page 515.

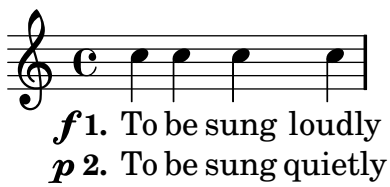
Note mode

This is the default mode or it may be activated with the `\notemode` command. Input is interpreted as pitches, durations, markup, etc and typeset as musical notation on a staff.

It is not normally necessary to specify note mode explicitly, but it may be useful to do so in certain situations, for example if you are in lyric mode, chord mode or any other mode and want to insert something that only can be done with note mode syntax.

For example, to indicate dynamic markings for the verses of a choral pieces it is necessary to enter note mode to interpret the markings:

```
{ c4 c4 c4 c4 }
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic f 1. } }
  To be sung loudly
}
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic p 2. } }
  To be sung quietly
}
```



5.4.2 Direction and placement

In typesetting music the direction and placement of many items is a matter of choice. For example, the stems of notes can be directed up or down; lyrics, dynamics, and other expressive marks may be placed above or below the staff; text may be aligned left, right or center; etc. Most of these choices may be left to be determined automatically by LilyPond, but in some cases it may be desirable to force a particular direction or placement.

Articulation direction indicators
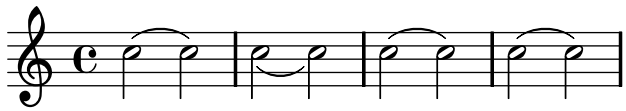
By default some directions are always up or always down (e.g. dynamics or fermata), while other things can alternate between up or down based on the stem direction (like slurs or accents).

The default action may be overridden by prefixing the articulation by a *direction indicator*. Three direction indicators are available: `^` (meaning « up »), `_` (meaning « down ») and `-` (meaning « use default direction »). The direction indicator can usually be omitted, in which case `-` is assumed, but a direction indicator is **always** required before

- `\tweak` commands
- `\markup` commands
- `\tag` commands
- string markups, e.g. `-"string"`
- fingering instructions, e.g. `-1`
- articulation shortcuts, e.g. `-.`, `->`, `--`

Direction indicators affect only the next note:

```
c2( c)
c2_( c)
c2( c)
c2^( c)
```



The direction property

The position or direction of many layout objects is controlled by the `direction` property.

The value of the `direction` property may be set to `1`, meaning « up » or « above », or to `-1`, meaning « down » or « below ». The symbols `UP` and `DOWN` may be used instead of `1` and `-1` respectively. The default direction may be specified by setting `direction` to `0` or `CENTER`. Alternatively, in many cases predefined commands exist to specify the direction. These are all of the form
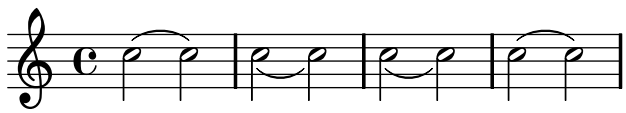
`\xxxUp`, `xxxDown`, `xxxNeutral`

where `xxxNeutral` means « use the default direction ». See Section “Within-staff objects” dans *Manuel d’initiation*.

In a few cases, arpeggio being the only common example, the value of the `direction` property specifies whether the object is to be placed to the right or left of the parent object. In this case `-1` or `LEFT` means « to the left » and `1` or `RIGHT` means « to the right ». `0` or `CENTER` means « use the default » direction, as before.

These indications affect all notes until they are cancelled.

```
c2( c)
\slurDown
c2( c)
c2( c)
\slurNeutral
c2( c)
```



5.4.3 Context layout order

Contexts are normally positioned in a system from top to bottom in the order in which they are encountered in the input file. When contexts are nested, the outer context will include inner nested contexts as specified in the input file, provided the inner contexts are included in the outer context's « accepts » list. Nested contexts which are not included in the outer context's « accepts » list will be repositioned below the outer context rather than nested within it.

Note that a context will be silently created implicitly if a command is encountered when there is no suitable context available to contain it. This can give rise to unexpected new staves or scores.

The default order in which contexts are laid out and the « accepts » list can be changed, see Section 5.1.7 [Aligning contexts], page 445.

Voir aussi

Usage Manual: Section "An extra staff appears" dans *Utilisation des programmes*.

5.4.4 Distances and measurements

Distances in LilyPond are of two types: absolute and scaled.

Absolute distances are used for specifying margins, indents, and other page layout details, and are by default specified in millimeters. Distances may be specified in other units by following the quantity by `\mm`, `\cm`, `\in` (inches), or `\pt` (points, 1/72.27 of an inch). Page layout distances can also be specified in scalable units (see the following paragraph) by appending `\staff-space` to the quantity. Page layout is described in detail in Section 4.1.2 [Page formatting], page 391.

Scaled distances are always specified in units of the staff-space or, rarely, the half staff-space. The staff-space is the distance between two adjacent staff lines. The default value can be changed globally by setting the global staff size, or it can be overridden locally by changing the `staff-space` property of `StaffSymbol`. Scaled distances automatically scale with any change to the either the global staff size or the `staff-space` property of `StaffSymbol`, but fonts scale automatically only with changes to the global staff size. The global staff size thus enables the overall size of a rendered score to be easily varied. For the methods of setting the global staff size see Section 4.2.1 [Setting the staff size], page 398.

If just a section of a score needs to be rendered to a different scale, for example an ossia section or a footnote, the global staff size cannot simply be changed as this would affect the entire score. In such cases the change in size is made by overriding both the `staff-space` property of `StaffSymbol` and the size of the fonts. A Scheme function, `magstep`, is available to convert from a font size change to the equivalent change in `staff-space`. For an explanation and an example of its use, see Section "Length and thickness of objects" dans *Manuel d'initiation*.

Voir aussi

Learning Manual: Section "Length and thickness of objects" dans *Manuel d'initiation*.
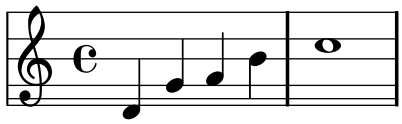
Notation Reference: Section 4.1.2 [Page formatting], page 391, Section 4.2.1 [Setting the staff size], page 398.

5.4.5 Staff symbol properties

The vertical position of staff lines and the number of staff lines can be defined at the same time. As the following example shows, note positions are not influenced by the staff line positions.
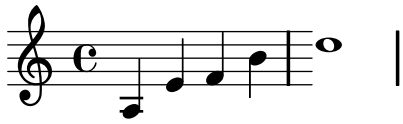
Note : The `'line-positions` property overrides the `'line-count` property. The number of staff lines is implicitly defined by the number of elements in the list of values for `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol #'line-positions = #'(7 3 0 -4 -6 -7)
}
{ a4 e' f b | d1 }
```



The width of a staff can be modified. The units are staff spaces. The spacing of objects inside the staff is not affected by this setting.

```
\new Staff \with {
  \override StaffSymbol #'width = #23
}
{ a4 e' f b | d1 }
```



5.4.6 Spanners

Many objects of musical notation extend over several notes or even several bars. Examples are slurs, beams, tuplet brackets, volta repeat brackets, crescendi, trills, and glissandi. Such objects are collectively called « spanners », and have special properties to control their appearance and behaviour. Some of these properties are common to all spanners; others are restricted to a sub-set of the spanners.

All spanners support the `spanner-interface`. A few, essentially those that draw a straight line between the two objects, support in addition the `line-spanner-interface`.
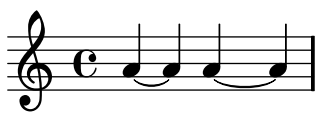
Using the `spanner-interface`

This interface provides two properties that apply to several spanners.

The `minimum-length` *property*

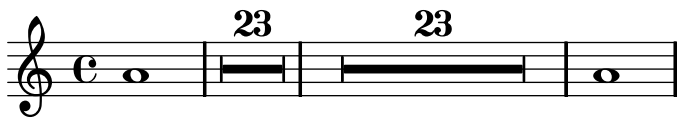
The minimum length of the spanner is specified by the `minimum-length` property. Increasing this usually has the necessary effect of increasing the spacing of the notes between the two end points. However, this override has no effect on many spanners, as their length is determined by other considerations. A few examples where it is effective are shown below.

```
a~a
a
% increase the length of the tie
-\tweak #'minimum-length #5
~a
```
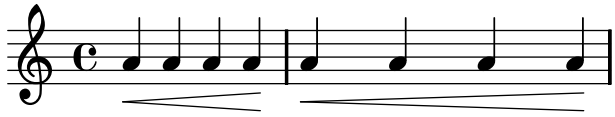


```
a1
\compressFullBarRests
R1*23
% increase the length of the rest bar
```

```
\once \override MultiMeasureRest #'minimum-length = #20
R1*23
a1
```




```
a \< a a a \!
% increase the length of the hairpin
\override Hairpin #'minimum-length = #20
a \< a a a \!
```



This override can also be used to increase the length of slurs and phrasing slurs:

```
a( a)
a
-\tweak #'minimum-length #5
( a)

a\( a\)
a
-\tweak #'minimum-length #5
\( a\)
```
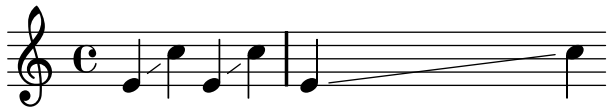


For some layout objects, the `minimum-length` property becomes effective only if the `set-spacing-rods` procedure is called explicitly. To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`. For example, the minimum length of a glissando has no effect unless the `springs-and-rods` property is set:

```
% default
e \glissando c'

% not effective alone
\once \override Glissando #'minimum-length = #20
e, \glissando c'
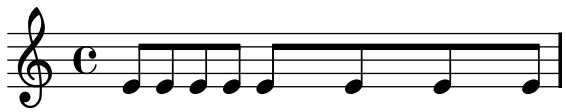
% effective only when both overrides are present
\once \override Glissando #'minimum-length = #20
\once \override Glissando #'springs-and-rods = #ly:spanner::set-spacing-rods
e, \glissando c'
```



The same is true of the `Beam` object:

```
% not effective alone
\once \override Beam #'minimum-length = #20
e8 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
e8 e e e
```
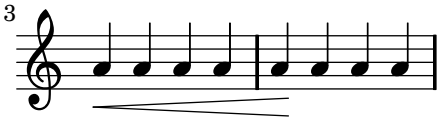


The `to-barline` *property*

The second useful property of the `spanner-interface` is `to-barline`. By default this is true, causing hairpins and other spanners which are terminated on the first note of a measure to end instead on the immediately preceding bar line. If set to false, the spanner will extend beyond the bar line and end on the note itself:

```
a \< a a a a \! a a a \break
\override Hairpin #'to-barline = ##f
a \< a a a a \! a a a
```





This property is not effective for all spanners. For example, setting it to `#t` has no effect on slurs or phrasing slurs or on other spanners for which terminating on the bar line would not be meaningful.
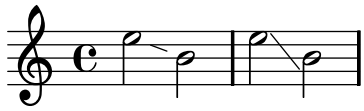
Using the `line-spanner-interface`

Objects which support the `line-spanner-interface` include

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

The routine responsible for drawing the stencils for these spanners is `ly:line-interface::print`. This routine determines the exact location of the two end points and draws a line between them, in the style requested. The locations of the two end points of the spanner are computed on-the-fly, but it is possible to override their Y-coordinates. The properties which need to be specified are nested two levels down within the property hierarchy, but the syntax of the `\override` command is quite simple:

```
e2 \glissando b
\once \override Glissando #'(bound-details left Y) = #3
\once \override Glissando #'(bound-details right Y) = #-2
e2 \glissando b
```
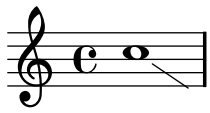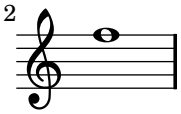


The units for the `Y` property are `staff-space`s, with the center line of the staff being the zero point. For the glissando, this is the value for `Y` at the X-coordinate corresponding to the center point of each note head, if the line is imagined to be extended to there.

If `Y` is not set, the value is computed from the vertical position of the corresponding attachment point of the spanner.

In case of a line break, the values for the end points are specified by the `left-broken` and `right-broken` sub-lists of `bound-details`. For example:

```
\override Glissando #'breakable = ##t
\override Glissando #'(bound-details right-broken Y) = #-3
c1 \glissando \break
f1
```





A number of further properties of the `left` and `right` sub-lists of the `bound-details` property may be modified in the same way as `Y`:

`Y` This sets the Y-coordinate of the end point, in `staff-space`s offset from the staff center line. By default, it is the center of the bound object, so a glissando points to the vertical center of the note head.

For horizontal spanners, such as text spanners and trill spanners, it is hardcoded to 0.
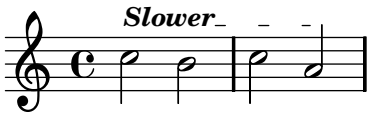
`attach-dir` This determines where the line starts and ends in the X-direction, relative to the bound object. So, a value of `-1` (or `LEFT`) makes the line start/end at the left side of the note head it is attached to.

`X` This is the absolute X-coordinate of the end point. It is usually computed on the fly, and overriding it has little useful effect.

`stencil` Line spanners may have symbols at the beginning or end, which is contained in this sub-property. This is for internal use; it is recommended that `text` be used instead.

`text` This is a markup that is evaluated to yield the stencil. It is used to put *cresc.*, *tr* and other text on horizontal spanners.

```
\override TextSpanner #'(bound-details left text)
   = \markup { \small \bold Slower }
c2\startTextSpan b c a\stopTextSpan
```
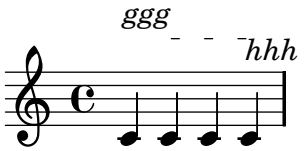


`stencil-align-dir-y`
`stencil-offset`

Without setting one of these, the stencil is simply put at the end-point, centered on the line, as defined by the `X` and `Y` sub-properties. Setting either `stencil-align-dir-y` or `stencil-offset` will move the symbol at the edge vertically relative to the end point of the line:

```
\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"
c4^\startTextSpan c c c \stopTextSpan
```
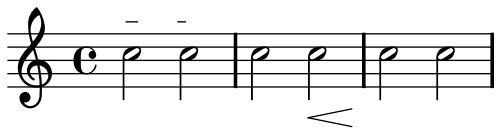


Note that negative values move the text *up*, contrary to the effect that might be expected, as a value of `-1` or `DOWN` means align the *bottom* edge of the text with the spanner line. A value of `1` or `UP` aligns the top edge of the text with the spanner line.

`arrow` Setting this sub-property to `#t` produces an arrowhead at the end of the line.

`padding` This sub-property controls the space between the specified end point of the line and the actual end. Without padding, a glissando would start and end in the center of each note head.

The music function `\endSpanners` terminates the spanner which starts on the immediately following note prematurely. It is terminated after exactly one note, or at the following bar line if `to-barline` is true and a bar line occurs before the next note.

```
\endSpanners
c2 \startTextSpan c2 c2
\endSpanners
c2 \< c2 c2
```



When using `\endSpanners` it is not necessary to close \startTextSpan with \stopTextSpan, nor is it necessary to close hairpins with `\!`.

Voir aussi

Internals Reference: Section "TextSpanner" dans *Référence des propriétés internes*, Section "Glissando" dans *Référence des propriétés internes*, Section "VoiceFollower" dans *Référence des propriétés internes*, Section "TrillSpanner" dans *Référence des propriétés internes*, Section "line-spanner-interface" dans *Référence des propriétés internes*.

5.4.7 Visibility of objects

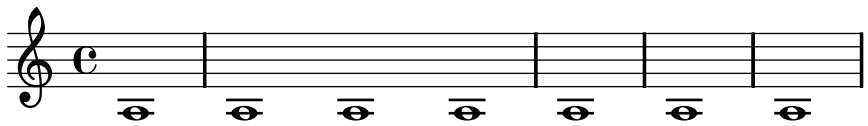
There are four main ways in which the visibility of layout objects can be controlled: their stencil can be removed, they can be made transparent, they can be colored white, or their `break-visibility` property can be overridden. The first three apply to all layout objects; the last to just a few – the *breakable* objects. The Learning Manual introduces these four techniques, see Section "Visibility and color of objects" dans *Manuel d'initiation*.

There are also a few other techniques which are specific to certain layout objects. These are covered under Special considerations.

Removing the stencil

Every layout object has a stencil property. By default this is set to the specific function which draws that object. If this property is overridden to `#f` no function will be called and the object will not be drawn. The default action can be recovered with `\revert`.
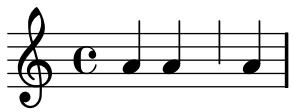
```
a1 a
\override Score.BarLine #'stencil = ##f
a a
\revert Score.BarLine #'stencil
a a a
```



Making objects transparent

Every layout object has a transparent property which by default is set to `#f`. If set to `#t` the object still occupies space but is made invisible.

```
a4 a
\once \override NoteHead #'transparent = ##t
a a
```



Painting objects white

Every layout object has a color property which by default is set to `black`. If this is overridden to `white` the object will be indistinguishable from the white background. However, if the object crosses other objects the color of the crossing points will be determined by the order in which they are drawn, and this may leave a ghostly image of the white object, as shown here:

```
\override Staff.Clef #'color = #white
a1
```



This may be avoided by changing the order of printing the objects. All layout objects have a `layer` property which should be set to an integer. 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`, although a few objects, including `StaffSymbol` and `BarLine`, are assigned a value of `0`. The order of printing objects with the same value of `layer` is indeterminate.

In the example above the white clef, with a default `layer` value of `1`, is drawn after the staff lines (default `layer` value `0`), so overwriting them. To change this, the `Clef` object must be given in a lower value of `layer`, say `-1`, so that it is drawn earlier:

```
\override Staff.Clef #'color = #white
\override Staff.Clef #'layer = #-1
a1
```



Using break-visibility

Most layout objects are printed only once, but some like bar lines, clefs, time signatures and key signatures, may need to be printed twice when a line break occurs – once at the end of the line and again at the start of the next line. Such objects are called *breakable*, and have a property, the `break-visibility` property to control their visibility at the three positions in which they may appear – at the start of a line, within a line if they are changed, and at the end of a line if a change takes place there.

For example, the time signature by default will be printed at the start of the first line, but nowhere else unless it changes, when it will be printed at the point at which the change occurs. If this change occurs at the end of a line the new time signature will be printed at the start of the next line and a cautionary time signature will be printed at the end of the previous line as well.

This behaviour is controlled by the `break-visibility` property, which is explained in Section "Visibility and color of objects" dans *Manuel d'initiation*. This property takes a vector of three booleans which, in order, determine whether the object is printed at the end of, within the body of, or at the beginning of a line. Or to be more precise, before a line break, where there is no line break, or after a line break.

Alternatively, these eight combinations may be specified by pre-defined functions, defined in '`scm/output-lib.scm`', where the last three columns indicate whether the layout objects will be visible in the positions shown at the head of the columns:

Function	**Vector**	**Before**	**At no**	**After**
form	**form**	**break**	**break**	**break**
`all-visible`	`'#(#t #t #t)`	yes	yes	yes
`begin-of-line-visible`	`'#(#f #f #t)`	no	no	yes
`center-visible`	`'#(#f #t #f)`	no	yes	no
`end-of-line-visible`	`'#(#t #f #f)`	yes	no	no
`begin-of-line-invisible`	`'#(#t #t #f)`	yes	yes	no
`center-invisible`	`'#(#t #f #t)`	yes	no	yes
`end-of-line-invisible`	`'#(#f #t #t)`	no	yes	yes
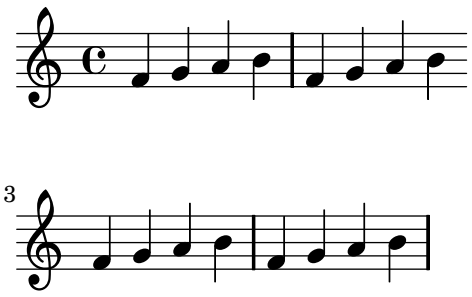
`all-invisible` `'#(#f #f #f)` no no no

The default settings of `break-visibility` depend on the layout object. The following table shows all the layout objects of interest which are affected by `break-visibility` and the default setting of this property:

Layout object	**Usual context**	**Default setting**
`BarLine`	`Score`	calculated
`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`
`OctavateEight`	`Staff`	`begin-of-line-visible`
`RehearsalMark`	`Score`	`end-of-line-invisible`
`TimeSignature`	`Staff`	`all-visible`

The example below shows the use of the vector form to control the visibility of barlines:

```
f4 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
```



Although all three components of the vector used to override `break-visibility` must be present, not all of them are effective with every layout object, and some combinations may even give errors. The following limitations apply:
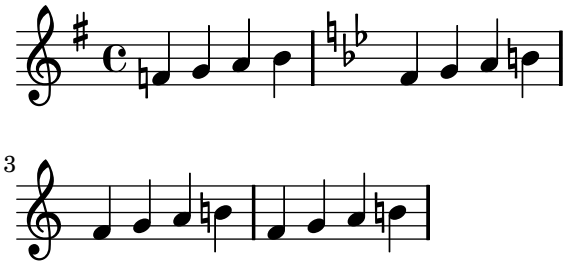
- Bar lines cannot be printed at start of line.
- A bar number cannot be printed at the start of the first line unless it is set to be different from 1.
- Clef – see below
- Double percent repeats are either all printed or all suppressed. Use begin-of line-invisible to print and all-invisible to suppress.
- Key signature – see below
- OctavateEight – see below

Special considerations

Visibility following explicit changes
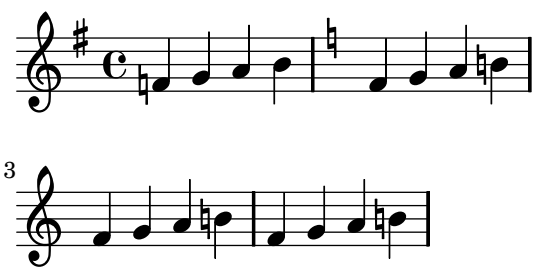
The `break-visibility` property controls the visibility of key signatures and changes of clef only at the start of lines, i.e. after a break. It has no effect on the visibility of the key signature or clef following an explicit key change or an explicit clef change within or at the end of a line. In the following example the key signature following the explicit change to B-flat major is still visible, even though `all-invisible` is set.

```
\key g \major
f4 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
f4 g a b
```



The visibility of such explicit key signature and clef changes is controlled by the `explicitKeySignatureVisibility` and `explicitClefVisibility` properties. These are the equivalent of the `break-visibility` property and both take a vector of three booleans or the predefined functions listed above, exactly like `break-visibility`. Both are properties of the Staff context, not the layout objects themselves, and so they are set using the `\set` command. Both are set by default to `all-visible`. These properties control only the visibility of key signatures and clefs resulting from explicit changes and do not affect key signatures and clefs at the beginning of lines; `break-visibility` must still be overridden in the appropriate object to remove these.
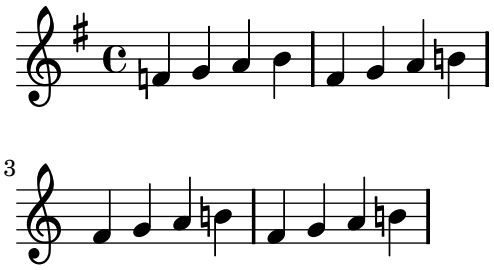
```
\key g \major
f4 g a b
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Staff.KeySignature #'break-visibility = #all-invisible
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
```



Visibility of cautionary accidentals

To remove the cautionary accidentals printed at an explicit key change, set the Staff context property `printKeyCancellation` to `#f`:

```
\key g \major
f4 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
```



With these overrides only the accidentals before the notes remain to indicate the change of key.

Automatic bars

As a special case, the printing of bar lines can also be turned off by setting the `automaticBars` property in the Score context. If set to `#f`, bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` predefined command, measures are still counted. Bar generation will resume according to that count if this property is later set to `#t`. When set to `#f`, line breaks can occur only at explicit `\bar` commands.

Octavated clefs

The small octavation symbol on octavated clefs is produced by the `OctavateEight` layout object. Its visibility is controlled independently from that of the `Clef` object, so it is necessary to apply any required `break-visibility` overrides to both the `Clef` and the `OctavateEight` layout objects to fully suppress such clef symbols at the start of each line.

For explicit clef changes, the `explicitClefVisibility` property controls both the clef symbol and any octavation symbol associated with it.

Voir aussi

Learning Manual: Section “Visibility and color of objects” dans *Manuel d’initiation*
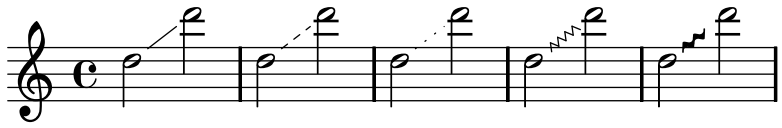
5.4.8 Line styles

Some performance indications, e.g., *rallentando* and *accelerando* and *trills* are written as text and are extended over many measures with lines, sometimes dotted or wavy.

These all use the same routines as the glissando for drawing the texts and the lines, and tuning their behavior is therefore also done in the same way. It is done with a spanner, and the routine responsible for drawing the spanners is `ly:line-interface::print`. This routine determines the exact location of the two *span points* and draws a line between them, in the style requested.
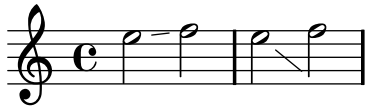
Here is an example showing the different line styles available, and how to tune them.

```
d2 \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
```



The locations of the end-points of the spanner are computed on-the-fly for every graphic object, but it is possible to override these:

```
e2 \glissando f
\once \override Glissando #'(bound-details right Y) = #-2
e2 \glissando f
```



The value for `Y` is set to `-2` for the right end point. The left side may be similarly adjusted by specifying `left` instead of `right`.

If `Y` is not set, the value is computed from the vertical position of the left and right attachment points of the spanner.

Other adjustments of spanners are possible, for details, see Section 5.4.6 [Spanners], page 461.

5.4.9 Rotating objects

Both layout objects and elements of markup text can be rotated by any angle about any point, but the method of doing so differs.
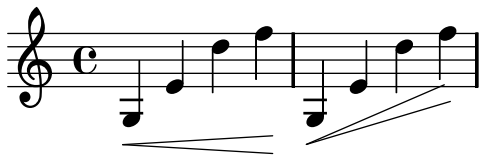
Rotating layout objects

All layout objects which support the `grob-interface` can be rotated by setting their `rotation` property. This takes a list of three items: the angle of rotation counter-clockwise, and the x and y coordinates of the point relative to the object's reference point about which the rotation is to be performed. The angle of rotation is specified in degrees and the coordinates in staff-spaces.

The angle of rotation and the coordinates of the rotation point must be determined by trial and error.

There are only a few situations where the rotation of layout objects is useful; the following example shows one situation where they may be:
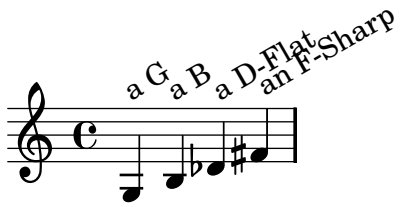
```
g4\< e' d' f\!
\override Hairpin #'rotation = #'(20 -1 0)
g,,4\< e' d' f\!
```



Rotating markup

All markup text can be rotated to lie at any angle by prefixing it with the `\rotate` command. The command takes two arguments: the angle of rotation in degrees counter-clockwise and the text to be rotated. The extents of the text are not rotated: they take their values from the extremes of the x and y coordinates of the rotated text. In the following example the `outside-staff-priority` property for text is set to `#f` to disable the automatic collision avoidance, which would push some of the text too high.

```
\override TextScript #'outside-staff-priority = ##f
g4^\markup { \rotate #30 "a G" }
b^\markup { \rotate #30 "a B" }
des^\markup { \rotate #30 "a D-Flat" }
fis^\markup { \rotate #30 "an F-Sharp" }
```



5.5 Advanced tweaks

This section discusses various approaches to fine tuning the appearance of the printed score.

Voir aussi

Learning Manual: Section “Tweaking output” dans *Manuel d’initiation*, Section “Other sources of information” dans *Manuel d’initiation*.

Notation Reference: Section 5.2 [Explaining the Internals Reference], page 446, Section 5.3 [Modifying properties], page 449.

Installed Files: ‘`scm/define-grobs.scm`’.

Snippets: Section “Tweaks and overrides” dans *Morceaux choisis*.

Extending: Section “Interfaces for programmers” dans *Extension de LilyPond*.

Internals Reference: Section “All layout objects” dans *Référence des propriétés internes*.

5.5.1 Aligning objects

Graphical objects which support the `self-alignment-interface` and/or the `side-position-interface` can be aligned to a previously placed object in a variety of ways. For a list of these objects, see Section “self-alignment-interface” dans *Référence des propriétés internes* and Section “side-position-interface” dans *Référence des propriétés internes*.

All graphical objects have a reference point, a horizontal extent and a vertical extent. The horizontal extent is a pair of numbers giving the displacements from the reference point of the left and right edges, displacements to the left being negative. The vertical extent is a pair of numbers giving the displacement from the reference point to the bottom and top edges, displacements down being negative.

An object’s position on a staff is given by the values of the `X-offset` and `Y-offset` properties. The value of `X-offset` gives the displacement from the X coordinate of the reference point of the parent object, and the value of `Y-offset` gives the displacement from the center line of the staff. The values of `X-offset` and `Y-offset` may be set directly or may be set to be calculated by procedures in order to achieve alignment with the parent object.

Note : Many objects have special positioning considerations which cause any setting of `X-offset` or `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface`. Overriding the `X-offset` or `Y-offset` properties to a fixed value causes the respective `self-alignment` property to be disregarded.

For example, an accidental can be repositioned vertically by setting `Y-offset` but any changes to `X-offset` have no effect.

Rehearsal marks may be aligned with breakable objects such as bar lines, clef symbols, time signature symbols and key signatures. There are special properties to be found in the `break-aligned-interface` for positioning rehearsal marks on such objects.

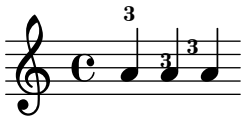
Voir aussi

[Using the break-alignable-interface], page 475, Section “Callback functions” dans *Extension de LilyPond*.

Setting `X-offset` and `Y-offset` directly

Numerical values may be given to the `X-offset` and `Y-offset` properties of many objects. The following example shows three notes with the default fingering position and the positions with `X-offset` and `Y-offset` modified.

```
a-3
a
-\tweak #'X-offset #0
-\tweak #'Y-offset #0
-3
a
-\tweak #'X-offset #-1
-\tweak #'Y-offset #1
-3
```



Using the `side-position-interface`

An object which supports the `side-position-interface` can be placed next to its parent object so that the specified edges of the two objects touch. The object may be placed above, below, to the right or to the left of the parent. The parent cannot be specified; it is determined by the order of elements in the input stream. Most objects have the associated note head as their parent.

The values of the `side-axis` and `direction` properties determine where the object is to be placed, as follows:

`side-axis` **property**	`direction` **property**	**Placement**
`0`	`-1`	left
`0`	`1`	right
`1`	`-1`	below
`1`	`1`	above

When `side-axis` is `0`, `X-offset` should be set to the procedure `ly:side-position-interface::x-aligned-side`. This procedure will return the correct value of `X-offset` to place the object to the left or right side of the parent according to value of `direction`.

When `side-axis` is `1`, `Y-offset` should be set to the procedure `ly:side-position-interface::y-aligned-side`. This procedure will return the correct value of `Y-offset` to place the object to the top or bottom of the parent according to value of `direction`.

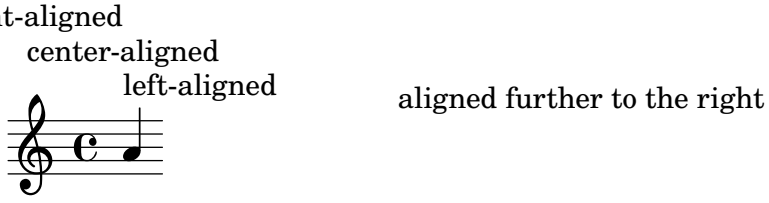
Using the `self-alignment-interface`

Self-aligning objects horizontally

The horizontal alignment of an object which supports the `self-alignment-interface` is controlled by the value of the `self-alignment-X` property, provided the object's `X-offset` property is set to `ly:self-alignment-interface::x-aligned-on-self`. `self-alignment-X` may be given any real value, in units of half the total X extent of the object. Negative values move the object to the right, positive to the left. A value of `0` centers the object on the reference point of its parent, a value of `-1` aligns the left edge of the object on the reference point of its parent, and a value of `1` aligns the right edge of the object on the reference point of its parent. The symbols `LEFT`, `CENTER` and `RIGHT` may be used instead of the values `-1, 0, 1` respectively.

Normally the `\override` command would be used to modify the value of `self-alignment-X`, but the `\tweak` command can be used to separately align several annotations on a single 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"
```



Self-aligning objects vertically
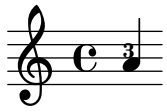
Objects may be aligned vertically in an analogous way to aligning them horizontally if the `Y-offset` property is set to `ly:self-alignment-interface::y-aligned-on-self`. However, other mechanisms are often involved in vertical alignment: the value of `Y-offset` is just one variable taken into account. This may make adjusting the value of some objects tricky. The units are just half the vertical extent of the object, which is usually quite small, so quite large numbers may be required. A value of `-1` aligns the lower edge of the object with the reference point of the parent object, a value of `0` aligns the center of the object with the reference point of the parent, and a value of `1` aligns the top edge of the object with the reference point of the parent. The symbols `DOWN`, `CENTER`, `UP` may be substituted for `-1, 0, 1` respectively.

Self-aligning objects in both directions

By setting both `X-offset` and `Y-offset`, an object may be aligned in both directions simultaneously.

The following example shows how to adjust a fingering mark so that it nestles close to the note head.

```
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
```

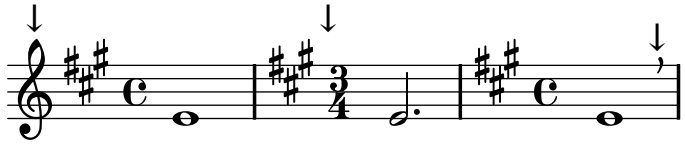


Using the break-alignable-interface

Rehearsal marks and bar numbers may be aligned with notation objects other than bar lines. These objects include ambitus, breathing-sign, clef, custos, staff-bar, left-edge, key-cancellation, key-signature, and time-signature.

By default, rehearsal marks and bar numbers will be horizontally centered above the object:

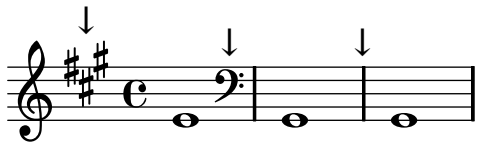
```
% The rehearsal mark will be centered above the Clef
\override Score.RehearsalMark #'break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark ""
e1
% The rehearsal mark will be centered above the Time Signature
\override Score.RehearsalMark #'break-align-symbols = #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark ""
e2.
% 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
e1
\breathe
\mark ""
```



A list of possible target alignment objects may be specified. If some of the objects are invisible at that point due to the setting of break-visibility or the explicit visibility settings for keys and clefs, the rehearsal mark or bar number is aligned to the first object in the list which is visible. If no objects in the list are visible the object is aligned to the bar line. If the bar line is invisible the object is aligned to the place where the bar line would be.

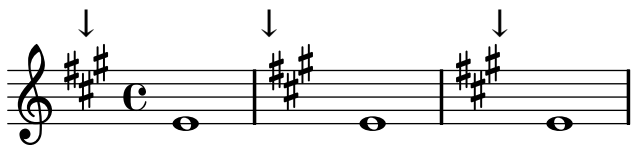
```
% The rehearsal mark will be centered above the Key Signature
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
```

```
\mark ""
e1
% The rehearsal mark will be centered above 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
```



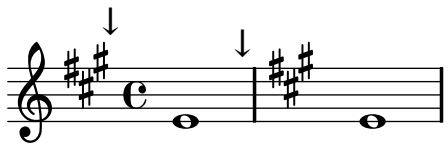
The alignment of the rehearsal mark relative to the notation object can be changed, as shown in the following example. In a score with multiple staves, this setting should be done for all the staves.

```
% The RehearsalMark will be centered above the Key Signature
\override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
\mark ""
e1
% The RehearsalMark will be aligned with the left edge of the Key Signature
\once \override Score.KeySignature #'break-align-anchor-alignment = #LEFT
\mark ""
\key a \major
e1
% The RehearsalMark will be aligned with the right edge of the Key Signature
\once \override Score.KeySignature #'break-align-anchor-alignment = #RIGHT
\key a \major
\mark ""
e1
```



The rehearsal mark can also be offset to the right or left of the left edge by an arbitrary amount. The units are staff-spaces:

```
% 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 ""
e1
% 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 ""
e1
```



5.5.2 Vertical grouping of grobs

The VerticalAlignment and VerticalAxisGroup grobs work together. VerticalAxisGroup groups together different grobs like Staff, Lyrics, etc. VerticalAlignment then vertically aligns the different grobs grouped together by VerticalAxisGroup. There is usually only one VerticalAlignment per score but every Staff, Lyrics, etc. has its own VerticalAxisGroup.

5.5.3 Modifying stencils

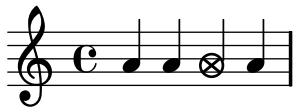
All layout objects have a `stencil` property which is part of the `grob-interface`. By default, this property is usually set to a function specific to the object that is tailor-made to render the symbol which represents it in the output. For example, the standard setting for the `stencil` property of the `MultiMeasureRest` object is `ly:multi-measure-rest::print`.

The standard symbol for any object can be replaced by modifying the `stencil` property to reference a different, specially-written, procedure. This requires a high level of knowledge of the internal workings of LilyPond, but there is an easier way which can often produce adequate results.

This is to set the `stencil` property to the procedure which prints text – `ly:text-interface::print` – and to add a `text` property to the object which is set to contain the markup text which produces the required symbol. Due to the flexibility of markup, much can be achieved – see in particular ⟨undefined⟩ [Graphic notation inside markup], page ⟨undefined⟩.

The following example demonstrates this by changing the note head symbol to a cross within a circle.

```
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 c'' {
  a a \XinO a a
}
```



Any of the glyphs in the feta Font can be supplied to the `\musicglyph` markup command – see Section A.6 [The Feta font], page 496.

Voir aussi

Notation Reference: ⟨undefined⟩ [Graphic notation inside markup], page ⟨undefined⟩, ⟨undefined⟩ [Formatting text], page ⟨undefined⟩, Section A.8 [Text markup commands], page 515, Section A.6 [The Feta font], page 496.

5.5.4 Modifying shapes
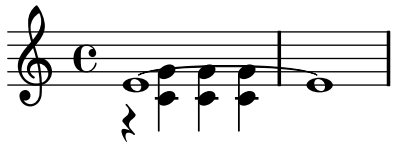
Modifying ties and slurs

Ties, slurs and phrasing slurs are drawn as third-order Bézier curves. If the shape of the tie or slur which is calculated automatically is not optimum, the shape may be modified manually by explicitly specifying the four control points required to define a third-order Bézier curve.

Third-order or cubic Bézier curves are defined by four control points. The first and fourth control points are precisely the starting and ending points of the curve. The intermediate two control points define the shape. Animations showing how the curve is drawn can be found on the web, but the following description may be helpful. The curve starts from the first control point heading directly towards the second, gradually bending over to head towards the third and continuing to bend over to head towards the fourth, arriving there travelling directly from the third control point. The curve is entirely contained in the quadrilateral defined by the four control points.

Here is an example of a case where the tie is not optimum, and where `\tieDown` would not help.

```
<<
  { e1 ~ e }
\\
  { r4 <g c,> <g c,> <g c,> }
>>
```
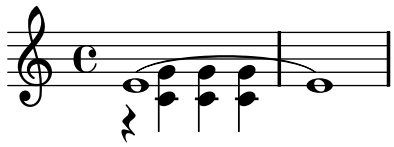


One way of improving this tie is to manually modify its control points, as follows.

The coordinates of the Bézier control points are specified in units of staff-spaces. The X coordinate is relative to the reference point of the note to which the tie or slur is attached, and the Y coordinate is relative to the staff center line. The coordinates are entered as a list of four pairs of decimal numbers (reals). One approach is to estimate the coordinates of the two end points, and then guess the two intermediate points. The optimum values are then found by trial and error.

It is useful to remember that a symmetric curve requires symmetric control points, and that Bézier curves have the useful property that transformations of the curve such as translation, rotation and scaling can be achieved by applying the same transformation to the curve's control points.

For the example above the following override gives a satisfactory tie. Note the placement – it has to be immediately before the note to which the start of the tie (or slur) is attached.

```
<<
  {
    \once \override Tie
      #'control-points = #'((1 . -1) (3 . 0.6) (12.5 . 0.6) (14.5 . -1))
     e1 ~ e1
  }
\\
  { r4 <g c,> <g c,> <g c,>4  }
>>
```



Problèmes connus et avertissements

It is not possible to modify shapes of ties or slurs by changing the `control-points` property if there are more than one at the same musical moment, not even by using the `\tweak` command. However, the `tie-configuration` property of `TieColumn` can be overridden to set start line and direction of ties as required.

5.6 Using music functions

Where tweaks need to be reused with different music expressions, it is often convenient to make the tweak part of a *music function*. In this section, we discuss only *substitution* functions, where the object is to substitute a variable into a piece of LilyPond input code. Other more complex functions are described in Section "Music functions" dans *Extension de LilyPond*.

5.6.1 Substitution function syntax

Making a function that substitutes a variable into LilyPond code is easy. The general form of these functions is

```
function =
#(define-music-function
     (parser location arg1 arg2 ...)
     (type1? type2? ...)
   #{
     ...music...
   #})
```

where

argN	*n*th argument
typeN?	a scheme *type predicate* for which *argN* must return `#t`.
...music...	normal LilyPond input, using `$` to reference arguments (eg. '`$arg1`').

The `parser` and `location` arguments are mandatory, and are used in some advanced situations as described in the 'Extending' manual (see Section "Music functions" dans *Extension de LilyPond*). For substitution functions, just be sure to include them.

The list of type predicates is also required. Some of the most common type predicates used in music functions are:

```
boolean?
cheap-list?  (use instead of 'list?'
 for faster processing)
ly:music?
markup?
number?
pair?
string?
symbol?
```

For a list of available type predicates, see Section A.16 [Predefined type predicates], page 591. User-defined type predicates are also allowed.

Voir aussi

Notation Reference: Section A.16 [Predefined type predicates], page 591.

Extending: Section "Music functions" dans *Extension de LilyPond*.

Installed Files: '`lily/music-scheme.cc`', '`scm/c++.scm`', '`scm/lily.scm`'.
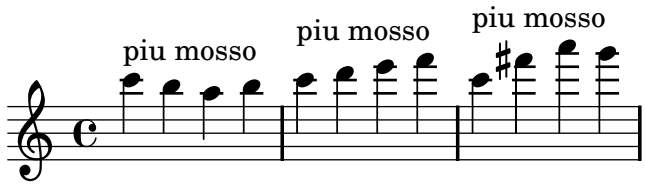
5.6.2 Substitution function examples

This section introduces some substitution function examples. These are not intended to be exhaustive, but rather to demonstrate some of the possibilities of simple substitution functions.

In the first example, a function is defined that simplifies setting the padding of a TextScript:

```
padText =
#(define-music-function
     (parser location padding)
     (number?)
   #{
     \once \override TextScript #'padding = $padding
   #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```
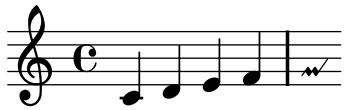


In addition to numbers, we can use music expressions such as notes for arguments to music functions:

```
custosNote =
#(define-music-function
     (parser location note)
     (ly:music?)
```

```
   #{
     \once \override Voice.NoteHead #'stencil =
       #ly:text-interface::print
     \once \override Voice.NoteHead #'text =
       \markup \musicglyph #"custodes.mensural.u0"
     \once \override Voice.Stem #'stencil = ##f
     $note
   #})

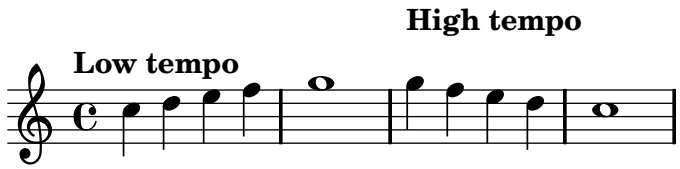
\relative c' { c4 d e f \custosNote g }
```



Substitution functions with multiple arguments can be defined:

```
tempoPadded =
#(define-music-function
     (parser location padding tempotext)
     (number? string?)
   #{
     \once \override Score.MetronomeMark #'padding = $padding
     \tempo \markup { \bold $tempotext }
   #})

\relative c'' {
  \tempo \markup { "Low tempo" }
  c4 d e f g1
  \tempoPadded #4.0 #"High tempo"
  g4 f e d c1
}
```



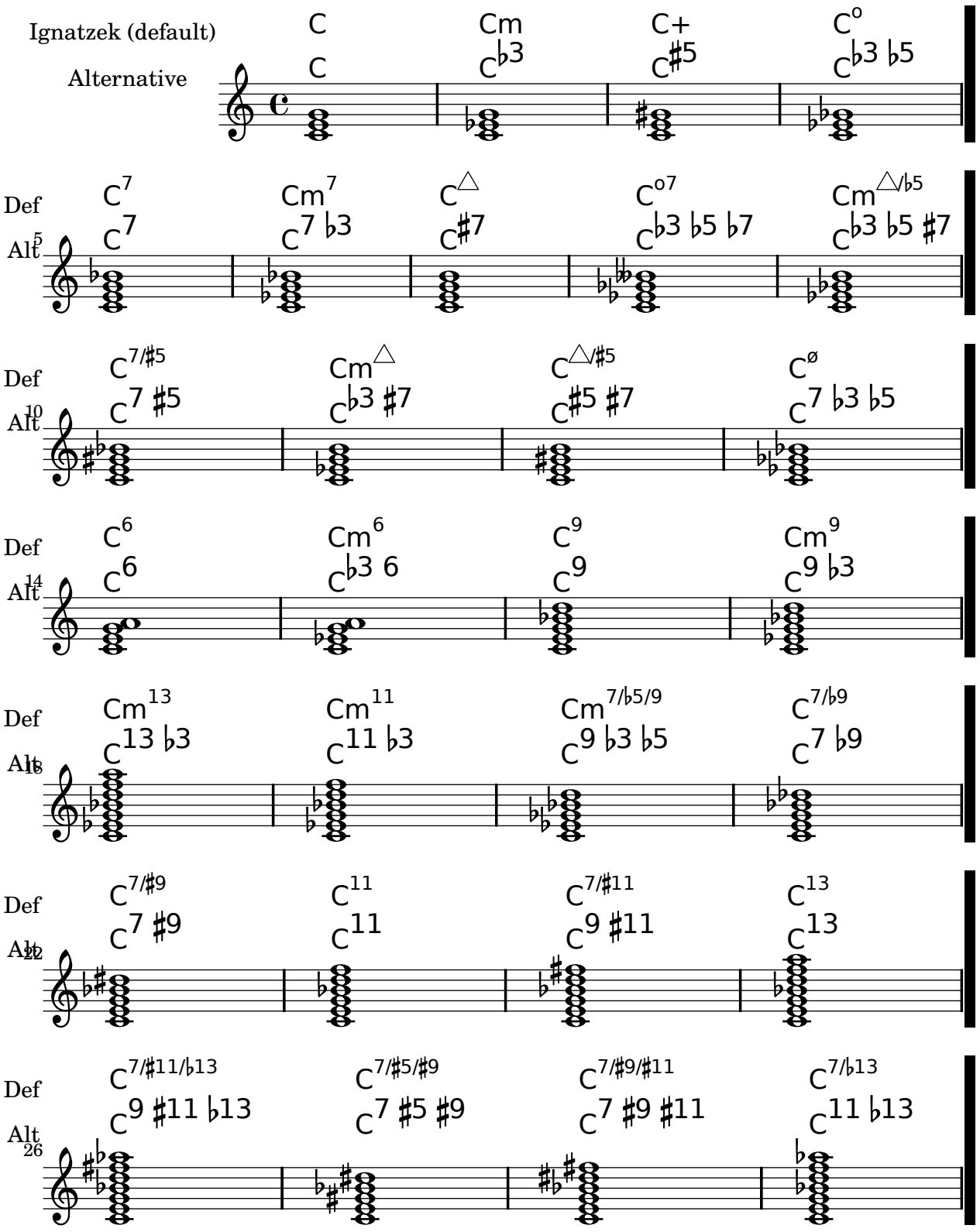
Voir aussi

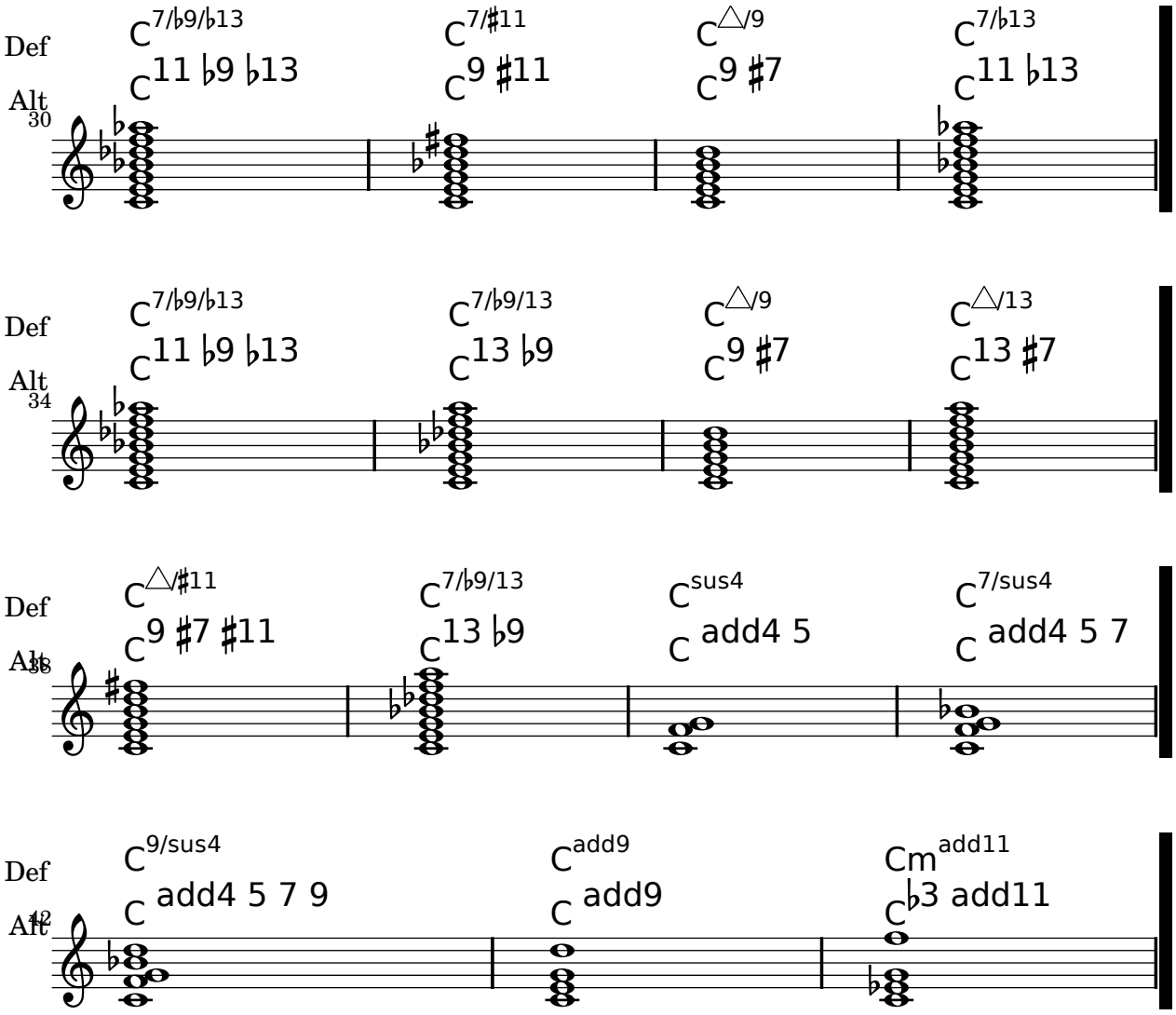
TODO: add missing @ref's here.

Annexe A Notation manual tables

A.1 Chord name chart

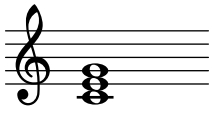
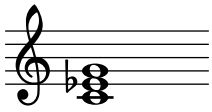
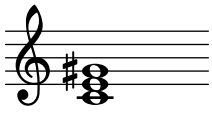
The following charts shows two standard systems for printing chord names, along with the pitches they represent.
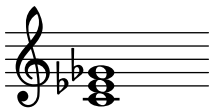
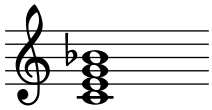
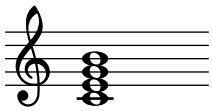
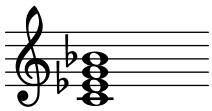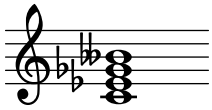
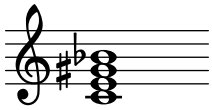
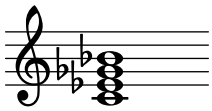
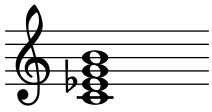
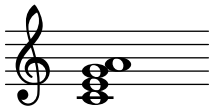
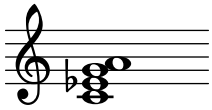


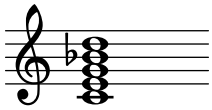
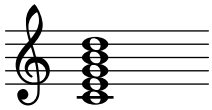
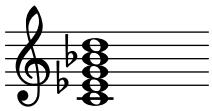
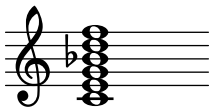
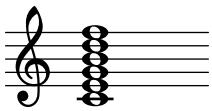
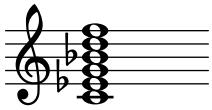
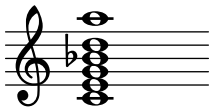
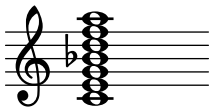
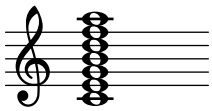
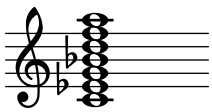


A.2 Common chord modifiers

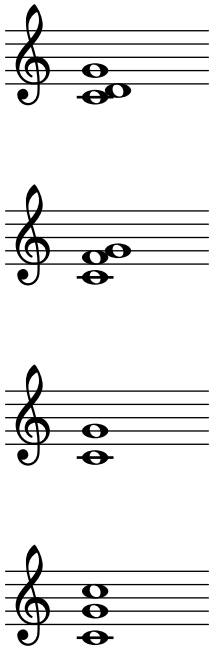
The following table shows chord modifiers that can be used in `\chordmode` to generate standard chord structures.

Chord type	Intervals	Modifier(s)	Example
Major	Major third, perfect fifth	`5` or nothing	
Minor	Minor third, perfect fifth	`m` or `m5`	
Augmented	Major third, augmented fifth	`aug`	

Diminished	Minor third, diminished fifth	`dim`	
Dominant seventh	Major triad, minor seventh	`7`	
Major seventh	Major triad, major seventh	`maj7` or `maj`	
Minor seventh	Minor triad, minor seventh	`m7`	
Diminished seventh	Diminished triad, diminished seventh	`dim7`	
Augmented seventh	Augmented triad, minor seventh	`aug7`	
Half-diminished seventh	Diminished triad, minor seventh	`m7.5-`	
Minor-major seventh	Minor triad, major seventh	`m7+`	
Major sixth	Major triad, sixth	`6`	
Minor sixth	Minor triad, sixth	`m6`	

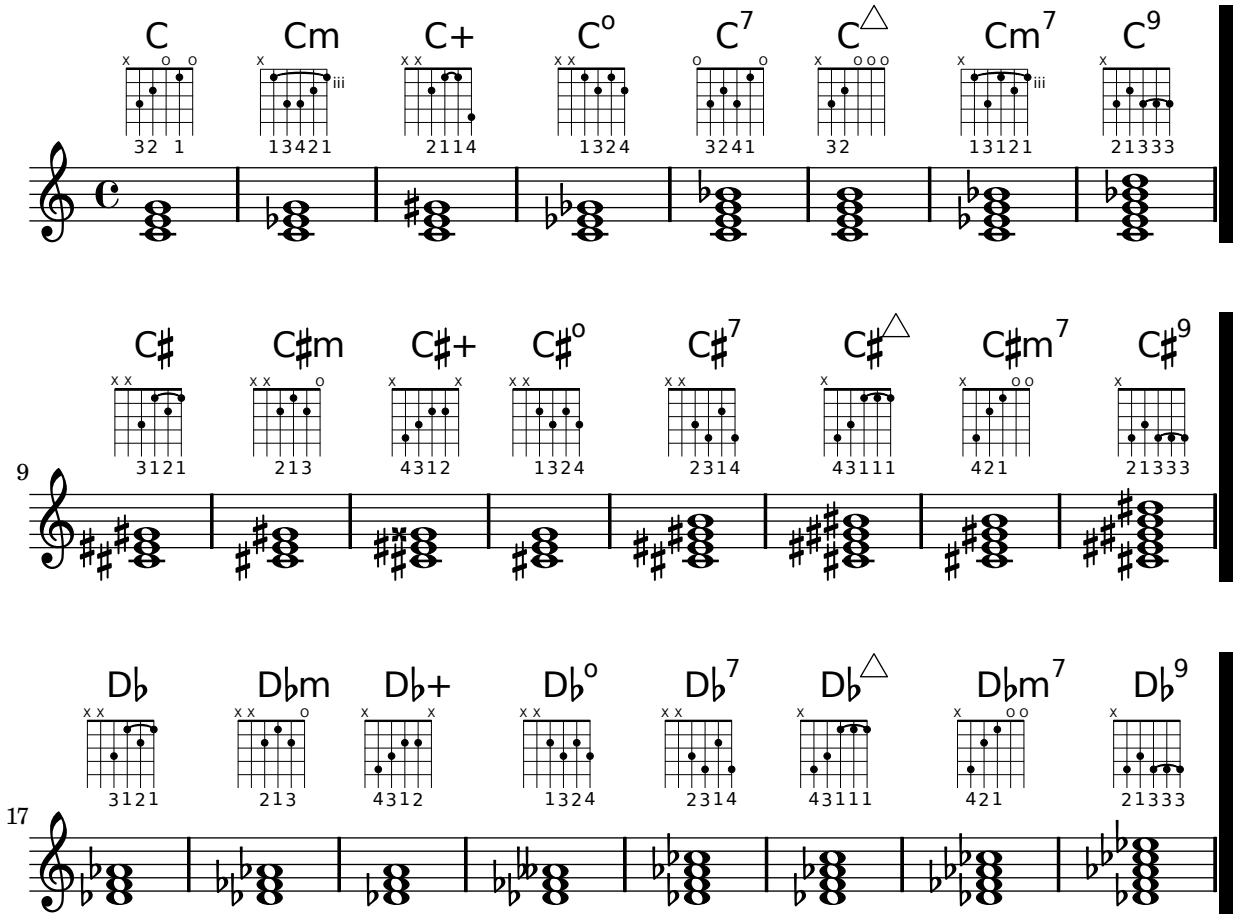
Dominant ninth	Dominant seventh, major ninth	`9`	
Major ninth	Major seventh, major ninth	`maj9`	
Minor ninth	Minor seventh, major ninth	`m9`	
Dominant eleventh	Dominant ninth, perfect eleventh	`11`	
Major eleventh	Major ninth, perfect eleventh	`maj11`	
Minor eleventh	Minor ninth, perfect eleventh	`m11`	
Dominant thirteenth	Dominant ninth, major thirteenth	`13`	
Dominant thirteenth	Dominant eleventh, major thirteenth	`13.11`	
Major thirteenth	Major eleventh, major thirteenth	`maj13.11`	
Minor thirteenth	Minor eleventh, major thirteenth	`m13.11`	

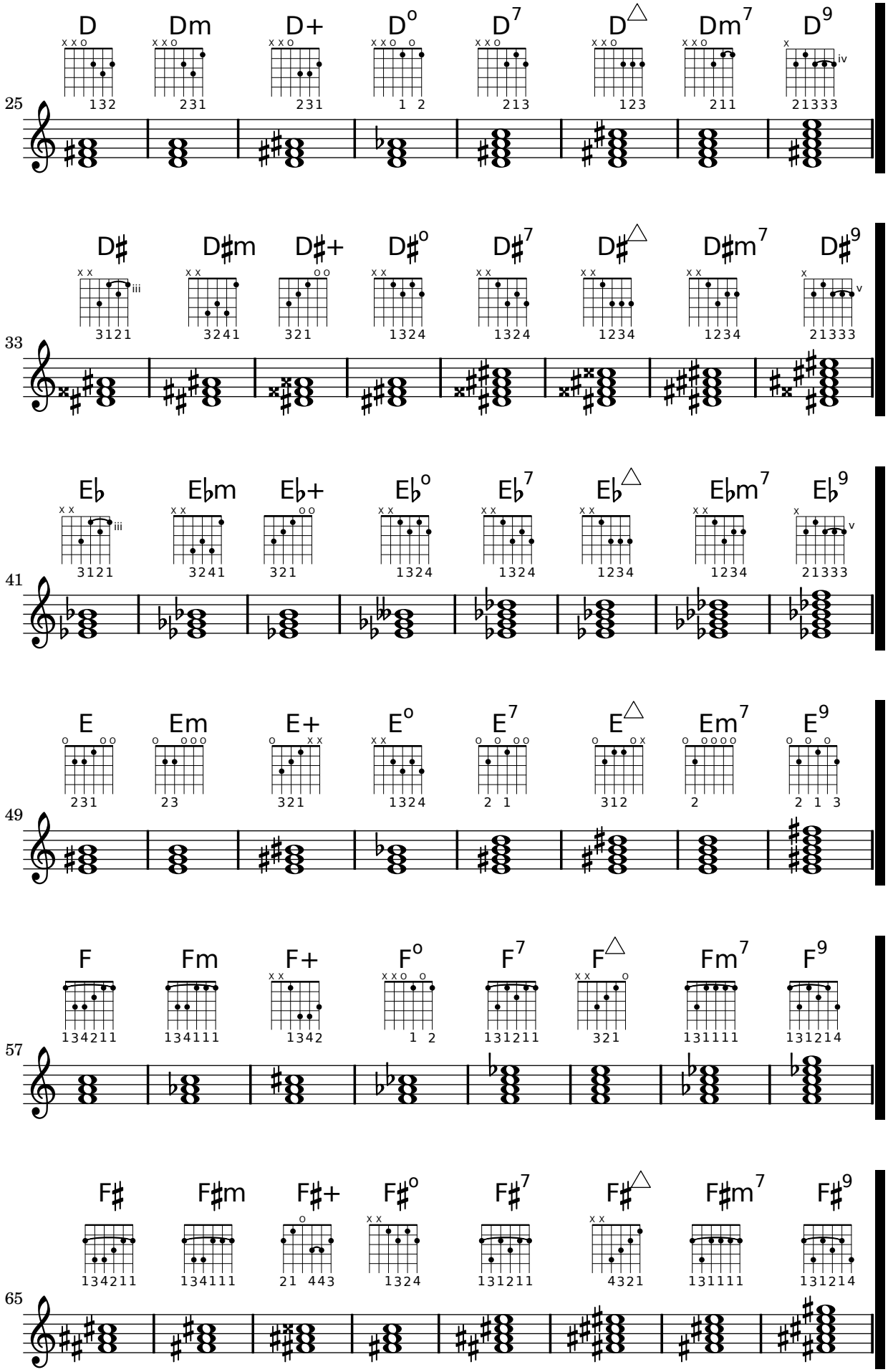
Suspended second	Major second, perfect fifth	`sus2`
Suspended fourth	Perfect fourth, perfect fifth	`sus4`
Power chord (two-voiced)	Perfect fifth	`1.5`
Power chord (three-voiced)	Perfect fifth, octave	`1.5.8`

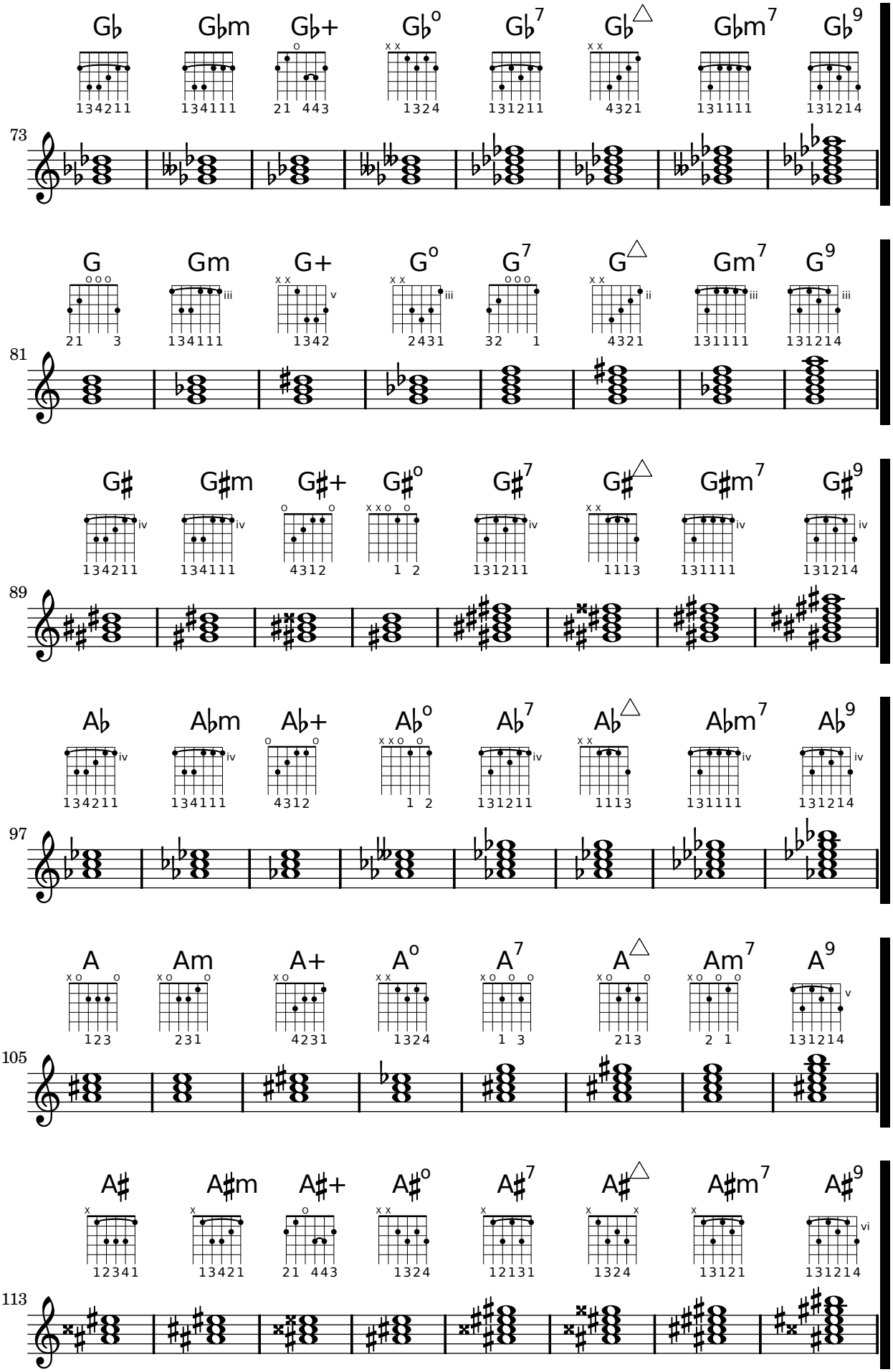


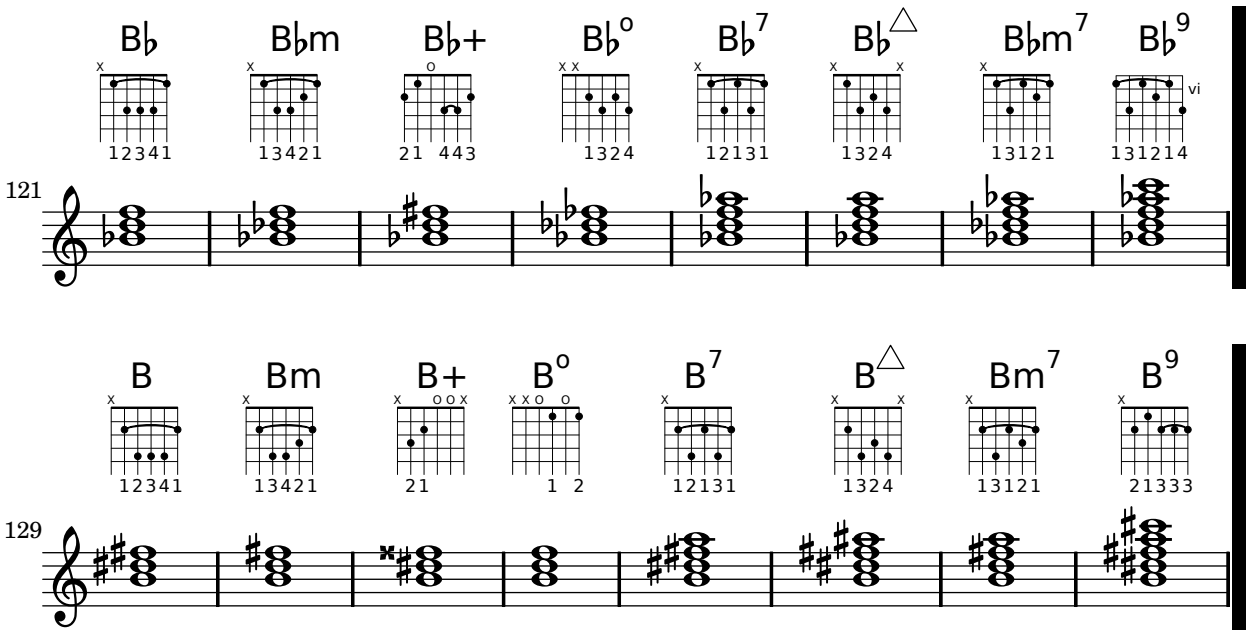
A.3 Predefined fretboard diagrams

The chart below shows the predefined fretboard diagrams for guitar.
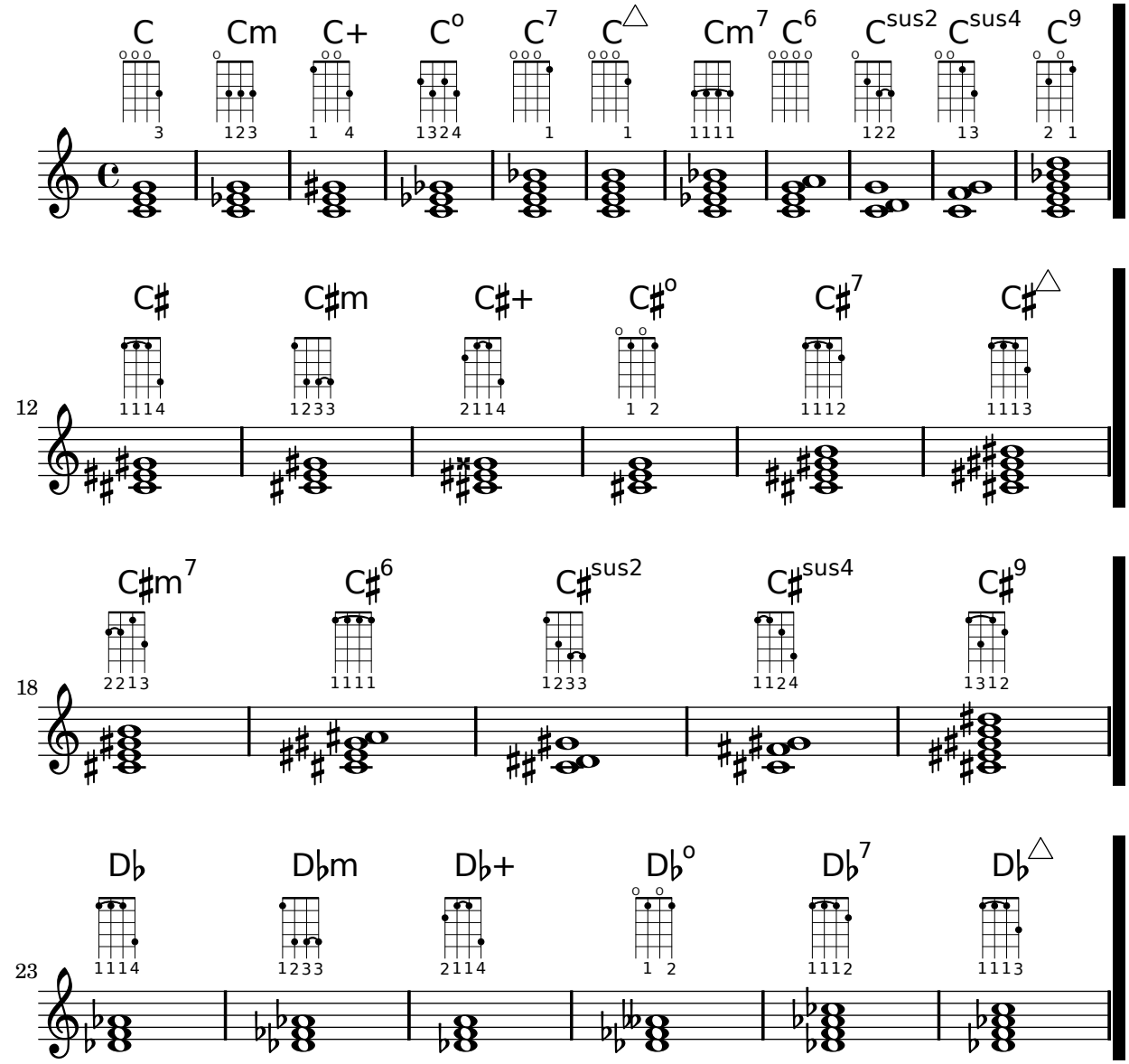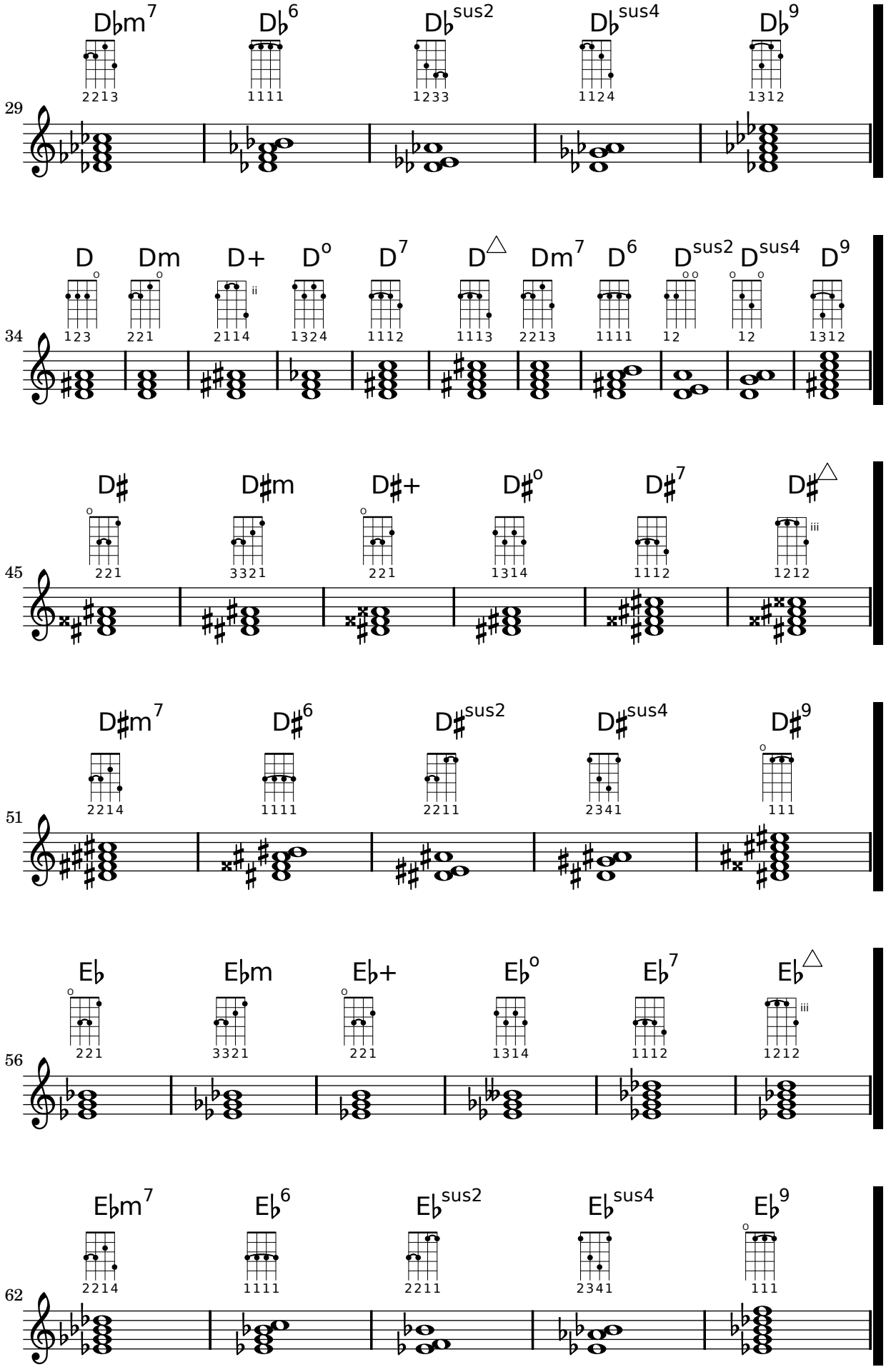



D Dm D+ D° D7 D△ Dm7 D9
132 231 231 1 2 213 123 211 21333
25
D♯ D♯m D♯+ D♯° D♯7 D♯△ D♯m7 D♯9
3121 3241 321 1324 1324 1234 1234 21333
33
E♭ E♭m E♭+ E♭° E♭7 E♭△ E♭m7 E♭9
3121 3241 321 1324 1324 1234 1234 21333
41
E Em E+ E° E7 E△ Em7 E9
231 23 321 1324 2 1 312 2 2 1 3
49
F Fm F+ F° F7 F△ Fm7 F9
134211 134111 1342 1 2 131211 321 131111 131214
57
F♯ F♯m F♯+ F♯° F♯7 F♯△ F♯m7 F♯9
134211 134111 21 443 1324 131211 4321 131111 131214
65

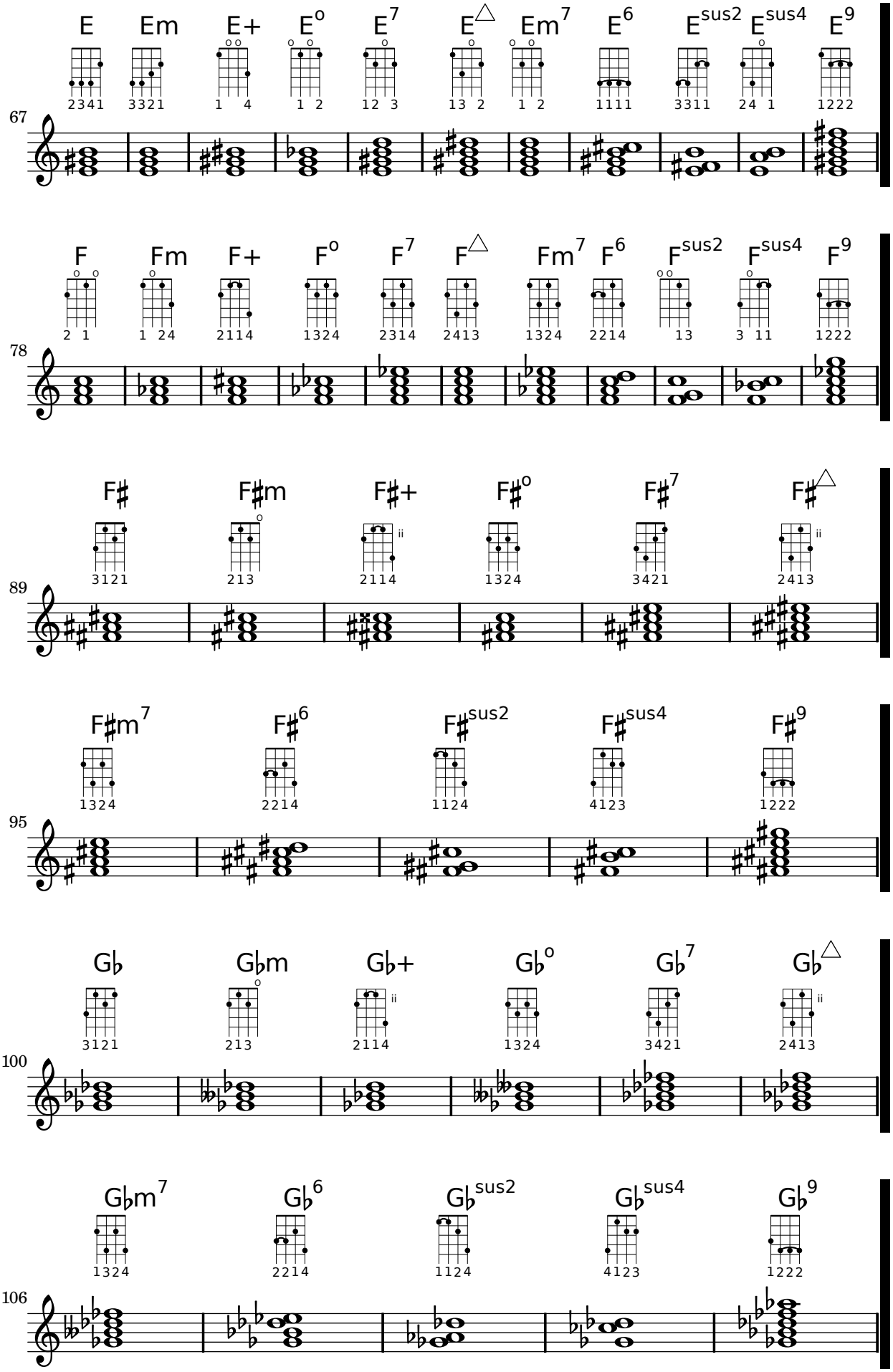

G♭ G♭m G♭+ G♭° G♭7 G♭△ G♭m7 G♭9
134211 134111 21 443 1324 131211 4321 131111 131214
73
G Gm G+ G° G7 G△ Gm7 G9
21 3 134111 1342 2431 32 1 4321 131111 131214
81
G♯ G♯m G♯+ G♯° G♯7 G♯△ G♯m7 G♯9
134211 134111 4312 1 2 131211 1113 131111 131214
89
A♭ A♭m A♭+ A♭° A♭7 A♭△ A♭m7 A♭9
134211 134111 4312 1 2 131211 1113 131111 131214
97
A Am A+ A° A7 A△ Am7 A9
123 231 4231 1324 1 3 213 2 1 131214
105
A♯ A♯m A♯+ A♯° A♯7 A♯△ A♯m7 A♯9
12341 13421 21 443 1324 12131 1324 13121 131214
113

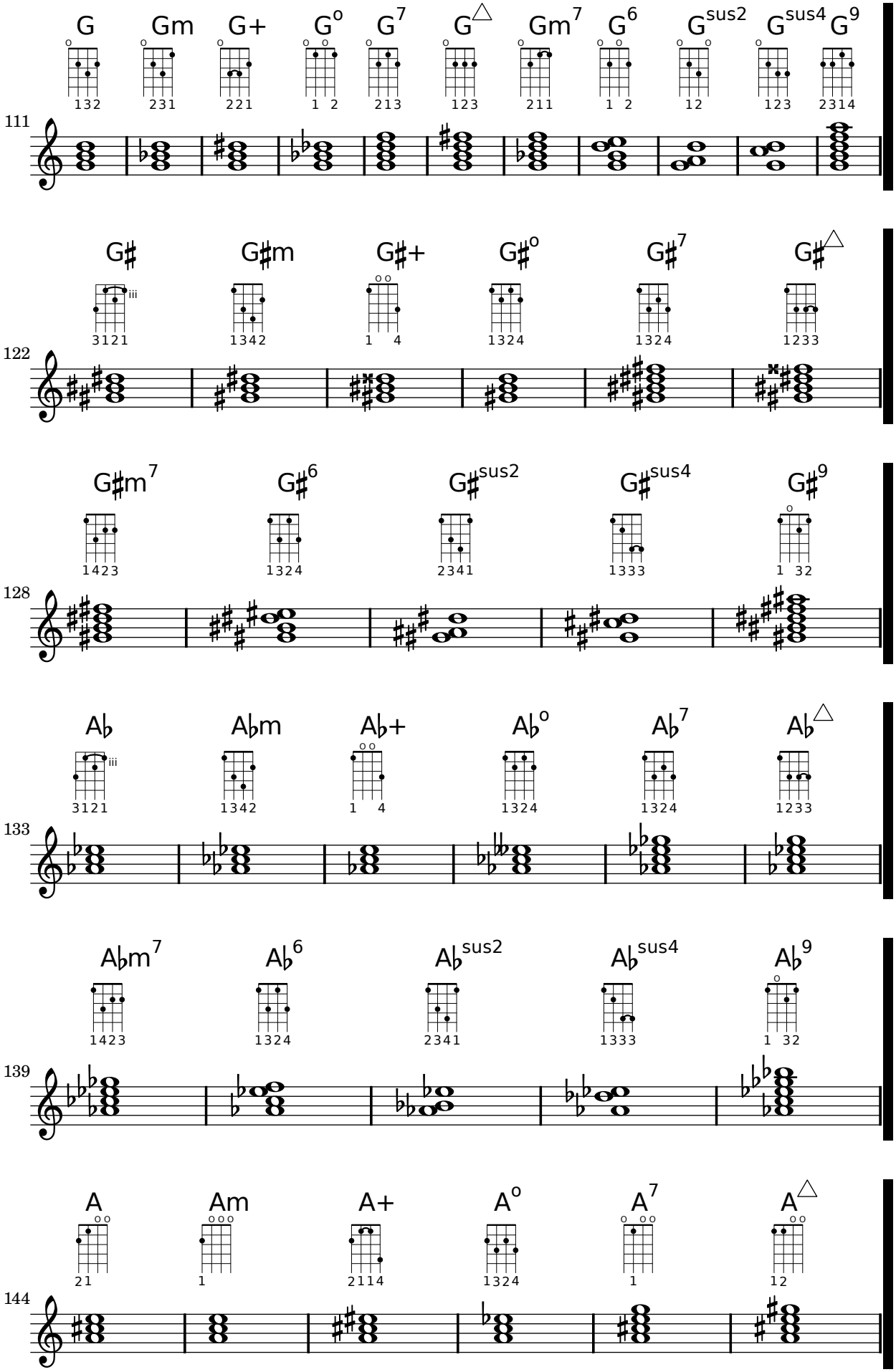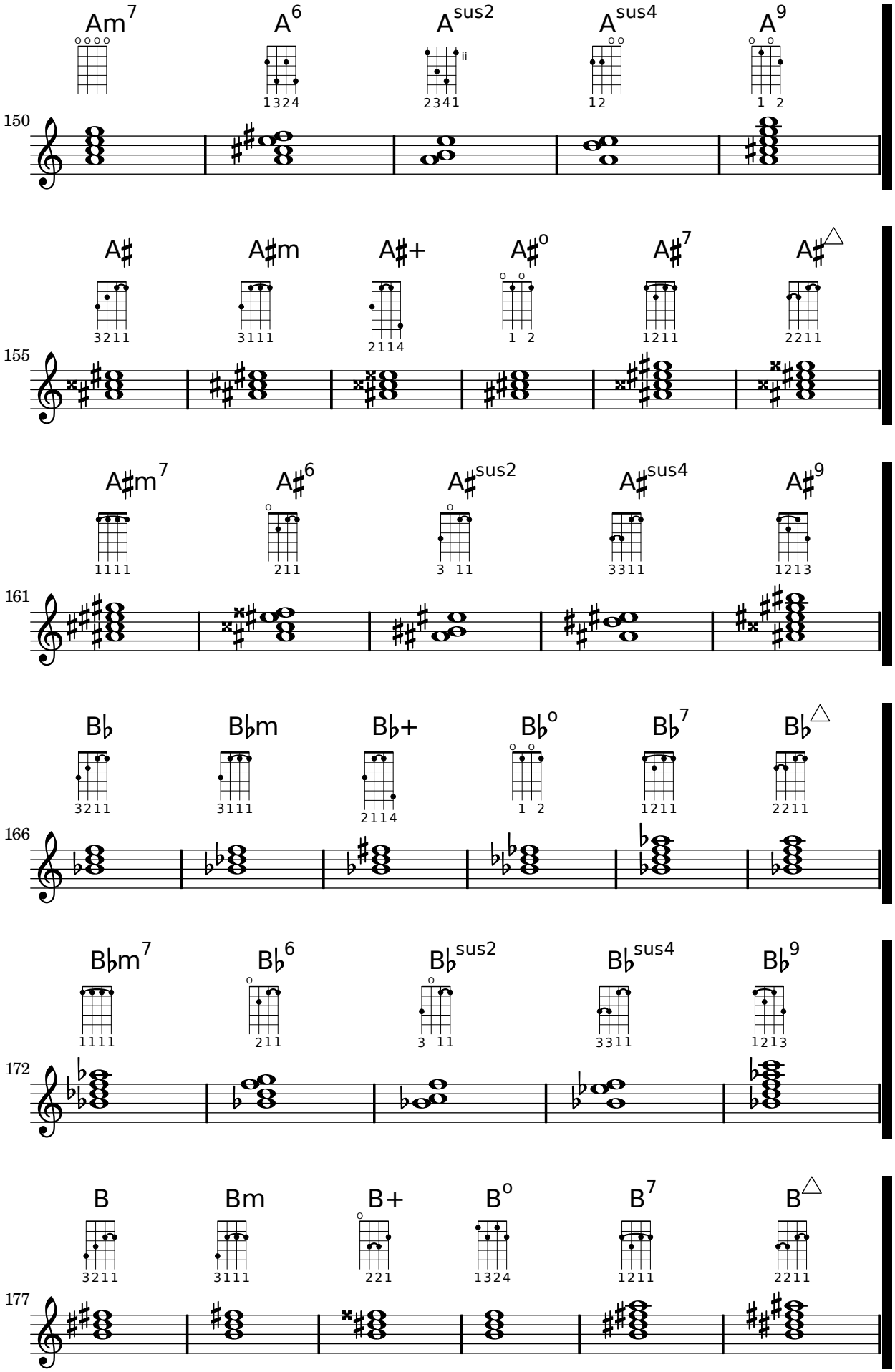


The chart below shows the predefined fretboard diagrams for ukulele.

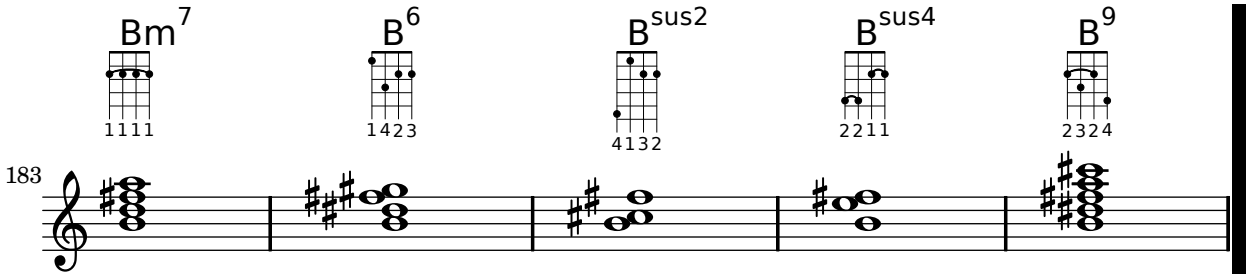



D♭m7
D♭6
D♭sus2
D♭sus4
D♭9
2213
1111
1233
1124
1312
29
D
Dm
D+
D°
D7
D△
Dm7
D6
Dsus2
Dsus4
D9
123
221
2114
1324
1112
1113
2213
1111
12
12
1312
34
D♯
D♯m
D♯+
D♯°
D♯7
D♯△
221
3321
221
1314
1112
1212
45
D♯m7
D♯6
D♯sus2
D♯sus4
D♯9
2214
1111
2211
2341
111
51
E♭
E♭m
E♭+
E♭°
E♭7
E♭△
221
3321
221
1314
1112
1212
56
E♭m7
E♭6
E♭sus2
E♭sus4
E♭9
2214
1111
2211
2341
111
62


E Em E+ E° E7 E△ Em7 E6 Esus2 Esus4 E9
2341 3321 1 4 1 2 12 3 13 2 1 2 1111 3311 24 1 1222
67
F Fm F+ F° F7 F△ Fm7 F6 Fsus2 Fsus4 F9
2 1 1 24 2114 1324 2314 2413 1324 2214 13 3 11 1222
78
F♯ F♯m F♯+ F♯° F♯7 F♯△
3121 213 2114 1324 3421 2413
89
F♯m7 F♯6 F♯sus2 F♯sus4 F♯9
1324 2214 1124 4123 1222
95
G♭ G♭m G♭+ G♭° G♭7 G♭△
3121 213 2114 1324 3421 2413
100
G♭m7 G♭6 G♭sus2 G♭sus4 G♭9
1324 2214 1124 4123 1222
106


G Gm G+ G° G7 G△ Gm7 G6 Gsus2 Gsus4 G9
132 231 221 1 2 213 123 211 1 2 12 123 2314
111
G♯ G♯m G♯+ G♯° G♯7 G♯△
3121 1342 1 4 1324 1324 1233
122
G♯m7 G♯6 G♯sus2 G♯sus4 G♯9
1423 1324 2341 1333 1 32
128
A♭ A♭m A♭+ A♭° A♭7 A♭△
3121 1342 1 4 1324 1324 1233
133
A♭m7 A♭6 A♭sus2 A♭sus4 A♭9
1423 1324 2341 1333 1 32
139
A Am A+ A° A7 A△
21 1 2114 1324 1 12
144


Am7
A6
Asus2
Asus4
A9
1324
2341
12
1 2
150
A♯
A♯m
A♯+
A♯o
A♯7
A♯△
3211
3111
2114
1 2
1211
2211
155
A♯m7
A♯6
A♯sus2
A♯sus4
A♯9
1111
211
3 11
3311
1213
161
B♭
B♭m
B♭+
B♭o
B♭7
B♭△
3211
3111
2114
1 2
1211
2211
166
B♭m7
B♭6
B♭sus2
B♭sus4
B♭9
1111
211
3 11
3311
1213
172
B
Bm
B+
Bo
B7
B△
3211
3111
221
1324
1211
2211
177



A.4 MIDI instruments

The following is a list of names that can be used for the `midiInstrument` property. The order of the instruments below, starting in the left-hand column moving down, corresponds to the General MIDI Standard's 128 Program Numbers.

```
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)
electric bass (pick)
fretless bass
slap bass 1
slap bass 2
synth bass 1
synth bass 2
violin
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
shakuhachi
whistle
ocarina
lead 1 (square)
lead 2 (sawtooth)
lead 3 (calliope)
lead 4 (chiff)
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
guitar fret noise
breath noise
seashore
bird tweet
telephone ring
helicopter
applause
```

```
viola                   lead 5 (charang)     gunshot
cello                   lead 6 (voice)
```

A.5 List of colors

Normal colors

Usage syntax is detailed in ⟨undefined⟩ [Coloring objects], page ⟨undefined⟩.

```
black       white          red          green
blue        cyan           magenta      yellow
grey        darkred        darkgreen    darkblue
darkcyan    darkmagenta    darkyellow
```

X color names

X color names come several variants:

Any name that is spelled as a single word with capitalization (e.g. 'LightSlateBlue') can also be spelled as space separated words without capitalization (e.g. 'light slate blue').

The word 'grey' can always be spelled 'gray' (e.g. 'DarkSlateGray').

Some names can take a numerical suffix (e.g. 'LightSalmon4').

Color Names without a numerical suffix:

```
snow             GhostWhite       WhiteSmoke       gainsboro        FloralWhite
OldLace          linen            AntiqueWhite     PapayaWhip       BlanchedAlmond
bisque           PeachPuff        NavajoWhite      moccasin         cornsilk
ivory            LemonChiffon     seashell         honeydew         MintCream
azure            AliceBlue        lavender         LavenderBlush    MistyRose
white            black            DarkSlateGrey    DimGrey          SlateGrey
LightSlateGrey   grey             LightGrey        MidnightBlue     navy
NavyBlue         CornflowerBlue   DarkSlateBlue    SlateBlue        MediumSlateBlue
LightSlateBlue   MediumBlue       RoyalBlue        blue             DodgerBlue
DeepSkyBlue      SkyBlue          LightSkyBlue     SteelBlue        LightSteelBlue
LightBlue        PowderBlue       PaleTurquoise    DarkTurquoise    MediumTurquoise
turquoise        cyan             LightCyan        CadetBlue        MediumAquamarine
aquamarine       DarkGreen        DarkOliveGreen   DarkSeaGreen     SeaGreen
MediumSeaGreen   LightSeaGreen    PaleGreen        SpringGreen      LawnGreen
green            chartreuse       MediumSpringGreen       GreenYellow     LimeGreen
YellowGreen      ForestGreen      OliveDrab        DarkKhaki        khaki
PaleGoldenrod    LightGoldenrodYellow    LightYellow     yellow    gold
LightGoldenrod   goldenrod        DarkGoldenrod    RosyBrown        IndianRed
SaddleBrown      sienna           peru             burlywood        beige
wheat            SandyBrown       tan              chocolate        firebrick
brown            DarkSalmon       salmon           LightSalmon      orange
DarkOrange       coral            LightCoral       tomato           OrangeRed
red              HotPink          DeepPink         pink             LightPink
PaleVioletRed    maroon           MediumVioletRed  VioletRed        magenta
violet           plum             orchid           MediumOrchid     DarkOrchid
DarkViolet       BlueViolet       purple           MediumPurple     thistle
DarkGrey         DarkBlue         DarkCyan         DarkMagenta      DarkRed
LightGreen
```

Color names with a numerical suffix

In the following names the suffix N can be a number in the range 1-4:

snowN	seashellN	AntiqueWhiteN	bisqueN	PeachPuffN
NavajoWhiteN	LemonChiffonN	cornsilkN	ivoryN	honeydewN
LavenderBlushN	MistyRoseN	azureN	SlateBlueN	RoyalBlueN
blueN	DodgerBlueN	SteelBlueN	DeepSkyBlueN	SkyBlueN
LightSkyBlueN	LightSteelBlueN	LightBlueN	LightCyanN	PaleTurquoiseN
CadetBlueN	turquoiseN	cyanN	aquamarineN	DarkSeaGreenN
SeaGreenN	PaleGreenN	SpringGreenN	greenN	chartreuseN
OliveDrabN	DarkOliveGreenN	khakiN	LightGoldenrodN	LightYellowN
yellowN	goldN	goldenrodN	DarkGoldenrodN	RosyBrownN
IndianRedN	siennaN	burlywoodN	wheatN	tanN
chocolateN	firebrickN	brownN	salmonN	LightSalmonN
orangeN	DarkOrangeN	coralN	tomatoN	OrangeRedN
redN	DeepPinkN	HotPinkN	pinkN	LightPinkN
PaleVioletRedN	maroonN	VioletRedN	magentaN	orchidN
plumN	MediumOrchidN	DarkOrchidN	purpleN	MediumPurpleN
thistleN				

Grey Scale

A grey scale can be obtained using:

`greyN`

Where N is in the range 0-100.

A.6 The Feta font

The following symbols are available in the Emmentaler font and may be accessed directly using text markup with the name of the glyph as shown in the tables below, such as `g^\markup {\musicglyph #"scripts.segno" }` or `\markup {\musicglyph #"five"}`. For more information, see ⟨undefined⟩ [Formatting text], page ⟨undefined⟩.

Clef glyphs

clefs.C		clefs.C_change	
clefs.F		clefs.F_change	
clefs.G		clefs.G_change	
clefs.percussion		clefs.percussion_change	
clefs.tab		clefs.tab_change	

Time Signature glyphs

timesig.C44		timesig.C22	

Number glyphs

plus	+	comma	,
hyphen	-	period	.
zero	0	one	1
two	2	three	3
four	4	five	5
six	6	seven	7
eight	8	nine	9

Accidental glyphs

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 .slashslash.stemstemstem	♯
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.doublesharp	
accidentals.rightparen		accidentals.leftparen	

Default Notehead glyphs

noteheads.uM2		noteheads.dM2	
noteheads.sM1		noteheads.s0	
noteheads.s1		noteheads.s2	

Special Notehead glyphs

noteheads.sM1double		noteheads.s0diamond	
noteheads.s1diamond		noteheads.s2diamond	
noteheads.s0triangle		noteheads.d1triangle	
noteheads.u1triangle		noteheads.u2triangle	
noteheads.d2triangle		noteheads.s0slash	

noteheads.s1slash		noteheads.s2slash	
noteheads.s0cross		noteheads.s1cross	
noteheads.s2cross		noteheads.s2xcircle	
noteheads.s0harmonic		noteheads.s2harmonic	

Shape-note Notehead glyphs

noteheads.s0do		noteheads.d1do	
noteheads.u1do		noteheads.d2do	
noteheads.u2do		noteheads.s0doThin	
noteheads.d1doThin		noteheads.u1doThin	
noteheads.d2doThin		noteheads.u2doThin	
noteheads.s0re		noteheads.u1re	
noteheads.d1re		noteheads.u2re	
noteheads.d2re		noteheads.s0reThin	
noteheads.u1reThin		noteheads.d1reThin	
noteheads.u2reThin		noteheads.d2reThin	
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.u1ti	

noteheads.d1ti		noteheads.u2ti	
noteheads.d2ti		noteheads.s0tiThin	
noteheads.u1tiThin		noteheads.d1tiThin	
noteheads.u2tiThin		noteheads.d2tiThin	
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			

Rest glyphs

rests.0		rests.1	
rests.0o		rests.1o	
rests.M3		rests.M2	
rests.M1		rests.2	
rests.2classical		rests.3	
rests.4		rests.5	
rests.6		rests.7	

Flag glyphs

flags.u3		flags.u4	
flags.u5		flags.u6	
flags.u7		flags.d3	
flags.ugrace		flags.dgrace	
flags.d4		flags.d5	
flags.d6		flags.d7	

Dot glyphs

dots.dot	

Dynamic glyphs

space		f	
m		p	
r		s	
z			

Script glyphs

scripts.ufermata		scripts.dfermata	
scripts.ushortfermata		scripts.dshortfermata	
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.stopped	
scripts.upbow		scripts.downbow	
scripts.reverseturn		scripts.turn	
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.trilelement		scripts.prall	
scripts.mordent		scripts.prallprall	
scripts.prallmordent		scripts.upprall	
scripts.upmordent		scripts.pralldown	
scripts.downprall		scripts.downmordent	
scripts.prallup		scripts.lineprall	
scripts.caesura.curved		scripts.caesura.straight	

scripts.snappizzicato		scripts.ictus	
scripts.uaccentus		scripts.daccentus	
scripts.usemicirculus		scripts.dsemicirculus	
scripts.circulus		scripts.augmentum	
scripts .usignumcongruentiae		scripts .dsignumcongruentiae	

Arrowhead glyphs

arrowheads.open.01		arrowheads.open.0M1	
arrowheads.open.11		arrowheads.open.1M1	
arrowheads.close.01		arrowheads.close.0M1	
arrowheads.close.11		arrowheads.close.1M1	

Bracket-tip glyphs

brackettips.up		brackettips.down	

Pedal glyphs

pedal.*		pedal.M	
pedal..		pedal.P	
pedal.d		pedal.e	
pedal.Ped			

Accordion glyphs

accordion.discant		accordion.dot	
accordion.freebass		accordion.stdbass	
accordion.bayanbass		accordion.oldEE	
accordion.push		accordion.pull	

Vaticana glyphs

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	

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	.		

Medicaea glyphs

noteheads.smedicaea .inclinatum		noteheads .smedicaea.punctum	
noteheads .smedicaea.rvirga		noteheads .smedicaea.virga	
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			

Hufnagel glyphs

noteheads .shufnagel.punctum

noteheads .shufnagel.virga



noteheads.shufnagel.lpes

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



Mensural glyphs

rests.M3mensural

rests.M2mensural

rests.M1mensural

rests.0mensural

rests.1mensural

rests.2mensural

rests.3mensural

rests.4mensural

noteheads.slmensural



noteheads.sM3mensural



noteheads.sM2mensural



noteheads.sM1mensural



noteheads.s0mensural		noteheads.s1mensural	
noteheads.s2mensural		clefs.mensural.c	
clefs.mensural.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	

Neomensural glyphs

rests.M3neomensural		rests.M2neomensural	
rests.M1neomensural		rests.0neomensural	
rests.1neomensural		rests.2neomensural	

rests.3neomensural		rests.4neomensural	
noteheads.slneomensural		noteheads.sM3neomensural	
noteheads.sM2neomensural		noteheads.sM1neomensural	
noteheads.s0neomensural		noteheads.s1neomensural	
noteheads.s2neomensural		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	

Petrucci glyphs

noteheads.s0petrucci		noteheads.s1petrucci	
noteheads.s2petrucci		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

Solesmes glyphs

noteheads.ssolesmes
.incl.parvum

noteheads
.ssolesmes.auct.asc

noteheads
.ssolesmes.auct.desc

noteheads.ssolesmes
.incl.auctum

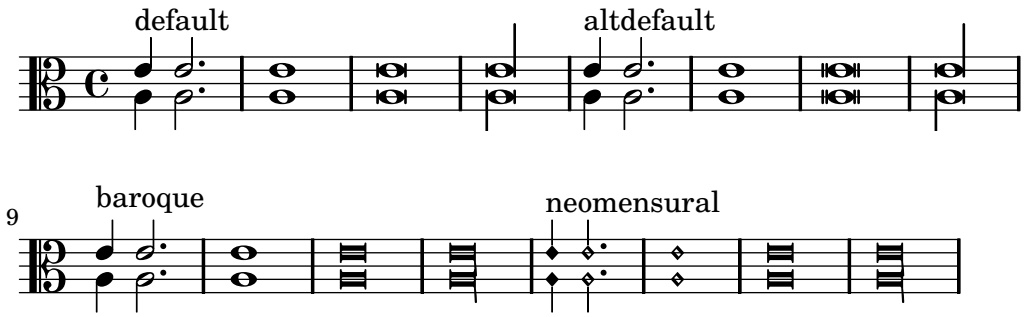
noteheads
.ssolesmes.stropha

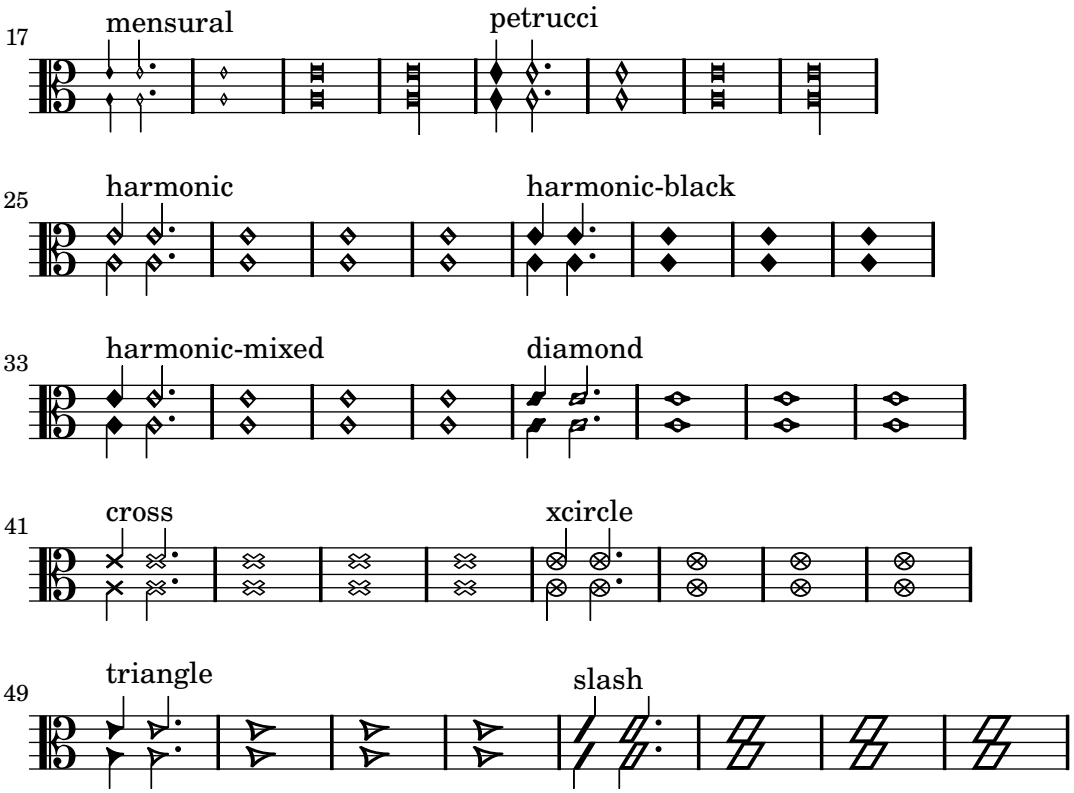
noteheads.ssolesmes
.stropha.aucta

noteheads
.ssolesmes.oriscus

A.7 Note head styles

The following styles may be used for note heads.





A.8 Text markup commands

The following commands can all be used inside `\markup { }`.

A.8.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)

Use *size* as the absolute font size 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 }
}
```

default text font size text font size 16 text font size 12

`\bold` *arg* (markup)

Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

default **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:

- `baseline-skip`

`\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:

- `baseline-skip`

`\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 {
    Some text with font overrides
    \hspace #2
    \normal-text {
      Default text, same font-size
    }
    \hspace #2
    More text as before
  }
}
```

SOME TEXT WITH FONT OVERRIDES Default text, same font-size **MORE**

`\normalsize` *arg* (markup)
Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small normal size 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 . ,
  }
}
```

0123456789.,

`\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 serif, bold text in roman font family return to sans

`\sans` *arg* (markup)
Switch to the sans serif font family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```

default 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 {
    some large text
    \hspace #2
    \smaller {
      a bit smaller
    }
    \hspace #2
    more large text
  }
}
```

some large text a bit smaller more large text

\sub *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

H_2O

Used properties:

- `baseline-skip`
- `font-size` (0)

`\super` *arg* (markup)
Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

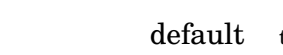


Used properties:

- `baseline-skip`
- `font-size` (0)

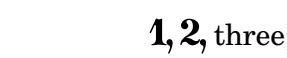
`\teeny` *arg* (markup)
Set font size to -3.

```
\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
  }
}
```



`\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, and `offset` to determine line y-offset.

```
\markup \fill-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
}
```

underlined underlined underlined

Used properties:

- `offset` (2)
- `thickness` (1)

`\upright` *arg* (markup)
Set `font-shape` to `upright`. This is the opposite of `italic`.

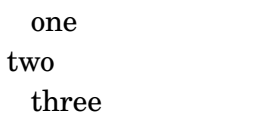
```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```

italic text upright text *italic again*

A.8.2 Align

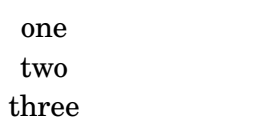
`\center-align` *arg* (markup)
Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```



`\center-column` *args* (markup list)
Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

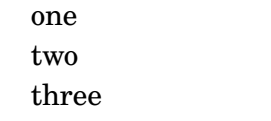


Used properties:

- `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
  }
}
```



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; the follow example will not compile:

```
\combine { a list }
```

```
\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 }
    }
  }
}
```

Words evenly spaced across the page

Text markups *evenly spaced* across the page

Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\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
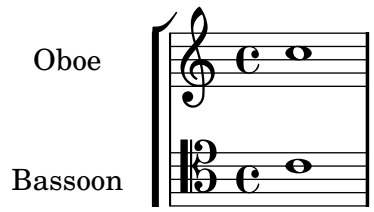
one
two
three

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

Used properties:

- `word-space`

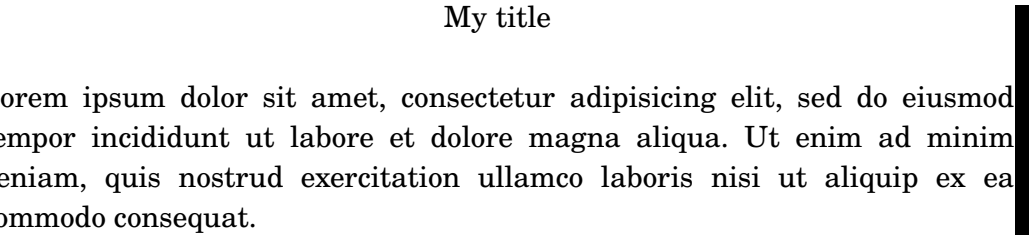
`\justify-field` *symbol* (symbol)

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
    }
  }
}

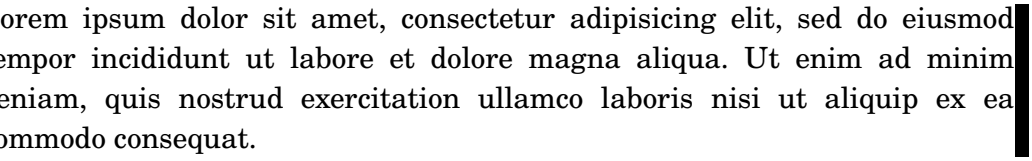
\markup {
  \null
}
```



`\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.
  }
}
```



Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

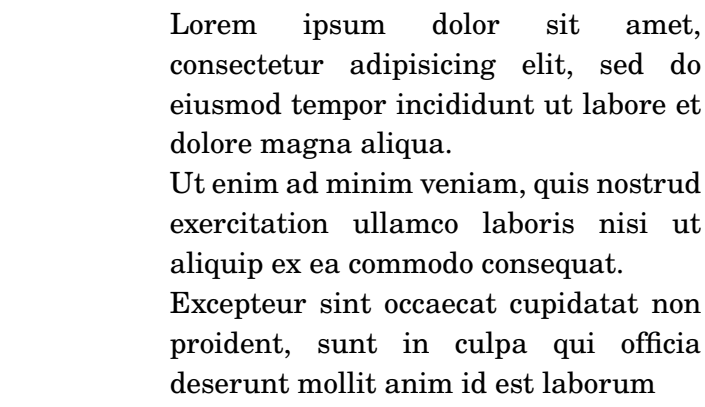
`\justify-string` *arg* (string)

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"
}
```



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`.

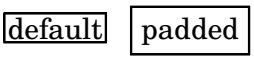
```
\markup {
  one
  \lower #3
  two
  three
}
```

one three
two

`\pad-around` *amount* (number) *arg* (markup)
: Add padding *amount* all around *arg*.

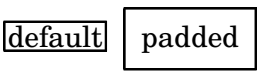
```
\markup {
  \box {
    default
```

```
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```



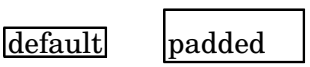
`\pad-markup` *amount* (number) *arg* (markup)
: Add space around a markup object.

```
\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.

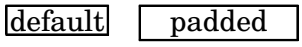
```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



`\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`.

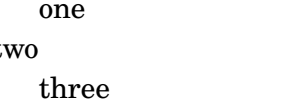
The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

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
  }
}
```

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 }
}
```

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
  }
}
```

* translate * translate-scaled

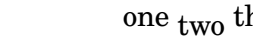
Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)

Align `arg` to its Y center.

```
\markup {
  one
  \vcenter
  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
  }
}
```


`\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

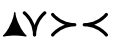
Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

A.8.3 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-line` *dest* (pair of numbers)
: A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
```

```
}
```



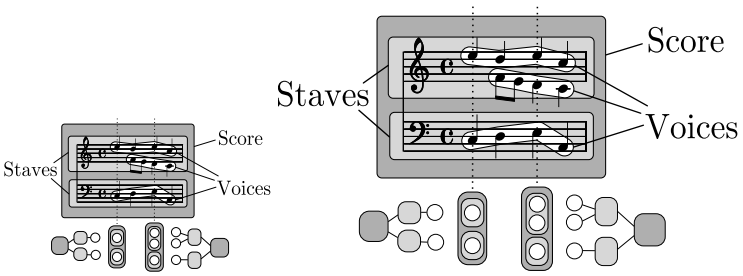
Used properties:

- `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
}
```

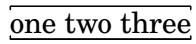


`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
```

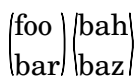
```
}
```



`\parenthesize` *arg* (markup)

Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \line {
    \parenthesize {
      \column {
        foo
        bar
      }
    }
    \override #'(angularity . 2) {
      \parenthesize {
        \column {
          bah
          baz
        }
      }
    }
  }
}
```



Used properties:

- `width` (0.25)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line thickness *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.

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.

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.

```
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
}
```



Used properties:

- `filled` (#f)
- `line-join-style` (round)
- `line-cap-style` (round)
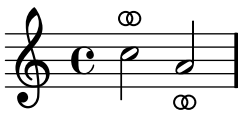
`\postscript` *str* (string)

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
}
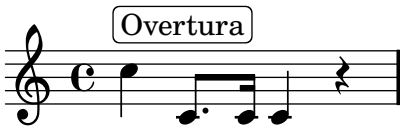
\relative c'' {
  c2^\rings
  a2_\rings
}
```



`\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
  }
}
```



`\triangle` *filled* (boolean)

A triangle, either filled or empty.

```
\markup {
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```



Used properties:

- `baseline-skip` (2)
- `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 #"http://lilypond.org/web/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

A.8.4 Music

`\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
}
```


`\flat`
: Draw a flat symbol.

```
\markup {
  \flat
}
```


`\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 Feta 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. By using fractional values for *dir*, longer or shorter stems can be obtained.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- `style` ('())
- `font-size` (0)

`\note` *duration* (string) *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:

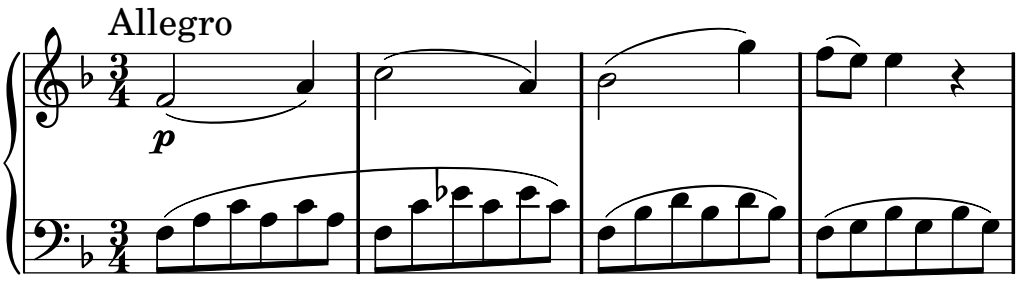
- `style` ('())
- `font-size` (0)

`\score` *score* (score)

Inline an image of music.

```
\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 {
  \tied-lyric #"Lasciate~i monti"
}
```

Lasciate‿i monti

A.8.5 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 (in staff spaces). Default: 0.05.
- `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*`)`
: Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*. By default, the fret playing indicator is a solid dot. This can be changed by setting the value of the variable *dot-color*. 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.

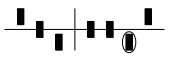
Possible elements in *definition-string*:

`^`	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. ie...

```
\markup \woodwind-diagram #'oboe #'()
```

Used properties:

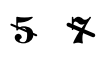
- `graphical` (#t)
- `thickness` (0.1)
- `size` (1)

A.8.6 Other

`\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`.

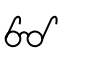
```
\markup {
  \char #65 \char ##x00a9
}
```



`\eyeglasses`

Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```

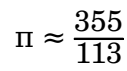


`\fraction` *arg1* (markup) *arg2* (markup)

Make a fraction of two markups.

```
\markup {

  \fraction 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
}
```

I AA

`\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
}
```

J AB

`\null`
: An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (symbol) *arg* (markup)
: Apply the *procedure* markup command to *arg*. *procedure* should take a single argument.

`\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*.

```
\markup {
  \line {
    \column {
      default
      baseline-skip
    }
    \hspace #2
    \override #'(baseline-skip . 4) {
      \column {
        increased
        baseline-skip
      }
    }
  }
}
```

default	increased
baseline-skip	baseline-skip

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)
: Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

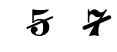
`\right-brace` *size* (number)
: A feta brace in point size *size*, rotated 180 degrees.

```
\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
  }
}
```

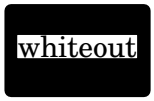
`\verbatim-file` *name* (string)
: Read the contents of file *name*, and include it verbatim.

```
\markup {
  \verbatim-file #"simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative c' {
  c d e f g a b c
}
%% Optional helper for automatic updating by convert-ly.  May be omit
\version "2.12.0"
```

`\whiteout` *arg* (markup)
: Provide a white background for *arg*.

```
\markup {
  \combine
    \filled-box #'(-1 . 10) #'(-3 . 4) #1
    \whiteout whiteout
}
```



`\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 #blue
  blue
}
```

red green blue

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
: Set the dimensions of *arg* to *x* and *y*.

A.9 Text markup list commands

The following commands can all be used with `\markuplines`:

`\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`

`\override-lines` *new-prop* (pair) *args* (markup list)
: Like `\override`, for markup lists.

`\table-of-contents`

`\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 used to define `\justify-string` and `\wordwrap-string`.

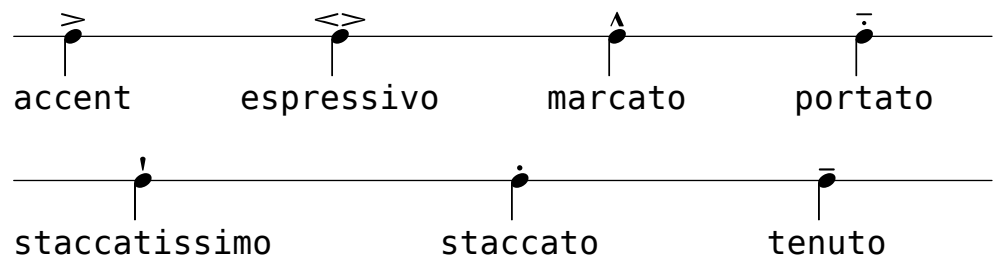
  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width`

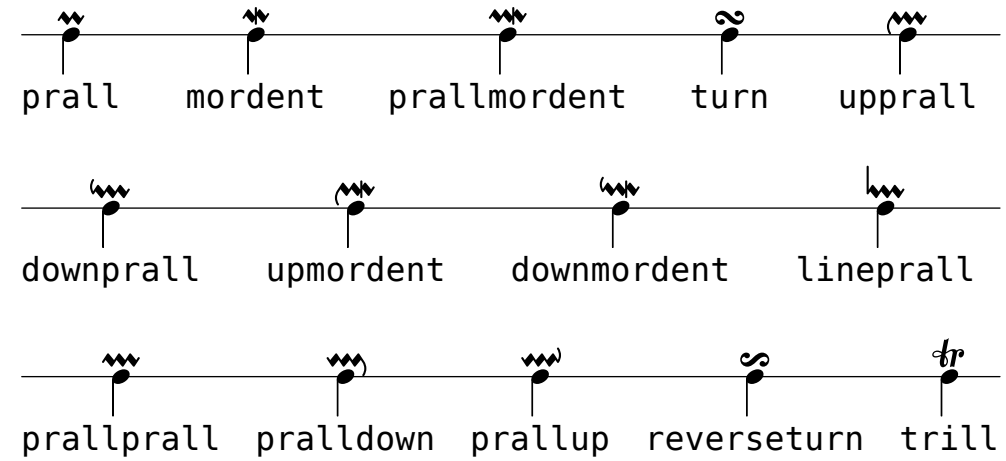
A.10 List of articulations

The following scripts are available in the Feta font and may be attached to notes (eg. '`c\accent`').

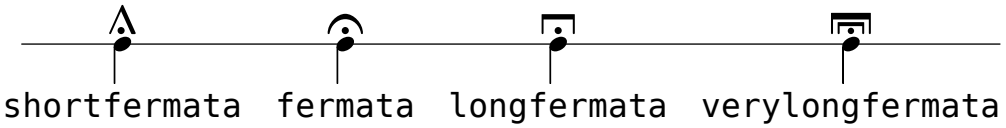
Articulation scripts



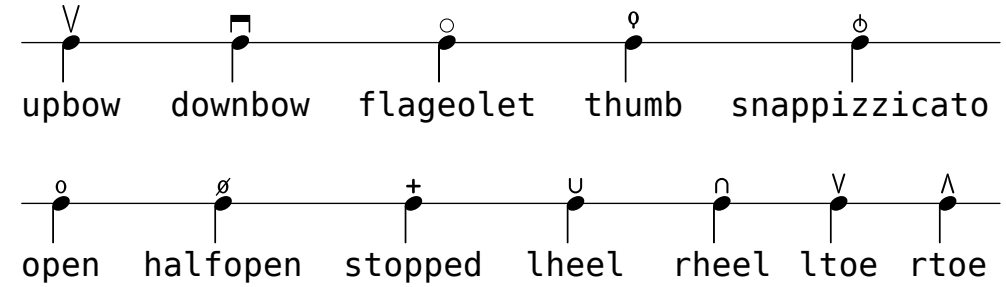
Ornament scripts



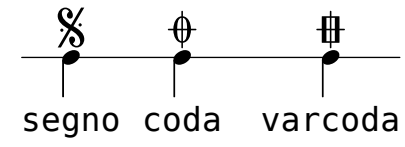
Fermata scripts



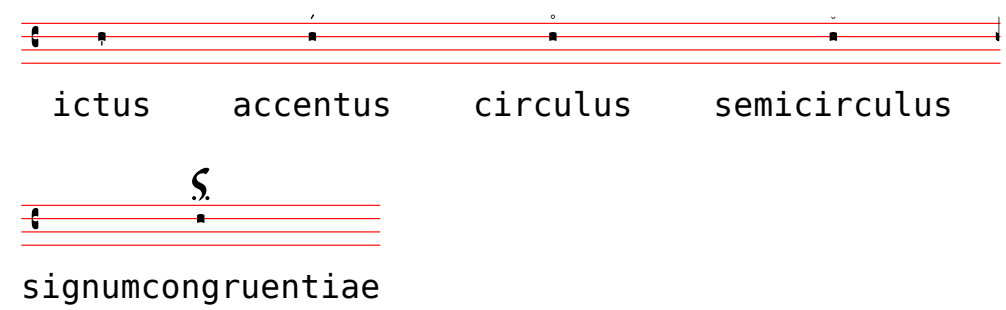
Instrument-specific scripts



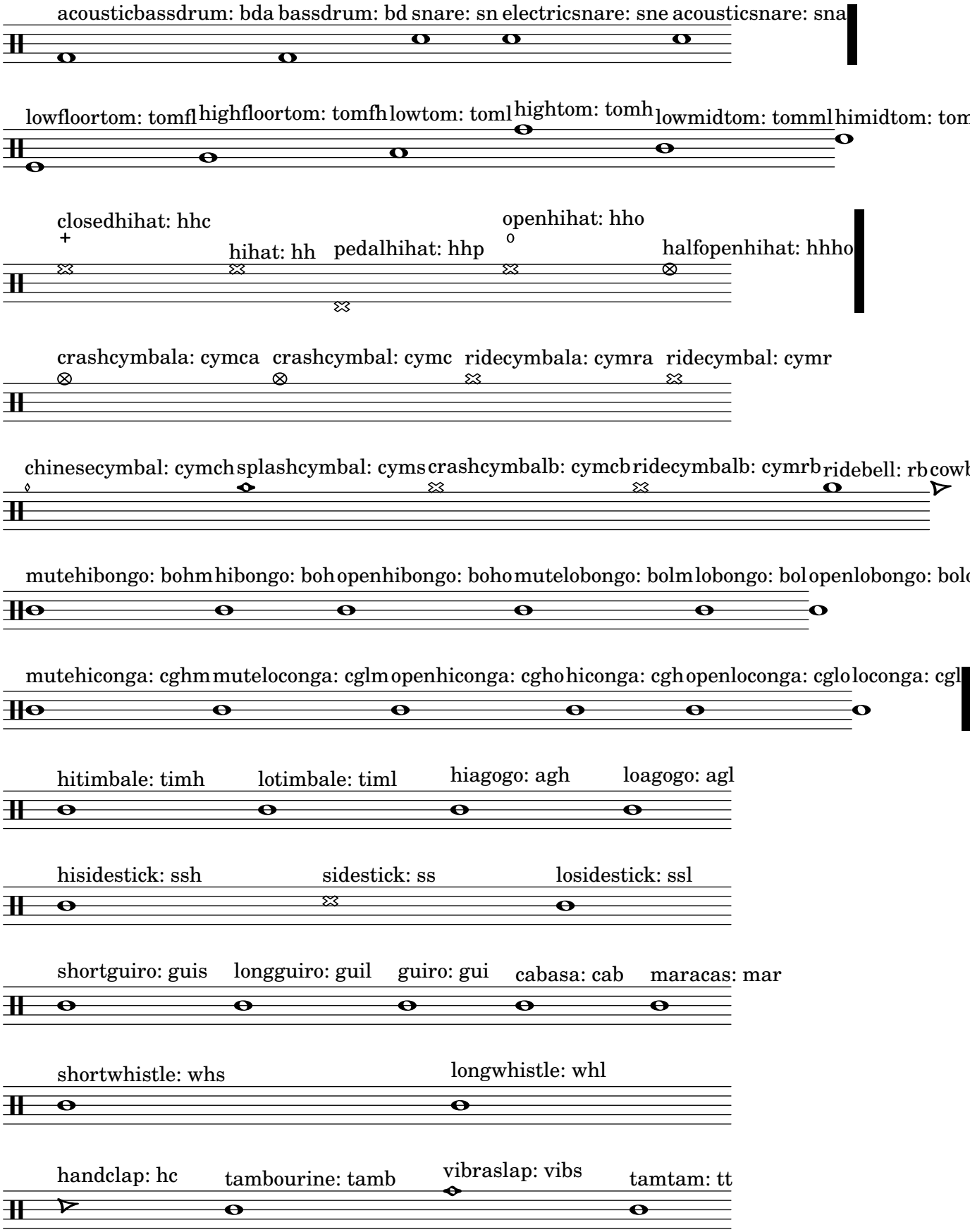
Repeat sign scripts

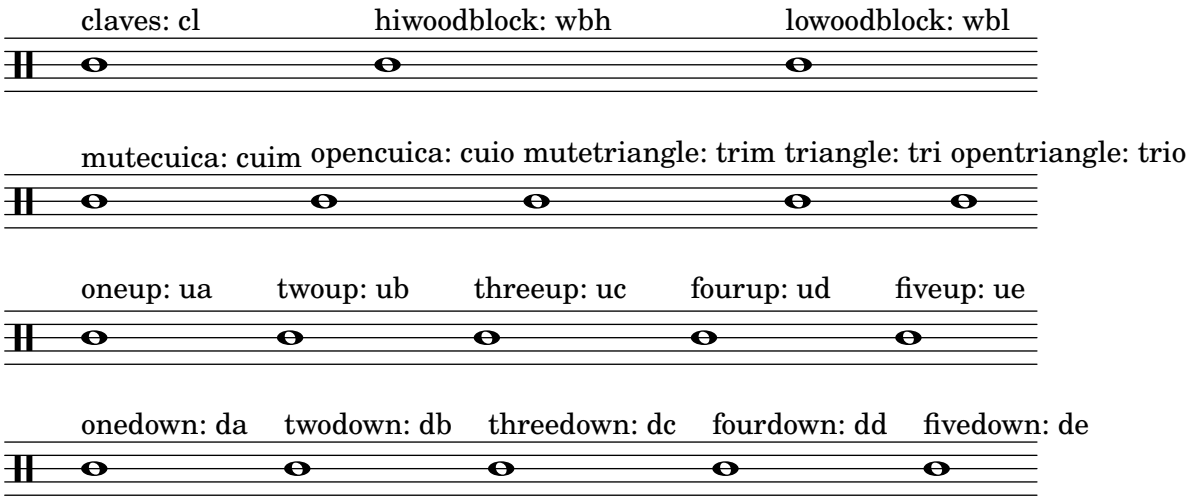


Ancient scripts



A.11 Percussion notes





A.12 Technical glossary

A glossary of the technical terms and concepts used internally in LilyPond. These terms may appear in the manuals, on mailing lists or in the source code.

alist

An association list or **alist** for short is a Scheme pair which associates a value with a key: `(key . value)`. For example, in '`scm/lily.scm`', the alist « type-p-name-alist » associates certain type predicates (e.g. `ly:music?`) with names (e.g. « music ») so that type-check failures can be reported with a console message that includes the name of the expected type predicate.

callback

A **callback** is a routine, function or method whose reference is passed as an argument in a call to another routine, so allowing the called routine to invoke it. The technique enables a lower-level software layer to call a function defined in a higher layer. Callbacks are used extensively in LilyPond to permit user-level Scheme code to define how many low-level actions are performed.

closure

In Scheme, a **closure** is created when a function, usually a lambda expression, is passed as a variable. The closure contains the function's code plus references to the lexical bindings of the function's free variables (i.e. those variables used in the expression but defined outside it). When this function is applied to different arguments later, the free variable bindings that were captured in the closure are used to obtain the values of the free variables to be used in the calculation. One useful property of closures is the retention of internal variable values between invocations, so permitting state to be maintained.

A **simple closure** is a closure whose expression has no free variables and hence no free variable bindings.

A simple closure is represented in LilyPond by a smob containing the expression and a method to apply the expression to a passed list of arguments.

glyph

A **glyph** is a particular graphical representation of a typographic character, or a combination of two characters formating a ligature. A set of glyphs with a single style and shape comprise a font, and a set of fonts covering several styles and sizes comprise a typeface.

Voir aussi

Notation Reference: ⟨undefined⟩ [Fonts], page ⟨undefined⟩, Section 3.3.3 [Text encoding], page 376.

grob

LilyPond objects which represent items of notation in the printed output such as note heads, stems, slurs, ties, fingering, clefs, etc are called 'Layout objects', often known as 'GRaphical OBjects', or **grobs** for short. They are represented by instances of the `Grob` class.

Voir aussi

Learning Manual: Section "Objects and interfaces" dans *Manuel d'initiation*, Section "Naming conventions of objects and properties" dans *Manuel d'initiation*, Section "Properties of layout objects" dans *Manuel d'initiation*.

Internals Reference: Section "grob-interface" dans *Référence des propriétés internes*, Section "All layout objects" dans *Référence des propriétés internes*.

immutable

An **immutable** object is one whose state cannot be modified after creation, in contrast to a mutable object, which can be modified after creation.

In LilyPond, immutable or shared properties define the default style and behavior of grobs. They are shared between many objects. In apparent contradiction to the name, they can be changed using `\override` and `\revert`.

Voir aussi

Notation Reference: [mutable], page 559.

interface

Actions and properties which are common to a number of grobs are grouped together in an object called a `grob-interface`, or just 'interface' for short.

Voir aussi

Learning Manual: Section "Objects and interfaces" dans *Manuel d'initiation*, Section "Naming conventions of objects and properties" dans *Manuel d'initiation*, Section "Properties found in interfaces" dans *Manuel d'initiation*.

Notation Reference: Section 5.2.2 [Layout interfaces], page 447.

Internals Reference: Section "Graphical Object Interfaces" dans *Référence des propriétés internes*.

lexer

A **lexer** is a program which converts a sequence of characters into a sequence of tokens, a process called lexical analysis. The LilyPond lexer converts the stream obtained from an input '`.ly`' file into a tokenized stream more suited to the next stage of processing - parsing, for which see [parser], page 560. The LilyPond lexer is built with Flex from the lexer file '`lily/lexer.ll`' which contains the lexical rules. This file is part of the source code and is not included in the LilyPond binary installation.

mutable

A **mutable** object is one whose state can be modified after creation, in contrast to an immutable object, whose state is fixed at the time of creation.

In LilyPond, mutable properties contain values that are specific to one grob. Typically, lists of other objects or results from computations are stored in mutable properties.

Voir aussi

Notation Reference: [immutable], page 559.

output-def

An instance of the `Output-def` class contains the methods and data structures associated with an output block. Instances are created for midi, layout and paper blocks.

parser

A **parser** analyzes the sequence of tokens produced by a lexer to determine its grammatical structure, grouping the tokens progressively into larger groupings according to the rules of the grammar. If the sequence of tokens is valid the end product is a tree of tokens whose root is the grammar's start symbol. If this cannot be achieved the file is invalid and an appropriate error message is produced. The syntactic groupings and the rules for constructing the groupings from their parts for the LilyPond syntax are defined in '`lily/parser.yy`' and shown in Backus Normal Form (BNF) in ⟨undefined⟩ [LilyPond grammar], page ⟨undefined⟩. This file is used to build the parser during the program build by the parser generator, Bison. It is part of the source code and is not included in the LilyPond binary installation.

parser variable

These are variables defined directly in Scheme. Their direct use by users is strongly discouraged, because their scoping semantics can be confusing.

When the value of such a variable is changed in a `.ly` file, the change is global, and unless explicitly reverted, the new value will persist to the end of the file, affecting subsequent `\score` blocks as well as external files added with the `\include` command. This can lead to unintended consequences and in complex typesetting projects the consequent errors can be difficult to track down.

LilyPond uses the following parser variables:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- parseStringResult
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

prob

PRoperty OBjects, or **probs** for short, are instances of the `Prob` class, a simple base class for objects which have mutable and immutable property alists and the methods to manipulate them. The `Music` and `Stream_event` classes derive from `Prob`. Instances of the `Prob` class are also created to hold the formatted content of system grobs and titling blocks during page layout.

simple closure

See [closure], page 558.

smob

Smobs, or ScheMe OBjects, are part of the mechanism used by Guile to export C and C++ objects to Scheme code. In LilyPond, smobs are created from C++ objects through macros. There are two types of smob objects: simple smobs, intended for simple immutable objects like numbers, and complex smobs, used for objects with identities. If you have access to the LilyPond sources, more information can be found in '`lily/includes/smob.hh`'.

stencil

An instance of the **stencil** class holds the information required to print a typographical object. It is a simple smob containing a confining box, which defines the vertical and horizontal extents of the object, and a Scheme expression which will print the object when evaluated. Stencils may be combined to form more complex stencils defined by a tree of Scheme expressions formed from the Scheme expressions of the component stencils.

The `stencil` property, which connects a grob to its stencil, is defined in the `grob-interface` interface.

Voir aussi

Internals Reference: Section "grob-interface" dans *Référence des propriétés internes*.

A.13 All context properties

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBassFigureAccidentals` (boolean)
: If true, then the accidentals are aligned in bass figure context.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`associatedVoice` (string)
: Name of the `Voice` 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.

`barNumberVisibility` (procedure)
A Procedure that takes an integer and returns whether the corresponding bar number should be printed.

`baseMoment` (moment)
Smallest unit of time that will stand on its own as a subdivided section.

`bassFigureFormatFunction` (procedure)
A procedure that is called to produce the formatting for a `BassFigure` grob. It takes a list of `BassFigureEvent`s, a context, and the grob to format.

`bassStaffProperties` (list)
An alist of property settings to apply for the down staff of `PianoStaff`. Used by `\autochange`.

`beamExceptions` (list)
An alist of exceptions to autobeam rules that normally end on beats.

`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.

`chordNameExceptionsFull` (list)
An alist of full chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameExceptionsPartial` (list)
: An alist of partial chord exceptions. Contains (*chord* . (*prefix-markup suffix-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.

`clefOctavation` (integer)
: Add this much extra octavation. Values of 7 and -7 are common.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`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.`'.

`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.

`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`', '`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.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for 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 changing from a non-natural to another non-natural.

`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.

`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.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`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.

`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.

`implicitTimeSignatureVisibility` (vector)
: break visibility for the default time signature.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Section "Lyrics" dans *Référence des propriétés internes*.

`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 like 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).

`keySignature` (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. `keySignature = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (direction)
: Alignment to use for a melisma syllable.

`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.

`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 four arguments: text, duration, count 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`.

`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`.

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`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.

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`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.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`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.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printOctaveNames` (boolean)
: Print octave marks for 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.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`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.

`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.

`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`.

`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 pitch (in semitones) of each string (starting with the lower 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)
: If set, accidentals are typeset as cautionary suggestions over the note.

`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.

`tempoText` (markup)
: Text for tempo marks.

`tempoUnitCount` (number)
: Count for specifying tempo.

`tempoUnitDuration` (duration)
: Unit for specifying tempo.

`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` (pair of numbers)
: 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*

`trebleStaffProperties` (list)
: An alist of property settings to apply for the up staff of `PianoStaff`. Used by `\autochange`.

`tremoloFlags` (integer)
: The number of tremolo flags to add if no number is specified.

`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 }
}
```

`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 Section "bar-line-interface" dans *Référence des propriétés internes*.

A.14 Layout properties

`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` (string)
: Annotate a grob for debug purposes.

`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 that is larger than this number, make a kneed beam.

`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-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-size` (dimension, in staff space)
: The size of a 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.

`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 break-align grob 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)
: Defines the order in which prefatory matter (clefs, key signatures) appears. The format is a vector of length 3, where each element is one order for end-of-line, middle of line, and start-of-line, respectively. An order is a list of symbols.

  For example, clefs are put after key signatures by setting

```
\override Score.BreakAlignment #'break-align-orders =
  #(make-vector 3 '(span-bar
                    breathing-sign
                    staff-bar
                    key
                    clef
                    time-signature))
```

`break-align-symbol` (symbol)
: This key is used for aligning and spacing breakable items.

`break-align-symbols` (list)
: A list of symbols that determine which break-aligned grobs 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 `left-edge`, `ambitus`, `breathing-sign`, `clef`, `staff-bar`, `key-cancellation`, `key-signature`, `time-signature`, and `custos`.

`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.

`c0-position` (integer)
: An integer indicating the position of middle C.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`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.

`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)
: 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.

`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 `0.0` (no line) and `1.0` (continuous line).

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`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.

`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`*`)`.

`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)`.

`extra-X-extent` (pair of numbers)
: A grob is enlarged in X dimension by this much.

`extra-Y-extent` (pair of numbers)
: A grob is enlarged in Y dimension by this much.

`flag` (stencil)
: A function returning the full flag stencil for the `Stem`, which is passed to the function as the only argument. The default ly:stem::calc-stencil function uses the `flag-style` property to determine the correct glyph for the flag. By providing your own function, you can create arbitrary flags.

`flag-count` (number)
: The number of tremolo beams.

`flag-style` (symbol)
: A symbol determining what style of flag glyph is typeset on a `Stem`. Valid options include `'()` for standard flags, `'mensural` and `'no-flag`, which switches off the flag.

`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-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. Fractional values are allowed.

`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*.

`fraction` (pair of numbers)
: 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-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.
  - `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`, and `arabic`. 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-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.

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

`glyph-name` (string)
: The glyph name within the font.

`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.

`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.

`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.

`ignore-collision` (boolean)
: If set, don’t do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-index` (integer)
: If debugging is set, set beam and slur configuration to this index, and print the respective scores.

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur quants to this position, and print the respective scores.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: Amount of extra white space to add. For bar lines, this is the amount of space after a thick line.

`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.

`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-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.

`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.

`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)
: The thickness of the tie or slur contour.

`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-stretch` (number)
: The maximum amount that this `VerticalAxisGroup` can be vertically stretched (for example, in order to better fill a page).

`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-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.

`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.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`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 this grob and the staff when spacing according to `outside-staff-priority`.

`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-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`.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates `(`*`left . right`*`)`, where both *left* and *right* are 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.

`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?

`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.

`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 sorting key that determines in what order a script is within a stack of scripts.

`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.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`shorten-pair` (pair of numbers)
: The lengths to shorten a text-spanner on both sides, for example a pedal bracket. Positive values shorten the text-spanner, while negative values lengthen it.

`shortest-duration-space` (dimension, in staff space)
: Start with this much space for the shortest duration. This is expressed in `spacing-increment` as unit. 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.

`size` (number)
: Size of object, relative to standard size.

`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.

`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.

`space-alist` (list)
: A table that specifies distances between prefatory items, like clef and time-signature. The format is an alist of spacing tuples: `(`*`break-align-symbol type . distance`*`)`, where *type* can be the symbols `minimum-space` or `extra-space`.

`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` (number)
: Add this much space for a doubled duration. 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)
  ```

`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:

  - `padding` – the minimum required amount of unobstructed vertical whitespace between the two items, measured in staff-spaces.
  - `space` – 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.
  - `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-end-position` (number)
: Where does the stem end (the end is opposite to the support-head)?

`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)
: Bar line thickness, measured in `line-thickness`.

`thickness` (number)
: Line thickness, generally measured in `line-thickness`.

`thin-kern` (number)
: The space after a hair-line in a bar line.

`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 keep the default position (centered on the note head), `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their durations. This looks better in complex polyphonic patterns.

`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.

`when` (moment)
: Global time step associated with this column happen?

`whiteout` (boolean)
: If true, the grob is printed over a white background to white-out underlying material, if the grob is visible. Usually #f by default.

`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-extent` (pair of numbers)
: Hard coded extent in X direction.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`Y-extent` (pair of numbers)
: Hard coded extent in Y direction.

`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 glissando line can be constructed from a whole number of squiggles.

A.15 Available music functions

`acciaccatura` - *music* (music)
: Create an acciaccatura from the following music expression

`addChordShape` - *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` - *name* (string) *lst* (list)
: Create instrument *name* with properties *list*.

`addQuote` - *name* (string) *music* (music)
: Define *music* as a quotable music expression named *name*

`afterGrace` - *main* (music) *grace* (music)
: Create *grace* note(s) after a *main* music expression.

`allowPageTurn`
: Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

`applyContext` - *proc* (procedure)
: Modify context properties with Scheme procedure *proc*.

`applyMusic` - *func* (procedure) *music* (music)
: Apply procedure *func* to *music*.

`applyOutput` - *ctx* (symbol) *proc* (procedure)
: Apply function `proc` to every layout object in context `ctx`

`appoggiatura` - *music* (music)
: Create an appoggiatura from *music*

`assertBeamQuant` - *l* (pair) *r* (pair)
: Testing function: check whether the beam quants *l* and *r* are correct

`assertBeamSlope` - *comp* (procedure)
: Testing function: check whether the slope of the beam is the same as `comp`

`autochange` - *music* (music)
: Make voices that switch between staves automatically

`balloonGrobText` - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
: Attach *text* to *grob-name* at offset *offset* (use like `\once`)

`balloonText` - *offset* (pair of numbers) *text* (markup)
: Attach *text* at *offset* (use like `\tweak`)

`bar` - *type* (string)
: Insert a bar line of type *type*

`barNumberCheck` - *n* (integer)
: Print a warning if the current bar number is not *n*.

`bendAfter` - *delta* (real number)
: Create a fall or doit of pitch interval *delta*.

`bookOutputName` - *newfilename* (string)
: Direct output for the current book block to *newfilename*.

`bookOutputSuffix` - *newsuffix* (string)
: Set the output filename suffix for the current book block to *newsuffix*.

`breathe` Insert a breath mark.

`clef` - *type* (string)
Set the current clef to *type*.

`cueDuring` - *what* (string) *dir* (direction) *main-music* (music)
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`deadNote` - *note* (music)
Print *note* with a cross-shaped note head.

`defaultNoteHeads`
Revert to the default note head style.

`displayLilyMusic` - *music* (music)
Display the LilyPond input representation of *music* to the console.

`displayMusic` - *music* (music)
Display the internal representation of *music* to the console.

`endSpanners` - *music* (music)
Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

`featherDurations` - *factor* (moment) *argument* (music)
Adjust durations of music in *argument* by rational *factor*.

`grace` - *music* (music)
Insert *music* as grace notes.

`harmonicNote` - *note* (music)
Print *note* with a diamond-shaped note head.

`harmonicsOn`
Set the default note head style to a diamond-shaped style.

`instrumentSwitch` - *name* (string)
Switch instrument to *name*, which must be predefined with `\addInstrumentDefinition`.

`keepWithTag` - *tag* (symbol) *music* (music)
Include only elements of *music* that are tagged with *tag*.

`killCues` - *music* (music)
Remove cue notes from *music*.

`label` - *label* (symbol)
Create *label* as a bookmarking label.

`language` - *str* (string)
Select note names language.

`makeClusters` - *arg* (music)
Display chords in *arg* as clusters.

`musicMap` - *proc* (procedure) *mus* (music)
Apply *proc* to *mus* and all of the music it contains.

`noPageBreak`
Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`noPageTurn`
: Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.

`octaveCheck` - *pitch-note* (music)
: Octave check.

`ottava` - *octave* (integer)
: Set the octavation.

`overrideProperty` - *name* (string) *property* (symbol) *value* (any type)
: Set *property* to *value* in all grobs named *name*. The *name* argument is a string of the form `"Context.GrobName"` or `"GrobName"`.

`overrideTimeSignatureSettings` - *context* (symbol) *time-signature* (pair) *base-moment* (pair) *beat-structure* (list) *beam-exceptions* (list)
: Override `timeSignatureSettings` in *context* for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.

`pageBreak`
: Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`pageTurn` Force a page turn between two scores or top-level markups.

`palmMute` - *note* (music)
: Print *note* with a triangle-shaped note head.

`palmMuteOn`
: Set the default note head style to a triangle-shaped style.

`parallelMusic` - *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 | }
```

`parenthesize` - *arg* (music)
: Tag *arg* to be parenthesized.

`partcombine` - *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff.

`partcombineForce` - *type* (symbol-or-boolean) *once* (boolean)
: Override the part-combiner.

`phrasingSlurDashPattern` - *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.

`pitchedTrill` - *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`
: Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

`pointAndClickOn`
: 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.

`quoteDuring` - *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.

`removeWithTag` - *tag* (symbol) *music* (music)
: Remove elements of *music* that are tagged with *tag*.

`resetRelativeOctave` - *reference-note* (music)
: Set the octave inside a \relative section.

`revertTimeSignatureSettings` - *context* (symbol) *time-signature* (pair)
: Revert `timeSignatureSettings` in *context* for time signatures of *time-signature*.

`rightHandFinger` - *finger* (number or string)
: Apply *finger* as a fingering indication.

`scaleDurations` - *fraction* (pair of numbers) *music* (music)
: Multiply the duration of events in *music* by *fraction*.

`shiftDurations` - *dur* (integer) *dots* (integer) *arg* (music)
: Scale *arg* up by a factor of *2^dur*(2-(1/2)^dots)*.

`slurDashPattern` - *dash-fraction* (number) *dash-period* (number)
: (undocumented; fixme)

`spacingTweaks` - *parameters* (list)
: Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

`storePredefinedDiagram` - *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*.

`styledNoteHeads` - *style* (symbol) *heads* (list or symbol) *music* (music)
: Set *heads* in *music* to *style*.

`tabChordRepetition`
: Include the string information in a chord repetition.

`tag` - *tag* (symbol) *arg* (music)
: Add *tag* to the `tags` property of *arg*.

`tieDashPattern` - *dash-fraction* (number) *dash-period* (number)
: (undocumented; fixme)

`tocItem` - *text* (markup)
: Add a line to the table of content, using the `tocItemMarkup` paper variable markup

`transposedCueDuring` - *what* (string) *dir* (direction) *pitch-note* (music) *main-music* (music)
: Insert notes from the part *what* into a voice called `cue`, using the transposition defined by *pitch-note*. 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` - *pitch-note* (music)
: Set instrument transposition

`tweak` - *sym* (symbol) *val* (any type) *arg* (music)
: Add `sym . val` to the `tweaks` property of *arg*.

`unfoldRepeats` - *music* (music)
: Force any `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`.

`withMusicProperty` - *sym* (symbol) *val* (any type) *music* (music)
: Set *sym* to *val* in *music*.

`xNote` - *note* (music)
: Print *note* with a cross-shaped note head.

`xNotesOn` Set the default note head style to a cross-shaped style.

A.16 Predefined type predicates

R5RS primary predicates

Type predicate	**Description**
`boolean?`	boolean
`char?`	character
`number?`	number
`pair?`	pair
`port?`	port
`procedure?`	procedure
`string?`	string
`symbol?`	symbol
`vector?`	vector

R5RS secondary predicates

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
`complex?`	complex number
`eof-object?`	end-of-file object
`even?`	even number
`exact?`	exact number
`inexact?`	inexact number
`input-port?`	input port
`integer?`	integer
`list?`	list *(use* `cheap-list?` *for faster processing)*

`negative?`	negative number
`null?`	null
`odd?`	odd number
`output-port?`	output port
`positive?`	positive number
`rational?`	rational number
`real?`	real number
`zero?`	zero

Guile predicates

Type predicate	**Description**
`hash-table?`	hash table

LilyPond scheme predicates

Type predicate	**Description**
`boolean-or-symbol?`	boolean or symbol
`cheap-list?`	list *(use this instead of* `list?` *for faster processing)*
`color?`	color
`grob-list?`	list of grobs
`list-or-symbol?`	list or symbol
`markup?`	markup
`markup-command-list?`	markup command list
`markup-list?`	markup list
`moment-pair?`	pair of moment objects
`number-or-grob?`	number or grob
`number-or-string?`	number or string
`number-pair?`	pair of numbers
`rhythmic-location?`	rhythmic location
`scheme?`	any type
`string-or-pair?`	string or pair
`string-or-symbol?`	string or symbol

LilyPond exported predicates

Type predicate	**Description**
`ly:box?`	box
`ly:context?`	context
`ly:dimension?`	dimension, in staff space
`ly:dir?`	direction
`ly:dispatcher?`	dispatcher
`ly:duration?`	duration
`ly:font-metric?`	font metric
`ly:grob?`	graphical (layout) object
`ly:grob-array?`	array of grobs
`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:simple-closure?`	simple closure
`ly:skyline?`	skyline
`ly:skyline-pair?`	pair of skylines
`ly:source-file?`	source file
`ly:spanner?`	spanner
`ly:stencil?`	stencil
`ly:stream-event?`	stream event
`ly:translator?`	translator
`ly:translator-group?`	translator group

A.17 Scheme functions

`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` *list disp cl* [Fonction]
Add the listener *list* to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it is forwarded to *list*.

`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: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: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-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:box?` *x* [Fonction]
Is *x* a `Box` object?

`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: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:command-line-verbose?` [Fonction]
Was `be_verbose_global` set?

`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-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-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 a single `\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:default-scale` [Fonction]
Get the global default scale.

`ly:dimension?` *d* [Fonction]
Return *d* as a number. 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: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:effective-prefix` [Fonction]
Return effective prefix.

`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:eval-simple-closure` *delayed closure scm-start scm-end* [Fonction]
Evaluate a simple *closure* with the given *delayed* argument. If *scm-start* and *scm-end* are defined, evaluate it purely with those start and end points.

`ly:event-deep-copy` *m* [Fonction]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym* [Fonction]
Get the property *sym* of stream event *mus*. If *sym* is undefined, return `'()`.

`ly:event-set-property!` *ev sym val* [Fonction]
Set property *sym* in event *ev* to *val*.

`ly:expand-environment` *str* [Fonction]
Expand `$VAR` and `${VAR}` in *str*.

`ly:export` *arg* [Fonction]
Export a Scheme object to the parser so it is treated as an identifier.

`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: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-context-mods` *contextmod* [Fonction]
Returns the list of context modifications stored in *contextmod*.

`ly:get-listened-event-classes` [Fonction]
Return a list of all event classes that some translator listens to.

`ly:get-option` *var* [Fonction]
Get a global option setting.

`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-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-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` *grob* [Fonction]
Get the mutable properties of *grob*.

`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-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-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: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: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* an `input-location`?

`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: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:interpret-stencil-expression` *expr func arg1 offset* [Fonction]
Parse *expr*, feed bits to *func* with first arg *arg1* having offset *offset*.

`ly:intlog2` *d* [Fonction]
The 2-logarithm of $1/d$.

`ly:is-listened-event-class` *sym* [Fonction]
Is *sym* a listened event class?

`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:iterator?` *x* [Fonction]
Is *x* a `Music_iterator` object?

`ly:lexer-keywords` *lexer* [Fonction]
Return a list of `(KEY . CODE)` pairs, signifying the LilyPond reserved words list.

`ly:lily-lexer?` *x* [Fonction]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Fonction]
Is *x* a `Lily_parser` object?

`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-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 *num* and *den*.

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-listener` *callback* [Fonction]
Create a listener. Any time the listener hears an object, it will call *callback* with that object.

callback should take exactly one argument.

`ly:make-moment` *n d gn gd* [Fonction]
Create the rational number with main timing n/d, and optional grace timing gn/gd.

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.

`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* is a list containing either `ly:music?` predicates or other type predicates.

`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 format* [Fonction]
Create an outputter that evaluates within `output-`*format*, writing to *port*.

`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. *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-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-score` *music* [Fonction]
Return score with *music* encapsulated in it.

`ly:make-simple-closure` *expr* [Fonction]
Make a simple closure. *expr* should be form of `(`*`func a1 a2 ...`*`)`, and will be invoked as `(`*`func delayed-arg a1 a2 ...`*`)`.

`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 `(1000 . -1000)` 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: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-denominator` *mom* [Fonction]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Fonction]
Extract numerator from grace timing.

`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 moment *factor*.

`ly:music-deep-copy` *m* [Fonction]
Copy *m* and all sub expressions of *m*.

`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`?

`ly:music-function-extract` *x* [Fonction]
Return the Scheme function 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-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* [Fonction]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name*.

`ly:number->string` *s* [Fonction]
Convert *s* to a string without generating many decimals.

`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` [Fonction]
Print `ly:set-option` usage.

`ly:otf->cff` *otf-file-name* [Fonction]
Convert the contents of an OTF file to a CFF file, returning it as a string.

`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?` *def* [Fonction]
Is *def* an output definition?

`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-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-module` *outputter* [Fonction]
Return output module of *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Fonction]
Eval *expr* in module of *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-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-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:parser-clear-error` *parser* [Fonction]
Clear the error flag for the parser.

`ly:parser-clone` *parser-smob* [Fonction]
Return a clone of *parser-smob*.

`ly:parser-define!` *parser-smob symbol val* [Fonction]
Bind *symbol* to *val* in *parser-smob*'s module.

`ly:parser-error` *parser msg input* [Fonction]
Display an error message and make the parser fail.

`ly:parser-has-error?` *parser* [Fonction]
Does *parser* have an error flag?

`ly:parser-include-string` *parser-smob ly-code* [Fonction]
Include the string *ly-code* into the input stream for *parser-smob*.

`ly:parser-lexer` *parser-smob* [Fonction]
Return the lexer for *parser-smob*.

`ly:parser-lookup` *parser-smob symbol* [Fonction]
Look up *symbol* in *parser-smob*'s module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Fonction]
Return the base name of the output file.

`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` *parser 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:parser-set-repetition-function` *parser fun* [Fonction]
Replace the current repetition function in *parser*. *fun* is the new repetition function.

`ly:parser-set-repetition-symbol` *parser sym* [Fonction]
Replace the current repetition symbol in *parser*. *sym* is the new repetition symbol.

`ly:performance-write` *performance filename* [Fonction]
Write *performance* to *filename*.

`ly:pfb->pfa` *pfb-file-name* [Fonction]
Convert the contents of a Type 1 font in PFB format to PFA format.

`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 *pitch* transposed by *delta* equals *root*.

`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-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:register-stencil-expression` *symbol* [Fonction]
Add *symbol* as head of a stencil expression.

`ly:relative-group-extent` *elements common axis* [Fonction]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`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* [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*.

`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: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-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.

`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-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.

`ly:simple-closure?` *clos* [Fonction]
Is *clos* a simple closure?

`ly:skyline?` *x* [Fonction]
Is *x* a `Skyline` object?

`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: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:staff-symbol-line-thickness` *grob* [Fonction]
Returns the `line-thickness` of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Fonction]
Returns the `staff-space` of the staff associated with *grob*.

`ly:start-environment` [Fonction]
Return the environment (a list of strings) that was in effect at program start.

`ly:stderr-redirect` *file-name mode* [Fonction]
Redirect stderr 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 minimum* [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. If this puts the reference points closer than *minimum*, they are moved by the latter amount. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil* [Fonction]
Return whether *stil* is empty.

`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-fonts` *s* [Fonction]
Analyze *s*, and return a list of fonts used in *s*.

`ly:stencil-in-color` *stc r g b* [Fonction]
Put *stc* in a different color.

`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 *stil* using the horizontal and vertical scaling factors *x* and *y*.

`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:success` *str rest* [Fonction]
A Scheme callable function to issue a success message *str*. The message is formatted with `format` and *rest*.

`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:translator?` *x* [Fonction]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Fonction]
Return the context of the translator object *trans*.

`ly:translator-description` *me* [Fonction]
Return an alist of properties of translator *me*.

`ly:translator-group?` *x* [Fonction]
Is *x* a `Translator_group` object?

`ly:translator-name` *trans* [Fonction]
Return the type name of the translator object *trans*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Fonction]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:truncate-list!` *lst i* [Fonction]
Take at most the first *i* of list *lst*.

`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:unit` [Fonction]
Return the unit used for lengths as a string.
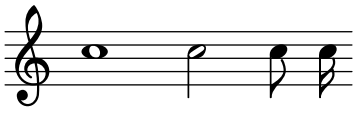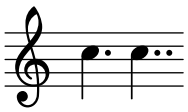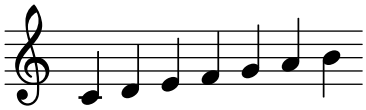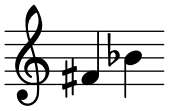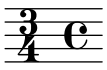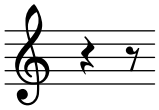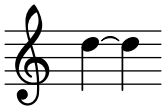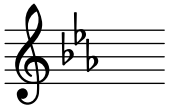
`ly:usage` [Fonction]
Print usage message.
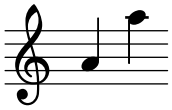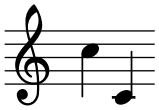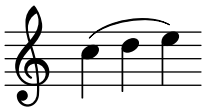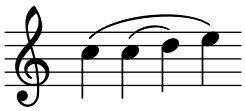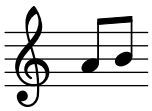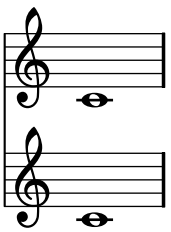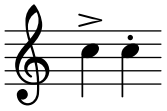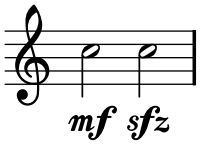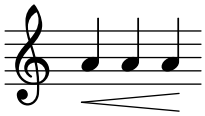
`ly:version` [Fonction]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.

`ly:warning` *str rest* [Fonction]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.
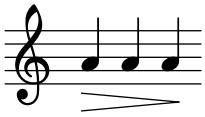
`ly:wide-char->utf-8` *wc* [Fonction]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.

Annexe B Cheat sheet
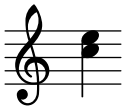
Syntax	Description	Example
`1 2 8 16`	durations	
`c4. c4..`	augmentation dots	
`c d e f g a b`	scale	
`fis bes`	alteration	
`\clef treble \clef bass`	clefs	
`\time 3/4 \time 4/4`	time signature	
`r4 r8`	rest	
`d ~ d`	tie	
`\key es \major`	key signature	

note`'`	raise octave	
note`,`	lower octave	
`c( d e)`	slur	
`c\( c( d) e\)`	phrasing slur	
`a8[ b]`	beam	
`<< \new Staff ... >>`	more staves	
`c-> c-.`	articulations	
`c2\mf c\sfz`	dynamics	
`a\< a a\!`	crescendo	
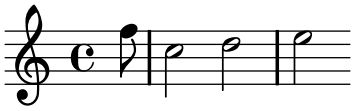
`a\> a a\!` decrescendo
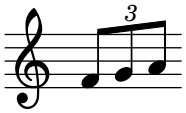


`< >` chord
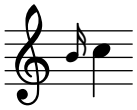


`\partial 8` pickup / upbeat



`\times 2/3 {f g a}` triplets



`\grace` grace notes
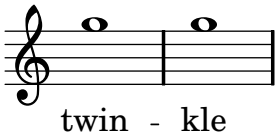


`\lyricmode { twinkle }` entering lyrics twinkle
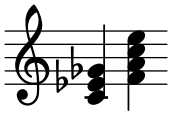
`\new Lyrics` printing lyrics

twinkle

`twin -- kle` lyric hyphen
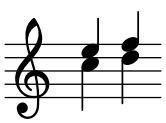


`\chordmode { c:dim f:maj7 }` chords



`\context ChordNames` printing chord names

C° F△

`<<{e f} \\ {c d}>>` polyphony



`s4 s8 s16`	spacer rests

Annexe C Grammaire de LilyPond

Cette annexe présente la grammaire du langage de LilyPond, telle qu'elle ressort de l'analyseur syntaxique.

`Grammar`

```
lilypond: /* empty */
        | lilypond toplevel_expression
        | lilypond assignment
        | lilypond error
        | lilypond "\invalid"

toplevel_expression: lilypond_header
                   | book_block
                   | bookpart_block
                   | score_block
                   | composite_music
                   | full_markup
                   | full_markup_list
                   | output_def

embedded_scm: SCM_TOKEN
            | SCM_IDENTIFIER

lilypond_header_body: /* empty */
                    | lilypond_header_body assignment

lilypond_header: "\header" '{' lilypond_header_body '}'

assignment_id: STRING
             | LYRICS_STRING

assignment: assignment_id '=' identifier_init
          | assignment_id property_path '=' identifier_init
          | embedded_scm

identifier_init: score_block
               | book_block
               | bookpart_block
               | output_def
               | context_def_spec_block
               | music
               | post_event
               | number_expression
               | string
               | embedded_scm
               | full_markup
               | DIGIT
               | context_modification

context_def_spec_block: "\context" '{' context_def_spec_body '}'
```

```
38 context_def_spec_body: /* empty */
39                      | CONTEXT_DEF_IDENTIFIER
40                      | context_def_spec_body
                            "\grobdescriptions"
                            embedded_scm
41                      | context_def_spec_body context_mod
42                      | context_def_spec_body context_modification

43 book_block: "\book" '{' book_body '}'

44 book_body: /* empty */
45          | BOOK_IDENTIFIER
46          | book_body paper_block
47          | book_body bookpart_block
48          | book_body score_block
49          | book_body composite_music
50          | book_body full_markup
51          | book_body full_markup_list
52          | book_body lilypond_header
53          | book_body error

54 bookpart_block: "\bookpart" '{' bookpart_body '}'

55 bookpart_body: /* empty */
56              | BOOK_IDENTIFIER
57              | bookpart_body paper_block
58              | bookpart_body score_block
59              | bookpart_body composite_music
60              | bookpart_body full_markup
61              | bookpart_body full_markup_list
62              | bookpart_body lilypond_header
63              | bookpart_body error

64 score_block: "\score" '{' score_body '}'

65 score_body: music
66           | SCORE_IDENTIFIER
67           | score_body lilypond_header
68           | score_body output_def
69           | score_body error

70 paper_block: output_def

71 output_def: output_def_body '}'

72 output_def_head: "\paper"
73                | "\midi"
74                | "\layout"

75 output_def_head_with_mode_switch: output_def_head

76 output_def_body: output_def_head_with_mode_switch '{'
```

```
 77                | output_def_head_with_mode_switch
                       '{'
                       OUTPUT_DEF_IDENTIFIER
 78                | output_def_body assignment
 79                | output_def_body context_def_spec_block
 80                | output_def_body error

 81 tempo_event: "\tempo" steno_duration '=' bare_unsigned
 82            | "\tempo" string steno_duration '=' bare_unsigned
 83            | "\tempo" full_markup steno_duration '=' bare_unsigned
 84            | "\tempo" string
 85            | "\tempo" full_markup

 86 music_list: /* empty */
 87           | music_list music
 88           | music_list embedded_scm
 89           | music_list error

 90 music: simple_music
 91      | composite_music

 92 alternative_music: /* empty */
 93                  | "\alternative" '{' music_list '}'

 94 repeated_music: "\repeat"
                     simple_string
                     unsigned_number
                     music
                     alternative_music

 95 sequential_music: "\sequential" '{' music_list '}'
 96                 | '{' music_list '}'

 97 simultaneous_music: "\simultaneous" '{' music_list '}'
 98                   | "<<" music_list ">>"

 99 simple_music: event_chord
100             | MUSIC_IDENTIFIER
101             | music_property_def
102             | context_change

104 context_modification: "\with"  '{' context_mod_list '}'
105                     | "\with" CONTEXT_MOD_IDENTIFIER
106                     | CONTEXT_MOD_IDENTIFIER

107 optional_context_mod: /* empty */
108                     | context_modification

109 context_mod_list: /* empty */
110                 | context_mod_list context_mod
111                 | context_mod_list CONTEXT_MOD_IDENTIFIER
```

```
112 composite_music: prefix_composite_music
113                | grouped_music_list

114 grouped_music_list: simultaneous_music
115                   | sequential_music

116 function_scm_argument: embedded_scm
117                      | simple_string

118 function_arglist_music_last: EXPECT_MUSIC function_arglist music

119 function_arglist_nonmusic_last: EXPECT_MARKUP
                                      function_arglist
                                      full_markup
120                                 | EXPECT_MARKUP
                                      function_arglist
                                      simple_string
121                                 | EXPECT_SCM
                                      function_arglist
                                      function_scm_argument

122 function_arglist_nonmusic: EXPECT_NO_MORE_ARGS
123                            | EXPECT_MARKUP
                                 function_arglist_nonmusic
                                 full_markup
124                            | EXPECT_MARKUP
                                 function_arglist_nonmusic
                                 simple_string
125                            | EXPECT_SCM
                                 function_arglist_nonmusic
                                 function_scm_argument

126 function_arglist: EXPECT_NO_MORE_ARGS
127                 | function_arglist_music_last
128                 | function_arglist_nonmusic_last

129 generic_prefix_music_scm: MUSIC_FUNCTION function_arglist

130 optional_id: /* empty */
131            | '=' simple_string

132 prefix_composite_music: generic_prefix_music_scm
133                        | "\context"
                             simple_string
                             optional_id
                             optional_context_mod
                             music
134                        | "\new"
                             simple_string
                             optional_id
                             optional_context_mod
                             music
```

```
135                       | "\times" fraction music
136                       | repeated_music
137                       | "\transpose"
                              pitch_also_in_chords
                              pitch_also_in_chords
                              music
138                       | mode_changing_head grouped_music_list
139                       | mode_changing_head_with_context
                              optional_context_mod
                              grouped_music_list
140                       | relative_music
141                       | re_rhythmed_music

142 mode_changing_head: "\notemode"
143                   | "\drummode"
144                   | "\figuremode"
145                   | "\chordmode"
146                   | "\lyricmode"

147 mode_changing_head_with_context: "\drums"
148                                 | "\figures"
149                                 | "\chords"
150                                 | "\lyrics"

151 relative_music: "\relative" absolute_pitch music
152               | "\relative" composite_music

154 new_lyrics: "\addlyrics"  grouped_music_list

156 new_lyrics: new_lyrics "\addlyrics"  grouped_music_list

157 re_rhythmed_music: grouped_music_list new_lyrics

159 re_rhythmed_music: "\lyricsto" simple_string  music

160 context_change: "\change" STRING '=' STRING

161 property_path_revved: embedded_scm
162                     | property_path_revved embedded_scm

163 property_path: property_path_revved

164 property_operation: STRING '=' scalar
165                   | "\unset" simple_string
166                   | "\override" simple_string property_path '=' scalar
167                   | "\revert" simple_string embedded_scm

168 context_def_mod: "\consists"
169                | "\remove"
170                | "\accepts"
171                | "\defaultchild"
172                | "\denies"
```

```
173                | "\alias"
174                | "\type"
175                | "\description"
176                | "\name"

177 context_mod: property_operation
178            | context_def_mod STRING
179            | context_def_mod embedded_scm

180 context_prop_spec: simple_string
181                  | simple_string '.' simple_string

182 simple_music_property_def: "\override"
                               context_prop_spec
                               property_path
                               '='
                               scalar
183                          | "\revert" context_prop_spec embedded_scm
184                          | "\set" context_prop_spec '=' scalar
185                          | "\unset" context_prop_spec

186 music_property_def: simple_music_property_def
187                   | "\once" simple_music_property_def

188 string: STRING
189       | STRING_IDENTIFIER
190       | string '+' string

191 simple_string: STRING
192              | LYRICS_STRING
193              | STRING_IDENTIFIER

194 scalar: string
195       | LYRICS_STRING
196       | bare_number
197       | embedded_scm
198       | full_markup
199       | DIGIT

200 event_chord: simple_chord_elements post_events
201            | CHORD_REPETITION optional_notemode_duration post_events
202            | MULTI_MEASURE_REST optional_notemode_duration post_events
203            | command_element
204            | note_chord_element

205 note_chord_element: chord_body optional_notemode_duration post_events

206 chord_body: "<" chord_body_elements ">"

207 chord_body_elements: /* empty */
208                    | chord_body_elements chord_body_element
```

```
209 chord_body_element: pitch
                        exclamations
                        questions
                        octave_check
                        post_events
210                   | DRUM_PITCH post_events
211                   | music_function_chord_body

212 music_function_identifier_musicless_prefix: MUSIC_FUNCTION

213 music_function_chord_body: music_function_identifier_musicless_prefix
                                EXPECT_MUSIC
                                function_arglist_nonmusic
                                chord_body_element
214                           | music_function_identifier_musicless_prefix
                                function_arglist_nonmusic

215 music_function_event: music_function_identifier_musicless_prefix
                          EXPECT_MUSIC
                          function_arglist_nonmusic
                          post_event
216                     | music_function_identifier_musicless_prefix
                          function_arglist_nonmusic

217 command_element: command_event
218                | "\skip" duration_length
219                | "\["
220                | "\]"
221                | "\"
222                | '|'
223                | "\partial" duration_length
224                | "\time" fraction
225                | "\mark" scalar

226 command_event: "\~"
227              | "\mark" "\default"
228              | tempo_event
229              | "\key" "\default"
230              | "\key" NOTENAME_PITCH SCM_IDENTIFIER

231 post_events: /* empty */
232            | post_events post_event

233 post_event: direction_less_event
234           | '-' music_function_event
235           | "--"
236           | "__"
237           | script_dir direction_reqd_event
238           | script_dir direction_less_event
239           | string_number_event

240 string_number_event: E_UNSIGNED
```

```
direction_less_char: '['
                   | ']'
                   | '~'
                   | '('
                   | ')'
                   | "\!"
                   | "\("
                   | "\)"
                   | "\>"
                   | "\<"

direction_less_event: direction_less_char
                    | EVENT_IDENTIFIER
                    | tremolo_type

direction_reqd_event: gen_text_def
                    | script_abbreviation

octave_check: /* empty */
            | '='
            | '=' sub_quotes
            | '=' sup_quotes

sup_quotes: '''
          | sup_quotes '''

sub_quotes: ','
          | sub_quotes ','

steno_pitch: NOTENAME_PITCH
           | NOTENAME_PITCH sup_quotes
           | NOTENAME_PITCH sub_quotes

steno_tonic_pitch: TONICNAME_PITCH
                 | TONICNAME_PITCH sup_quotes
                 | TONICNAME_PITCH sub_quotes

pitch: steno_pitch

pitch_also_in_chords: pitch
                    | steno_tonic_pitch

gen_text_def: full_markup
            | string
            | DIGIT

script_abbreviation: '^'
                   | '+'
                   | '-'
                   | '|'
                   | ">"
```

```
281                   | '.'
282                   | '_'

283 script_dir: '_'
284           | '^'
285           | '-'

286 absolute_pitch: steno_pitch

287 duration_length: multiplied_duration

288 optional_notemode_duration: /* empty */
289                           | multiplied_duration

290 steno_duration: bare_unsigned dots
291               | DURATION_IDENTIFIER dots

292 multiplied_duration: steno_duration
293                   | multiplied_duration '*' bare_unsigned
294                   | multiplied_duration '*' FRACTION

295 fraction: FRACTION
296         | UNSIGNED '/' UNSIGNED

297 dots: /* empty */
298     | dots '.'

299 tremolo_type: ':'
300             | ':' bare_unsigned

301 bass_number: DIGIT
302            | UNSIGNED
303            | STRING
304            | full_markup

305 figured_bass_alteration: '-'
306                        | '+'
307                        | '!'

308 bass_figure: "_"
309            | bass_number
310            | bass_figure ']'
311            | bass_figure figured_bass_alteration
312            | bass_figure figured_bass_modification

313 figured_bass_modification: "\+"
314                          | "\!"
315                          | '/'
316                          | "\"

317 br_bass_figure: bass_figure
318               | '[' bass_figure
```

```
319 figure_list: /* empty */
320            | figure_list br_bass_figure

321 figure_spec: FIGURE_OPEN figure_list FIGURE_CLOSE

322 optional_rest: /* empty */
323              | "\rest"

324 simple_element: pitch
                      exclamations
                      questions
                      octave_check
                      optional_notemode_duration
                      optional_rest
325               | DRUM_PITCH optional_notemode_duration
326               | RESTNAME optional_notemode_duration
327               | lyric_element optional_notemode_duration

328 simple_chord_elements: simple_element
329                      | new_chord
330                      | figure_spec optional_notemode_duration

331 lyric_element: lyric_markup
332              | LYRICS_STRING

333 new_chord: steno_tonic_pitch optional_notemode_duration
334          | steno_tonic_pitch
                optional_notemode_duration
                chord_separator
                chord_items

335 chord_items: /* empty */
336            | chord_items chord_item

337 chord_separator: ":"
338                | "^"
339                | "/" steno_tonic_pitch
340                | "/+" steno_tonic_pitch

341 chord_item: chord_separator
342           | step_numbers
343           | CHORD_MODIFIER

344 step_numbers: step_number
345             | step_numbers '.' step_number

346 step_number: bare_unsigned
347            | bare_unsigned '+'
348            | bare_unsigned "-"

349 number_expression: number_expression '+' number_term
```

```
350                   | number_expression '-' number_term
351                   | number_term

352 number_term: number_factor
353            | number_factor '*' number_factor
354            | number_factor '/' number_factor

355 number_factor: '-' number_factor
356              | bare_number

357 bare_number: UNSIGNED
358            | REAL
359            | NUMBER_IDENTIFIER
360            | REAL NUMBER_IDENTIFIER
361            | UNSIGNED NUMBER_IDENTIFIER

362 bare_unsigned: UNSIGNED
363              | DIGIT

364 unsigned_number: bare_unsigned
365                | NUMBER_IDENTIFIER

366 exclamations: /* empty */
367             | exclamations '!'

368 questions: /* empty */
369          | questions '?'

370 lyric_markup: LYRIC_MARKUP_IDENTIFIER

372 lyric_markup: LYRIC_MARKUP  markup_top

374 full_markup_list: "\markuplines"  markup_list

375 full_markup: MARKUP_IDENTIFIER

377 full_markup: "\markup"  markup_top

378 markup_top: markup_list
379           | markup_head_1_list simple_markup
380           | simple_markup

381 markup_list: markup_composed_list
382            | markup_braced_list
383            | markup_command_list

384 markup_composed_list: markup_head_1_list markup_braced_list

385 markup_braced_list: '{' markup_braced_list_body '}'

386 markup_braced_list_body: /* empty */
387                        | markup_braced_list_body markup
```

```
  388                         | markup_braced_list_body markup_list

  389 markup_command_list: MARKUP_LIST_FUNCTION markup_command_list_arguments

  390 markup_command_basic_arguments: EXPECT_MARKUP_LIST
                                        markup_command_list_arguments
                                        markup_list
  391                                 | EXPECT_SCM
                                        markup_command_list_arguments
                                        embedded_scm
  392                                 | EXPECT_NO_MORE_ARGS

  393 markup_command_list_arguments: markup_command_basic_arguments
  394                                | EXPECT_MARKUP
                                       markup_command_list_arguments
                                       markup

  395 markup_head_1_item: MARKUP_FUNCTION
                            EXPECT_MARKUP
                            markup_command_list_arguments

  396 markup_head_1_list: markup_head_1_item
  397                   | markup_head_1_list markup_head_1_item

  398 simple_markup: STRING
  399              | MARKUP_IDENTIFIER
  400              | LYRIC_MARKUP_IDENTIFIER
  401              | STRING_IDENTIFIER

  403 simple_markup: "\score"  '{' score_body '}'
  404              | MARKUP_FUNCTION markup_command_basic_arguments

  405 markup: markup_head_1_list simple_markup
  406       | simple_markup


Terminals, with rules where they appear



"-" (319) 348
"--" (340) 235
"/" (320) 339
"/+" (316) 340
":" (318) 337
"<" (321) 206
"<<" (323) 98
">" (322) 206 280
">>" (324) 98
"\!" (329) 246 314
"\" (325) 221 316
"\(" (331) 247
```

"\)" (328) 248
"\+" (334) 313
"\<" (333) 250
"\>" (326) 249
"\[" (330) 219
"\]" (332) 220
"\accepts" (261) 170
"\addlyrics" (259) 154 156
"\alias" (262) 173
"\alternative" (263) 93
"\book" (264) 43
"\bookpart" (265) 54
"\C[haracter]" (327)
"\change" (266) 160
"\chordmode" (267) 145
"\chords" (268) 149
"\consists" (269) 168
"\context" (270) 37 133
"\default" (271) 227 229
"\defaultchild" (272) 171
"\denies" (273) 172
"\description" (274) 175
"\drummode" (275) 143
"\drums" (276) 147
"\figuremode" (277) 144
"\figures" (278) 148
"\grobdescriptions" (279) 40
"\header" (280) 18
"\invalid" (281) 5
"\key" (282) 229 230
"\layout" (283) 74
"\lyricmode" (284) 146
"\lyrics" (285) 150
"\lyricsto" (286) 159
"\mark" (287) 225 227
"\markup" (288) 377
"\markuplines" (289) 374
"\midi" (290) 73
"\name" (291) 176
"\new" (315) 134
"\notemode" (292) 142
"\octave" (293)
"\once" (294) 187
"\override" (295) 166 182
"\paper" (296) 72
"\partial" (297) 223
"\relative" (298) 151 152
"\remove" (299) 169
"\repeat" (300) 94
"\rest" (301) 323
"\revert" (302) 167 183
"\score" (303) 64 403

```
"\sequential" (304) 95
"\set" (305) 184
"\simultaneous" (306) 97
"\skip" (307) 218
"\tempo" (308) 81 82 83 84 85
"\time" (314) 224
"\times" (309) 135
"\transpose" (310) 137
"\type" (311) 174
"\unset" (312) 165 185
"\with" (313) 104 105
"\~" (335) 226
"^" (317) 338
"_" (339) 308
"__" (336) 236
$end (0) 0
'!' (33) 307 367
''' (39) 260 261
'(' (40) 244
')' (41) 245
'*' (42) 293 294 353
'+' (43) 190 277 306 347 349
',' (44) 262 263
'-' (45) 234 278 285 305 350 355
'.' (46) 181 281 298 345
'/' (47) 296 315 354
':' (58) 299 300
'=' (61) 21 22 81 82 83 131 160 164 166 182 184 257 258 259
'?' (63) 369
'[' (91) 241 318
']' (93) 242 310
'^' (94) 276 284
'_' (95) 282 283
'{' (123) 18 37 43 54 64 76 77 93 95 96 97 104 385 403
'|' (124) 222 279
'}' (125) 18 37 43 54 64 71 93 95 96 97 104 385 403
'~' (126) 243
BOOK_IDENTIFIER (352) 45 56
CHORD_MODIFIER (354) 343
CHORD_REPETITION (355) 201
CHORDMODIFIER_PITCH (353)
CHORDMODIFIERS (341)
CONTEXT_DEF_IDENTIFIER (356) 39
CONTEXT_MOD_IDENTIFIER (357) 105 106 111
DIGIT (344) 35 199 275 301 363
DRUM_PITCH (358) 210 325
DURATION_IDENTIFIER (359) 291
E_UNSIGNED (345) 240
error (256) 4 53 63 69 80 89
EVENT_IDENTIFIER (360) 252
EXPECT_MARKUP (347) 119 120 123 124 394 395
EXPECT_MARKUP_LIST (350) 390
```

```
EXPECT_MUSIC (348) 118 213 215
EXPECT_NO_MORE_ARGS (351) 122 126 392
EXPECT_SCM (349) 121 125 391
FIGURE_CLOSE (337) 321
FIGURE_OPEN (338) 321
FRACTION (361) 294 295
LYRIC_MARKUP (342) 372
LYRIC_MARKUP_IDENTIFIER (363) 370 400
LYRICS_STRING (362) 20 192 195 332
MARKUP_FUNCTION (364) 395 404
MARKUP_IDENTIFIER (366) 375 399
MARKUP_LIST_FUNCTION (365) 389
MULTI_MEASURE_REST (343) 202
MUSIC_FUNCTION (367) 129 212
MUSIC_IDENTIFIER (368) 100
NOTENAME_PITCH (369) 230 264 265 266
NUMBER_IDENTIFIER (370) 359 360 361 365
OUTPUT_DEF_IDENTIFIER (371) 77
PREC_BOT (260)
PREC_TOP (258)
REAL (372) 358 360
RESTNAME (373) 326
SCM_IDENTIFIER (374) 15 230
SCM_TOKEN (375) 14
SCORE_IDENTIFIER (376) 66
STRING (377) 19 160 164 178 188 191 303 398
STRING_IDENTIFIER (378) 189 193 401
TONICNAME_PITCH (379) 267 268 269
UNARY_MINUS (380)
UNSIGNED (346) 296 302 357 361 362

Nonterminals, with rules where they appear


absolute_pitch (237)
    on left: 286, on right: 151
alternative_music (172)
    on left: 92 93, on right: 94
assignment (154)
    on left: 21 22 23, on right: 3 17 78
assignment_id (153)
    on left: 19 20, on right: 21 22
bare_number (265)
    on left: 357 358 359 360 361, on right: 196 356
bass_number (245)
    on left: 301 302 303 304, on right: 309
book_block (158)
    on left: 43, on right: 7 25
bookpart_block (160)
    on left: 54, on right: 8 26 47
br_bass_figure (249)
    on left: 317 318, on right: 320
```

```
chord_body (213)
    on left: 206, on right: 205
chord_body_element (215)
    on left: 209 210 211, on right: 208 213
chord_body_elements (214)
    on left: 207 208, on right: 206 208
chord_item (259)
    on left: 341 342 343, on right: 336
chord_items (257)
    on left: 335 336, on right: 334 336
chord_separator (258)
    on left: 337 338 339 340, on right: 334 341
command_event (220)
    on left: 226 227 228 229 230, on right: 217
composite_music (181)
    on left: 112 113, on right: 10 49 59 91 152
context_change (199)
    on left: 160, on right: 102
context_def_spec_block (156)
    on left: 37, on right: 28 79
context_mod (204)
    on left: 177 178 179, on right: 41 110
context_mod_list (180)
    on left: 109 110 111, on right: 104 110 111
context_modification (177)
    on left: 104 105 106, on right: 36 42 108
context_prop_spec (205)
    on left: 180 181, on right: 182 183 184 185
direction_less_event (225)
    on left: 251 252 253, on right: 233 238
direction_reqd_event (226)
    on left: 254 255, on right: 237
dots (243)
    on left: 297 298, on right: 290 291 298
duration_length (238)
    on left: 287, on right: 218 223
event_chord (211)
    on left: 200 201 202 203 204, on right: 99
exclamations (268)
    on left: 366 367, on right: 209 324 367
figure_list (250)
    on left: 319 320, on right: 320 321
figure_spec (251)
    on left: 321, on right: 330
figured_bass_alteration (246)
    on left: 305 306 307, on right: 311
figured_bass_modification (248)
    on left: 313 314 315 316, on right: 312
fraction (242)
    on left: 295 296, on right: 135 224
full_markup_list (272)
    on left: 374, on right: 12 51 61
```

```
function_arglist_music_last (184)
    on left: 118, on right: 127
function_arglist_nonmusic_last (185)
    on left: 119 120 121, on right: 128
function_scm_argument (183)
    on left: 116 117, on right: 121 125
gen_text_def (234)
    on left: 273 274 275, on right: 254
generic_prefix_music_scm (188)
    on left: 129, on right: 132
lilypond (148)
    on left: 1 2 3 4 5, on right: 0 2 3 4 5
lilypond_header (152)
    on left: 18, on right: 6 52 62 67
lilypond_header_body (151)
    on left: 16 17, on right: 17 18
lyric_element (255)
    on left: 331 332, on right: 327
lyric_markup (270)
    on left: 370 372, on right: 331
markup (288)
    on left: 405 406, on right: 387 394
markup_braced_list (279)
    on left: 385, on right: 382 384
markup_braced_list_body (280)
    on left: 386 387 388, on right: 385 387 388
markup_command_list (281)
    on left: 389, on right: 383
markup_composed_list (278)
    on left: 384, on right: 381
markup_head_1_item (284)
    on left: 395, on right: 396 397
markup_head_1_list (285)
    on left: 396 397, on right: 379 384 397 405
markup_list (277)
    on left: 381 382 383, on right: 374 378 388 390
markup_top (276)
    on left: 378 379 380, on right: 372 377
mode_changing_head (191)
    on left: 142 143 144 145 146, on right: 138
multiplied_duration (241)
    on left: 292 293 294, on right: 287 289 293 294
music_function_chord_body (217)
    on left: 213 214, on right: 211
music_function_event (218)
    on left: 215 216, on right: 234
music_list (170)
    on left: 86 87 88 89, on right: 87 88 89 93 95 96 97 98
music_property_def (207)
    on left: 186 187, on right: 101
new_chord (256)
    on left: 333 334, on right: 329
```

```
new_lyrics (194)
    on left: 154 156, on right: 156 157
note_chord_element (212)
    on left: 205, on right: 204
number_expression (262)
    on left: 349 350 351, on right: 31 349 350
number_factor (264)
    on left: 355 356, on right: 352 353 354 355
number_term (263)
    on left: 352 353 354, on right: 349 350 351
octave_check (227)
    on left: 256 257 258 259, on right: 209 324
optional_context_mod (179)
    on left: 107 108, on right: 133 134 139
optional_id (189)
    on left: 130 131, on right: 133 134
optional_rest (252)
    on left: 322 323, on right: 324
output_def (165)
    on left: 71, on right: 13 27 68 70
output_def_body (168)
    on left: 76 77 78 79 80, on right: 71 78 79 80
output_def_head (166)
    on left: 72 73 74, on right: 75
output_def_head_with_mode_switch (167)
    on left: 75, on right: 76 77
paper_block (164)
    on left: 70, on right: 46 57
pitch (232)
    on left: 270, on right: 209 271 324
pitch_also_in_chords (233)
    on left: 271 272, on right: 137
post_events (221)
    on left: 231 232, on right: 200 201 202 205 209 210 232
property_operation (202)
    on left: 164 165 166 167, on right: 177
property_path (201)
    on left: 163, on right: 22 166 182
property_path_revved (200)
    on left: 161 162, on right: 162 163
questions (269)
    on left: 368 369, on right: 209 324 369
re_rhythmed_music (197)
    on left: 157 159, on right: 141
relative_music (193)
    on left: 151 152, on right: 140
repeated_music (173)
    on left: 94, on right: 136
score_block (162)
    on left: 64, on right: 9 24 48 58
score_body (163)
    on left: 65 66 67 68 69, on right: 64 67 68 69 403
```

```
script_dir (236)
    on left: 283 284 285, on right: 237 238
sequential_music (174)
    on left: 95 96, on right: 115
simple_chord_elements (254)
    on left: 328 329 330, on right: 200
simple_element (253)
    on left: 324 325 326 327, on right: 328
simple_music (176)
    on left: 99 100 101 102, on right: 90
simultaneous_music (175)
    on left: 97 98, on right: 114
steno_duration (240)
    on left: 290 291, on right: 81 82 83 292
steno_pitch (230)
    on left: 264 265 266, on right: 270 286
step_number (261)
    on left: 346 347 348, on right: 344 345
step_numbers (260)
    on left: 344 345, on right: 342 345
string (208)
    on left: 188 189 190, on right: 32 82 84 190 194 274
string_number_event (223)
    on left: 240, on right: 239
sub_quotes (229)
    on left: 262 263, on right: 258 263 266 269
sup_quotes (228)
    on left: 260 261, on right: 259 261 265 268
tempo_event (169)
    on left: 81 82 83 84 85, on right: 228
toplevel_expression (149)
    on left: 6 7 8 9 10 11 12 13, on right: 2
tremolo_type (244)
    on left: 299 300, on right: 253
unsigned_number (267)
    on left: 364 365, on right: 94
```

Annexe D GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://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 http://www.gnu.org/copyleft/.

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 E 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. Chaque lien est constitué de deux parties : la première pointe directement là où la commande ou le terme apparaît, la seconde au début de la section où la commande ou le terme est abordé.

!

! 5

'

' 1

,

, ... 1

-

- 99

.

. 37

/

/ 312
/+ 312

:

: 132

<

< 133
<...> 133

=

= 8

>

> 133

?

? 5

[

[..... 77

]

] 77

^

^ 311

_

_ 217, 221

\

\! 102
\(..... 110
\) 110
\< 102
\> 102
\\ 138
\abs-fontsize 515
\accent 99
\accepts 443, 444, 445
\addChordShape 271
\addInstrumentDefinition 169
\addlyrics 219
\addQuote 170
\aeolian 16
\afterGrace 93
\aikenHeads 33
\aikenHeadsMinor 33
\alias 443
\allowPageTurn 403
\alternative 120
\arpeggio 115
\arpeggioArrowDown 115
\arpeggioArrowUp 115
\arpeggioBracket 115
\arpeggioNormal 115
\arpeggioParenthesis 115
\arpeggioParenthesisDashed 115
\arrow-head 207, 537
\ascendens 340, 346
\auctum 340, 346
\augmentum 346
\autoBeamOff 69
\autoBeamOn 69
\autochange 235
\backslashed-digit 550
\balloonGrobText 185
\balloonLengthOff 185
\balloonLengthOn 185
\balloonText 185
\bar 81, 85
\barNumberCheck 90
\beam 537
\bendAfter 113
\bold 200, 515
\book 360, 363
\bookOutputName 362

\bookOutputSuffix ... 362
\bookpart ... 361, 363, 401
\box ... 206, 515
\bracket ... 106, 206, 538
\break ... 400
\breathe ... 112
\breve ... 37, 47
\cadenzaOff ... 61
\cadenzaOn ... 61
\caesura ... 338
\caps ... 516
\cavum ... 340, 346
\center-align ... 202, 524
\center-column ... 204, 524
\change ... 234
\char ... 550
\chordmode ... 3, 12, 269
\chords ... 314
\circle ... 206, 538
\clef ... 12
\cm ... 460
\coda ... 99
\column ... 204, 524
\column-lines ... 554
\combine ... 207, 525
\compressFullBarRests ... 51, 52
\concat ... 525
\consists ... 443
\context ... 437
\cr ... 102
\crescHairpin ... 103
\crescTextCresc ... 103
\cueDuring ... 174
\customTabClef ... 543
\decr ... 102
\defaultTimeSignature ... 55
\deminutum ... 340, 346
\denies ... 443, 444, 445
\descendens ... 340, 346
\dimHairpin ... 103
\dimTextDecr ... 103
\dimTextDecresc ... 103
\dimTextDim ... 103
\dir-column ... 525
\displayLilyMusic ... 378
\divisioMaior ... 338
\divisioMaxima ... 338
\divisioMinima ... 338
\dorian ... 16
\dotsDown ... 38
\dotsNeutral ... 38
\dotsUp ... 38
\doubleflat ... 543
\doublesharp ... 543
\downbow ... 99
\downbow ... 246
\downmordent ... 99
\downprall ... 99
\draw-circle ... 207, 538
\draw-line ... 207, 538
\drummode ... 149
\dynamic ... 106, 516
\dynamicDown ... 103
\dynamicNeutral ... 103
\dynamicUp ... 103
\easyHeadsOff ... 31
\easyHeadsOn ... 31
\epsfile ... 207, 539
\espressivo ... 99
\espressivo ... 102
\expandFullBarRests ... 51, 52
\eyeglasses ... 550
\f ... 101
\featherDurations ... 80
\fermata ... 99
\fermataMarkup ... 51, 52
\fermataMarkup ... 99
\ff ... 101
\fff ... 101
\ffff ... 101
\fffff ... 101
\fill-line ... 204, 526
\filled-box ... 207, 539
\finalis ... 338
\finger ... 179, 516
\flageolet ... 99
\flat ... 543
\flexa ... 346
\fontCaps ... 516
\fontsize ... 200, 517
\fp ... 101
\fraction ... 550
\frenchChords ... 317
\fret-diagram ... 260, 546
\fret-diagram-terse ... 262, 547
\fret-diagram-verbose ... 263, 548
\fromproperty ... 551
\funkHeads ... 33
\funkHeadsMinor ... 33
\general-align ... 203, 526
\germanChords ... 317
\glissando ... 114
\grace ... 92
\halfopen ... 99
\halign ... 202, 527
\harmonic ... 246
\harmonicsOff ... 246
\harmonicsOn ... 246
\harp-pedal ... 548
\hbracket ... 206, 539
\hcenter-in ... 528
\header ... 363
\hideKeySignature ... 298
\hideNotes ... 181
\hideSplitTiedTabNotes ... 254
\hideStaffSwitch ... 237
\hspace ... 528
\huge ... 177, 202, 517
\improvisationOff ... 35, 67
\improvisationOn ... 35, 67
\in ... 460
\inclinatum ... 340, 346
\include ... 371
\instrumentSwitch ... 169
\ionian ... 16
\italianChords ... 317
\italic ... 200, 517
\justified-lines ... 211, 555
\justify ... 205
\justify ... 529

\justify-field 529
\justify-string 530
\keepWithTag 374
\key 16, 33
\killCues 175
\label 369
\laissezVibrer 45
\large 177, 202, 517
\larger 200, 202, 517
\layout 363, 399
\left-align 202, 530
\left-brace 551
\left-column 531
\lheel 99
\line 531
\linea 340, 346
\lineprall 99
\locrian 16
\longa 37, 47
\longfermata 99
\lookup 551
\lower 203, 531
\ltoe 99
\lydian 16
\lyricmode 216, 219
\lyricsto 219
\magnify 200, 518
\major 16
\makeClusters 136
\marcato 99
\mark 90, 193
\markalphabet 551
\markletter 552
\markup 193, 197, 198
\markuplines 197, 211
\maxima 37, 47
\medium 518
\melisma 223
\melismaEnd 223
\mergeDifferentlyDottedOff 140
\mergeDifferentlyDottedOn 140
\mergeDifferentlyHeadedOff 140
\mergeDifferentlyHeadedOn 140
\mf 101
\midi 363
\minor 16
\mixolydian 16
\mm 460
\mordent 99
\mp 101
\musicglyph 91, 543
\name 443
\natural 544
\new 437
\noBeam 77
\noBreak 400
\noPageBreak 401
\noPageTurn 403
\normal-size-sub 518
\normal-size-super 519
\normal-text 519
\normalsize 177, 202, 519
\note 544
\note-by-number 544
\null 203, 552
\number 519
\numericTimeSignature 55
\octaveCheck 8
\on-the-fly 552
\once 453
\oneVoice 137
\open 99
\open 246
\oriscus 340, 346
\ottava 17
\override 451, 552
\override-lines 555
\overrideTimeSignatureSettings 55
\p 101
\pad-around 206, 531
\pad-markup 206, 532
\pad-to-box 206, 532
\pad-x 206, 532
\page-ref 369, 553
\pageBreak 401
\pageTurn 403
\paper 363, 391
\parallelMusic 147
\parenthesize 183, 540
\partcombine 143
\partial 61, 120, 121
\path 540
\pes 346
\phrasingSlurDashed 111
\phrasingSlurDashPattern 111
\phrasingSlurDotted 111
\phrasingSlurDown 111
\phrasingSlurHalfDashed 111
\phrasingSlurHalfSolid 111
\phrasingSlurNeutral 111
\phrasingSlurSolid 111
\phrasingSlurUp 111
\phrygian 16
\pitchedTrill 119
\portato 99
\postscript 207, 541
\powerChords 284
\pp 101
\ppp 101
\pppp 101
\ppppp 101
\prall 99
\pralldown 99
\prallmordent 99
\prallprall 99
\prallup 99
\predefinedFretboardsOff 278
\predefinedFretboardsOn 278
\property en mode \lyricmode 216
\pt 460
\put-adjacent 533
\quilisma 340, 346
\quoteDuring 170, 174
\raise 203, 533
\relative 2, 12, 236
\RemoveEmptyStaves 163, 165
\removeWithTag 374
\repeat 120
\repeat percent 129
\repeat tremolo 131

\repeat unfold ... 127
\repeatTie ... 44, 123
\rest ... 47
\reverseturn ... 99
\revert ... 452
\revertTimeSignatureSettings ... 56
\rfz ... 101
\rheel ... 99
\right-align ... 202, 533
\right-brace ... 553
\right-column ... 533
\rightHandFinger ... 280
\roman ... 520
\rotate ... 534
\rounded-box ... 206, 542
\rtoe ... 99
\sacredHarpHeads ... 33
\sacredHarpHeadsMinor ... 33
\sans ... 520
\scale ... 542
\scaleDurations ... 43, 63
\score ... 359, 363, 544
\segno ... 99
\semiflat ... 545
\semiGermanChords ... 317
\semisharp ... 546
\sesquiflat ... 546
\sesquisharp ... 546
\set ... 71, 450
\sf ... 101
\sff ... 101
\sfz ... 101
\sharp ... 546
\shiftOff ... 140
\shiftOn ... 140
\shiftOnn ... 140
\shiftOnnn ... 140
\shortfermata ... 99
\showKeySignature ... 298
\showStaffSwitch ... 237
\signumcongruentiae ... 99
\simple ... 520
\skip ... 49
\slashed-digit ... 553
\slurDashed ... 108
\slurDashPattern ... 109
\slurDotted ... 108
\slurDown ... 108
\slurHalfDashed ... 108
\slurHalfSolid ... 108
\slurNeutral ... 108
\slurSolid ... 108
\slurUp ... 109
\small ... 177, 202, 520
\smallCaps ... 521
\smaller ... 200, 202, 521
\snappizzicato ... 99
\sostenutoOff ... 239
\sostenutoOn ... 239
\southernHarmonyHeads ... 33
\southernHarmonyHeadsMinor ... 33
\sp ... 101
\spp ... 101
\staccatissimo ... 99
\staccato ... 99
\startGroup ... 188
\startStaff ... 158, 160
\startTrillSpan ... 118
\stemDown ... 184
\stemNeutral ... 184
\stemUp ... 184
\stencil ... 553
\stopGroup ... 188
\stopped ... 99
\stopStaff ... 158, 160, 163
\stopTrillSpan ... 118
\storePredefinedDiagram ... 271
\stropha ... 340, 346
\strut ... 553
\sub ... 201, 521
\super ... 201, 522
\sustainOff ... 239
\sustainOn ... 239
\tabChordRepetition ... 252
\tabFullNotation ... 251
\table-of-contents ... 371, 555
\tag ... 374
\taor ... 298
\teeny ... 177, 202, 522
\tempo ... 58
\tenuto ... 99
\text ... 522
\textLengthOff ... 52, 190
\textLengthOn ... 52, 190
\textSpannerDown ... 191
\textSpannerNeutral ... 191
\textSpannerUp ... 191
\thumb ... 99
\thumb ... 179
\tied-lyric ... 546
\tieDashed ... 45
\tieDotted ... 45
\tieDown ... 45
\tieNeutral ... 45
\tieSolid ... 45
\tieUp ... 45
\time ... 54, 71
\times ... 38, 63
\tiny ... 177, 202, 522
\tocItem ... 371
\translate ... 203, 534
\translate-scaled ... 203, 534
\transparent ... 553
\transpose ... 2, 3, 9, 12
\transposedCueDuring ... 175
\transposition ... 18, 170
\treCorde ... 239
\triangle ... 207, 542
\trill ... 99
\trill ... 118
\tupletDown ... 39
\tupletNeutral ... 39
\tupletUp ... 39
\turn ... 99
\tweak ... 453
\type ... 443
\typewriter ... 523
\unaCorda ... 239
\underline ... 200, 523
\unfoldRepeats ... 384

\unHideNotes ... 181
\unset ... 451
\upbow ... 99
\upbow ... 246
\upmordent ... 99
\upprall ... 99
\upright ... 523
\varcoda ... 99
\vcenter ... 535
\verbatim-file ... 554
\verylongfermata ... 99
\virga ... 340, 346
\virgula ... 338
\voiceFourStyle ... 139
\voiceNeutralStyle ... 139
\voiceOne ... 137
\voiceOne ... \voiceFour ... 137
\voiceOneStyle ... 139
\voiceThreeStyle ... 139
\voiceTwoStyle ... 139
\vspace ... 535
\walkerHeads ... 33
\walkerHeadsMinor ... 33
\whiteout ... 554
\with ... 441
\with-color ... 182, 554
\with-dimensions ... 554
\with-url ... 543
\woodwind-diagram ... 549
\wordwrap ... 205
\wordwrap ... 536
\wordwrap-field ... 535
\wordwrap-internal ... 555
\wordwrap-lines ... 211, 555
\wordwrap-string ... 536
\wordwrap-string-internal ... 555

|

| ... 89

~

~ ... 44

A

accepts ... 443
addChordShape ... 271
addInstrumentDefinition ... 169
addQuote ... 170
aeolian ... 16
afterGrace ... 93
aikenHeads ... 33
aikenHeadsMinor ... 33
alias ... 443
alignAboveContext ... 445
alignBelowContext ... 445
annotate-spacing ... 431
arpeggio ... 115
arpeggioArrowDown ... 115
arpeggioArrowUp ... 115
arpeggioBracket ... 115
arpeggioNormal ... 115
arpeggioParenthesis ... 115
arpeggioParenthesisDashed ... 115
arranger ... 365
arrow-head ... 207
aug ... 309
auto-first-page-number ... 395
autoBeaming ... 71
autoBeamOff ... 69
autoBeamOn ... 69
autochange ... 235

B

Balloon_engraver ... 185
balloonGrobText ... 185
balloonLengthOff ... 185
balloonLengthOn ... 185
balloonText ... 185
banjo-c-tuning ... 285
banjo-modal-tuning ... 285
banjo-open-d-tuning ... 285
banjo-open-dm-tuning ... 285
bar ... 81, 85
barCheckSynchronize ... 89
BarNumber ... 86
barNumberCheck ... 90
barNumberVisibility ... 86
bartype ... 85
base-shortest-duration ... 420
baseMoment ... 71
beamExceptions ... 71
beatStructure ... 71
bendAfter ... 113
binding-offset ... 394
blank-last-page-force ... 395
blank-page-force ... 395
bold ... 200
bookTitleMarkup ... 368
bottom-margin ... 391
box ... 206
bracket ... 106, 206, 239
breakable ... 70
breakbefore ... 365
breathe ... 112
breve ... 37, 47

C

cadenzaOff ... 61
cadenzaOn ... 61
center-align ... 202
center-column ... 204
change ... 234
check-consistency ... 396
chordChanges ... 314
chordmode ... 3, 12, 269
chordNameExceptions ... 316
chordNameLowercaseMinor ... 316
ChordNames ... 269
chordNameSeparator ... 316
chordNoteNamer ... 316
chordPrefixSpacer ... 317
chordRootNamer ... 316
circle ... 206

clef 12
color 182
column 204
combine 207
common-shortest-duration 420
Completion_heads_engraver 66
composer 365
compressFullBarRests 51, 52
consists 443
controlpitch 8
copyright 365
cr 102
crescHairpin 103
crescTextCresc 103
cross 29
cross-staff 237
cueDuring 174
currentBarNumber 86, 97

D

decr 102
dedication 365
default 20, 21
defaultBarType 85
defaultTimeSignature 55
denies 443
dim 309
dimHairpin 103
dimTextDecr 103
dimTextDecresc 103
dimTextDim 103
dodecaphonic 24
dorian 16
dotsDown 38
dotsNeutral 38
dotsUp 38
draw-circle 207
draw-line 207
drummode 149
DrumStaff 149
dynamic 106
dynamicDown 103
DynamicLineSpanner 103
dynamicNeutral 103
dynamicUp 103

E

easyHeadsOff 31
easyHeadsOn 31
epsfile 207
espressivo 102
evenFooterMarkup 368
evenHeaderMarkup 368
expandFullBarRests 51, 52

F

f 101
featherDurations 80
fermataMarkup 51, 52
ff 101
fff 101
ffff 101
fffff 101
fill-line 204
filled-box 207
finger 179
first-page-number 396
flag-style 237
followVoice 237
font-interface 178, 212
font-size 177, 178
fontsize 200
fontSize 177
forget 25
four-string-banjo 285
fp 101
fret-diagram 260
fret-diagram-interface 265
fret-diagram-terse 262
fret-diagram-verbose 263
FretBoards 268
funkHeads 33
funkHeadsMinor 33

G

general-align 203
glissando 114
grace 92
GregorianTranscriptionStaff 149
Grid_line_span_engraver 186
Grid_point_engraver 186
gridInterval 186
grow-direction 80

H

halign 202
hbracket 206
hideKeySignature 298
hideNotes 181
hideStaffSwitch 237
horizontal-shift 394
Horizontal_bracket_engraver 188
huge 177, 202

I

improvisationOff 35, 67
improvisationOn 35, 67
indent 168, 394, 424
inner-margin 394
instrument 365
instrumentSwitch 169
ionian 16
italic 200

J

justified-lines 211
justify 205

K

key 16, 33

killCues 175

L

laissezVibrer 45
large 177, 202
larger 200, 202
last-bottom-spacing 393
layout file 398
left-align 202
left-margin 394
length 237
line-width 394, 424
locrian 16
longa 37, 47
lower 203
ly:minimal-breaking 403
ly:optimal-breaking 402
ly:page-turn-breaking 402
lydian 16

M

m 309
magnify 200
magstep 177, 460
maj 309
major 16
major seven symbols 317
majorSevenSymbol 316
make-dynamic-script 106
make-pango-font-tree 214
makeClusters 136
mark 90, 193
markup 193, 197, 198
markup-markup-spacing 392
markup-system-spacing 392
markuplines 197, 211
max-systems-per-page 396
maxima 37, 47
measureLength 71, 97
measurePosition 61, 97
MensuralStaff 149
mergeDifferentlyDottedOff 140
mergeDifferentlyDottedOn 140
mergeDifferentlyHeadedOff 140
mergeDifferentlyHeadedOn 140
meter 365
mf 101
min-systems-per-page 396
minimumFret 251, 279
minimumPageTurnLength 402
minimumRepeatLengthForPageTurn 402
minor 16
mixed 239
mixolydian 16
modern 22
modern-cautionary 22
modern-voice 22
modern-voice-cautionary 23
mp 101
MultiMeasureRestText 51
musicglyph 91

N

name 443
neo-modern 23
neo-modern-cautionary 24
neo-modern-voice 24
neo-modern-voice-cautionary 24
no-reset 25
noBeam 77
normalsize 177, 202
Note_heads_engraver 66
null 203
numericTimeSignature 55

O

octaveCheck 8
oddFooterMarkup 368
oddHeaderMarkup 368
oneVoice 137
opus 365
ottava 17
outer-margin 394
outside-staff-horizontal-padding 418
outside-staff-padding 418
outside-staff-priority 418

P

p 101
pad-around 206
pad-markup 206
pad-to-box 206
pad-x 206
page-breaking-system-system-spacing 396
page-count 396
page-spacing-weight 396
paper-height 391
paper-width 394
parallelMusic 147
parenthesize 183
partcombine 143
partial 61
pedalSustainStyle 239
percent 129
phrasingSlurDashed 111
phrasingSlurDashPattern 111
phrasingSlurDotted 111
phrasingSlurDown 111
phrasingSlurHalfDashed 111
phrasingSlurHalfSolid 111
phrasingSlurNeutral 111
phrasingSlurSolid 111
phrasingSlurUp 111
phrygian 16
piano accidentals 23
piano-cautionary 23
PianoStaff 233, 235
piece 365
pipeSymbol 89
Pitch_squash_engraver 67
pitchedTrill 119
poet 365
postscript 207
powerChords 284

pp 101
ppp 101
pppp 101
ppppp 101
predefinedFretboardsOff 278
predefinedFretboardsOn 278
print-all-headers 367, 396
print-first-page-number 396
print-page-number 396

Q

quotedCueEventTypes 172
quotedEventTypes 172
quoteDuring 170, 174

R

r 47
R 50
ragged-bottom 396
ragged-last 396, 424
ragged-last-bottom 396
ragged-right 397
ragged-right 424
raise 203
relative 2, 12, 236
repeatCommands 125
repeatTie 44
rest 47
rfz 101
rgb-color 182
RhythmicStaff 149
right-align 202
right-margin 394
rightHandFinger 280
rounded-box 206

S

s 49
sacredHarpHeads 33
sacredHarpHeadsMinor 33
scaleDurations 43, 63
score-markup-spacing 392
score-system-spacing 392
scoreTitleMarkup 368
set 71
set-accidental-style 20
set-octavation 17
sf 101
sff 101
sfz 101
shiftOff 140
shiftOn 140
shiftOnn 140
shiftOnnn 140
short-indent 168, 394
show-available-fonts 214
showFirstLength 379
showKeySignature 298
showLastLength 379
showStaffSwitch 237
skip 49
skipTypesetting 379
slurDashed 108
slurDashPattern 109
slurDotted 108
slurDown 108
slurHalfDashed 108
slurHalfSolid 108
slurNeutral 108
slurSolid 108
slurUp 109
small 177, 202
smaller 200, 202
sostenutoOff 239
sostenutoOn 239
southernHarmonyHeads 33
southernHarmonyHeadsMinor 33
sp 101
spacing 420
spp 101
staff-padding 234
Staff.midiInstrument 381
Staff_symbol_engraver 163
start-repeat 125
startGroup 188
startStaff 158, 160
startTrillSpan 118
Stem 237
stem-spacing-correction 420
stemDown 184
stemLeftBeamCount 78
stemNeutral 184
stemRightBeamCount 78
stemUp 184
stopGroup 188
stopStaff 158, 160, 163
stopTrillSpan 118
storePredefinedDiagram 271
stringTunings 268
StringTunings 257
sub 201
subsubtitle 365
subtitle 365
suggestAccidentals 334
super 201
sus 312
sustainOff 239
sustainOn 239
system-count 397
system-separator-markup 397
system-system-spacing 392
systems-per-page 397

T

TabStaff 149, 250
TabVoice 250
tagline 365
taor 298
teaching 25
teeny 177, 202
tempo 58
text 239
textLengthOff 190
textLengthOn 52, 190

textLenthOff 52
textSpannerDown 191
textSpannerNeutral 191
textSpannerUp 191
thumb 179
tieDashed 45
tieDotted 45
tieDown 45
tieNeutral 45
tieSolid 45
tieUp 45
time 54, 71
times 38, 63
timeSignatureFraction 63
tiny 177, 202
title 365
top-margin 391
top-markup-spacing 393
top-system-spacing 393
translate 203
translate-scaled 203
transpose 2, 3, 9, 12
transposedCueDuring 175
transposition 18, 170
treCorde 239
tremolo 131
tremoloFlags 132
triangle 207
trill 118
tupletDown 39
tupletNeutral 39
TupletNumber 40
tupletNumberFormatFunction 39
tupletSpannerDuration 39
tupletUp 39
two-sided 397
type 443

U

unaCorda 239
underline 200
unfold 127
unHideNotes 181

V

VaticanaStaff 149
voice 20, 22
Voice 137
voiceOne 137

W

walkerHeads 33
walkerHeadsMinor 33
whichBar 85
with-color 182
wordwrap 205
wordwrap-lines 211

X

x11-color 182, 183

Annexe F Index de LilyPond

En plus des commandes et mots réservés de LilyPond, cet index recense les termes musicaux qui s'y rapportent. Tout comme dans l'index des commandes, chaque lien est constitué de deux parties : la première pointe directement là où la commande ou le terme apparaît, la seconde au début de la section où la commande ou le terme est abordé.

!

! 5

'

' 1

,

, ... 1

-

- 99

.

. 37

/

/ 312
/+ 312

:

: 132

<

< 133
<...> 133

=

= 8

>

> 133

?

? 5

[

[..... 77

]

] 77

^

^ 311

_

_ 217, 221

\

\! 102
\(..... 110
\) 110
\< 102
\> 102
\\ 138
\abs-fontsize 515
\accent 99
\accepts 443, 444, 445
\addChordShape 271
\addInstrumentDefinition 169
\addlyrics 218
\addlyrics 219
\addQuote 170
\aeolian 16
\afterGrace 93
\aikenHeads 33
\aikenHeadsMinor 33
\alias 443
\allowPageTurn 403
\alternative 120
\arpeggio 115
\arpeggioArrowDown 115
\arpeggioArrowUp 115
\arpeggioBracket 115
\arpeggioNormal 115
\arpeggioParenthesis 115
\arpeggioParenthesisDashed 115
\arrow-head 207, 537
\ascendens 340, 346
\auctum 340, 346
\augmentum 346
\autoBeamOff 69
\autoBeamOff et \partcombine 70
\autoBeamOn 69
\autochange 235
\backslashed-digit 550
\balloonGrobText 185
\balloonLengthOff 185
\balloonLengthOn 185
\balloonText 185
\bar 81, 85
\barNumberCheck 90
\beam 537
\bendAfter 113
\bold 200, 515

\book ... 360, 363
\bookOutputName ... 362
\bookOutputSuffix ... 362
\bookpart ... 361, 363, 401
\box ... 206, 515
\bracket ... 106, 206, 538
\break ... 400
\breathe ... 112
\breve ... 37, 47
\cadenzaOff ... 61
\cadenzaOn ... 61
\caesura ... 338
\caps ... 516
\cavum ... 340, 346
\center-align ... 202, 524
\center-column ... 204, 524
\change ... 234
\char ... 550
\chordmode ... 3, 12, 269
\chords ... 314
\circle ... 206, 538
\clef ... 12
\cm ... 460
\coda ... 99
\column ... 204, 524
\column-lines ... 554
\combine ... 207, 525
\compressFullBarRests ... 51, 52
\concat ... 525
\consists ... 443
\context ... 437
\cr ... 102
\crescHairpin ... 103
\crescTextCresc ... 103
\cueDuring ... 174
\customTabClef ... 543
\decr ... 102
\defaultTimeSignature ... 55
\deminutum ... 340, 346
\denies ... 443, 444, 445
\descendens ... 340, 346
\dimHairpin ... 103
\dimTextDecr ... 103
\dimTextDecresc ... 103
\dimTextDim ... 103
\dir-column ... 525
\displayLilyMusic ... 378
\divisioMaior ... 338
\divisioMaxima ... 338
\divisioMinima ... 338
\dorian ... 16
\dotsDown ... 38
\dotsNeutral ... 38
\dotsUp ... 38
\doubleflat ... 543
\doublesharp ... 543
\downbow ... 99
\downbow ... 246
\downmordent ... 99
\downprall ... 99
\draw-circle ... 207, 538
\draw-line ... 207, 538
\drummode ... 149
\dynamic ... 106, 516
\dynamicDown ... 103
\dynamicNeutral ... 103
\dynamicUp ... 103
\easyHeadsOff ... 31
\easyHeadsOn ... 31
\epsfile ... 207, 539
\espressivo ... 99
\espressivo ... 102
\expandFullBarRests ... 51, 52
\eyeglasses ... 550
\f ... 101
\featherDurations ... 80
\fermata ... 99
\fermataMarkup ... 51, 52
\fermataMarkup ... 99
\ff ... 101
\fff ... 101
\ffff ... 101
\fffff ... 101
\fill-line ... 204, 526
\filled-box ... 207, 539
\finalis ... 338
\finger ... 179, 516
\flageolet ... 99
\flat ... 543
\flexa ... 346
\fontCaps ... 516
\fontsize ... 200, 517
\fp ... 101
\fraction ... 550
\frenchChords ... 317
\fret-diagram ... 260, 546
\fret-diagram-terse ... 262, 547
\fret-diagram-verbose ... 263, 548
\fromproperty ... 551
\funkHeads ... 33
\funkHeadsMinor ... 33
\general-align ... 203, 526
\germanChords ... 317
\glissando ... 114
\grace ... 92
\halfopen ... 99
\halign ... 202, 527
\harmonic ... 246
\harmonicsOff ... 246
\harmonicsOn ... 246
\harp-pedal ... 548
\hbracket ... 206, 539
\hcenter-in ... 528
\header ... 363
\hideKeySignature ... 298
\hideNotes ... 181
\hideSplitTiedTabNotes ... 254
\hideStaffSwitch ... 237
\hspace ... 528
\huge ... 177, 202, 517
\improvisationOff ... 35, 67
\improvisationOn ... 35, 67
\in ... 460
\inclinatum ... 340, 346
\include ... 371
\instrumentSwitch ... 169
\ionian ... 16
\italianChords ... 317
\italic ... 200, 517
\justified-lines ... 211, 555

\justify 205
\justify 529
\justify-field 529
\justify-string 530
\keepWithTag 374
\key 16, 33
\killCues 175
\label 369
\laissezVibrer 45
\large 177, 202, 517
\larger 200, 202, 517
\layout 363, 399
\left-align 202, 530
\left-brace 551
\left-column 531
\lheel 99
\line 531
\linea 340, 346
\lineprall 99
\locrian 16
\longa 37, 47
\longfermata 99
\lookup 551
\lower 203, 531
\ltoe 99
\lydian 16
\lyricmode 216, 219
\lyricsto 219
\magnify 200, 518
\major 16
\makeClusters 136
\marcato 99
\mark 90, 193
\markalphabet 551
\markletter 552
\markup 193, 197, 198
\markuplines 197, 211
\maxima 37, 47
\medium 518
\melisma 223
\melismaEnd 223
\mergeDifferentlyDottedOff 140
\mergeDifferentlyDottedOn 140
\mergeDifferentlyHeadedOff 140
\mergeDifferentlyHeadedOn 140
\mf 101
\midi 363
\minor 16
\mixolydian 16
\mm 460
\mordent 99
\mp 101
\musicglyph 91, 543
\name 443
\natural 544
\new 437
\noBeam 77
\noBreak 400
\noPageBreak 401
\noPageTurn 403
\normal-size-sub 518
\normal-size-super 519
\normal-text 519
\normalsize 177, 202, 519
\note 544
\note-by-number 544
\null 203, 552
\number 519
\numericTimeSignature 55
\octaveCheck 8
\on-the-fly 552
\once 451
\once 453
\oneVoice 137
\open 99
\open 246
\oriscus 340, 346
\ottava 17
\override 451, 552
\override-lines 555
\overrideTimeSignatureSettings 55
\p 101
\pad-around 206, 531
\pad-markup 206, 532
\pad-to-box 206, 532
\pad-x 206, 532
\page-ref 369, 553
\pageBreak 401
\pageTurn 403
\paper 363
\paper 368
\paper 391
\parallelMusic 147
\parenthesize 183, 540
\partcombine 143
\partcombine et \autoBeamOff 70
\partial 61, 120, 121
\path 540
\pes 346
\phrasingSlurDashed 111
\phrasingSlurDashPattern 111
\phrasingSlurDotted 111
\phrasingSlurDown 111
\phrasingSlurHalfDashed 111
\phrasingSlurHalfSolid 111
\phrasingSlurNeutral 111
\phrasingSlurSolid 111
\phrasingSlurUp 111
\phrygian 16
\pitchedTrill 119
\portato 99
\postscript 207, 541
\powerChords 284
\pp 101
\ppp 101
\pppp 101
\ppppp 101
\prall 99
\pralldown 99
\prallmordent 99
\prallprall 99
\prallup 99
\predefinedFretboardsOff 278
\predefinedFretboardsOn 278
\property en mode \lyricmode 216
\pt 460
\put-adjacent 533
\quilisma 340, 346
\quoteDuring 170, 174
\raise 203, 533

\relative ... 2, 12, 236
\RemoveEmptyStaves ... 163, 165
\removeWithTag ... 374
\repeat ... 120
\repeat percent ... 129
\repeat tremolo ... 131
\repeat unfold ... 127
\repeatTie ... 44, 123
\rest ... 47
\reverseturn ... 99
\revert ... 452
\revertTimeSignatureSettings ... 56
\rfz ... 101
\rheel ... 99
\right-align ... 202, 533
\right-brace ... 553
\right-column ... 533
\rightHandFinger ... 280
\roman ... 520
\rotate ... 534
\rounded-box ... 206, 542
\rtoe ... 99
\sacredHarpHeads ... 33
\sacredHarpHeadsMinor ... 33
\sans ... 520
\scale ... 542
\scaleDurations ... 43, 63
\score ... 359, 363, 544
\segno ... 99
\semiflat ... 545
\semiGermanChords ... 317
\semisharp ... 546
\sesquiflat ... 546
\sesquisharp ... 546
\set ... 71, 450
\sf ... 101
\sff ... 101
\sfz ... 101
\sharp ... 546
\shiftOff ... 140
\shiftOn ... 140
\shiftOnn ... 140
\shiftOnnn ... 140
\shortfermata ... 99
\showKeySignature ... 298
\showStaffSwitch ... 237
\signumcongruentiae ... 99
\simple ... 520
\skip ... 49
\slashed-digit ... 553
\slurDashed ... 108
\slurDashPattern ... 109
\slurDotted ... 108
\slurDown ... 108
\slurHalfDashed ... 108
\slurHalfSolid ... 108
\slurNeutral ... 108
\slurSolid ... 108
\slurUp ... 109
\small ... 177, 202, 520
\smallCaps ... 521
\smaller ... 200, 202, 521
\snappizzicato ... 99
\sostenutoOff ... 239
\sostenutoOn ... 239
\southernHarmonyHeads ... 33
\southernHarmonyHeadsMinor ... 33
\sp ... 101
\spp ... 101
\staccatissimo ... 99
\staccato ... 99
\startGroup ... 188
\startStaff ... 158, 160
\startTrillSpan ... 118
\stemDown ... 184
\stemNeutral ... 184
\stemUp ... 184
\stencil ... 553
\stopGroup ... 188
\stopped ... 99
\stopStaff ... 158, 160, 163
\stopTrillSpan ... 118
\storePredefinedDiagram ... 271
\stropha ... 340, 346
\strut ... 553
\sub ... 201, 521
\super ... 201, 522
\sustainOff ... 239
\sustainOn ... 239
\tabChordRepetition ... 252
\tabFullNotation ... 251
\table-of-contents ... 371, 555
\tag ... 374
\taor ... 298
\teeny ... 177, 202, 522
\tempo ... 58
\tenuto ... 99
\text ... 522
\textLengthOff ... 52, 190
\textLengthOn ... 52, 190
\textSpannerDown ... 191
\textSpannerNeutral ... 191
\textSpannerUp ... 191
\thumb ... 99
\thumb ... 179
\tied-lyric ... 546
\tieDashed ... 45
\tieDotted ... 45
\tieDown ... 45
\tieNeutral ... 45
\tieSolid ... 45
\tieUp ... 45
\time ... 54, 71
\times ... 38, 63
\tiny ... 177, 202, 522
\tocItem ... 371
\translate ... 203, 534
\translate-scaled ... 203, 534
\transparent ... 553
\transpose ... 2, 3, 9, 12
\transposedCueDuring ... 175
\transposition ... 18, 170
\treCorde ... 239
\triangle ... 207, 542
\trill ... 99
\trill ... 118
\tupletDown ... 39
\tupletNeutral ... 39
\tupletUp ... 39
\turn ... 99

\tweak ... 453
\type ... 443
\typewriter ... 523
\unaCorda ... 239
\underline ... 200, 523
\unfoldRepeats ... 384
\unHideNotes ... 181
\unset ... 451
\upbow ... 99
\upbow ... 246
\upmordent ... 99
\upprall ... 99
\upright ... 523
\varcoda ... 99
\vcenter ... 535
\verbatim-file ... 554
\verylongfermata ... 99
\virga ... 340, 346
\virgula ... 338
\voiceFourStyle ... 139
\voiceNeutralStyle ... 139
\voiceOne ... 137
\voiceOne ... \voiceFour ... 137
\voiceOneStyle ... 139
\voiceThreeStyle ... 139
\voiceTwoStyle ... 139
\vspace ... 535
\walkerHeads ... 33
\walkerHeadsMinor ... 33
\whiteout ... 554
\with ... 441
\with-color ... 182, 554
\with-dimensions ... 554
\with-url ... 543
\woodwind-diagram ... 549
\wordwrap ... 205
\wordwrap ... 536
\wordwrap-field ... 535
\wordwrap-internal ... 555
\wordwrap-lines ... 211, 555
\wordwrap-string ... 536
\wordwrap-string-internal ... 555

Callback functions

Callback functions ... 473

Displaying music expressions

Displaying music expressions ... 455

Interfaces for programmers

Interfaces for programmers ... 472

Literature list

Literature list ... 316

Markup construction in Scheme

Markup construction in Scheme ... 107, 108

Music functions

Music functions ... 479, 480

Optical spacing

Optical spacing ... 420

Running a function on all layout objects

Running a function on all layout objects ... 438

Scheme tutorial

Scheme tutorial ... 435

|

| ... 89

~

~ ... 44

1

15ma ... 17

8

8va ... 17
8ve ... 17

A

a due ... 143
absolues, hauteurs ... 1
absolues, octaves ... 1
accent ... 99, 555
accentus ... 555
accepts ... 443
acciaccatura ... 587
acciaccature ... 92
acciaccature, multi-note ... 96
Accidental, musica ficta ... 334
accidentals ... 334, 337
accidentel, quart de ton ... 7
accidentelles automatiques ... 20
accidentelles, altérations ... 20
accolade verticale ... 151
accolades, taille ... 212
accord arpégé ... 115
accord et mode relatif ... 134
accordéon, symboles de registre ... 240
accordéon, tirettes ... 240
accords ... 133
accords et altérations ... 26
accords et couleur ... 183
accords et liaisons de tenue ... 44
accords et octave relative ... 4
accords, doigtés ... 179
accords, répartition sur plusieurs portées avec \autochange ... 236
addChordShape ... 271
addChordShape ... 587

adding a white background to text 554
adding custom fret diagrams 270
addInstrumentDefinition 169
addInstrumentDefinition 587
additions, in chords 311
addQuote 170
addQuote 587
adjusting staff symbol 460
aeolian 16
afterGrace 93
afterGrace 587
afterGraceFraction 560
Aiken, tête de note 33
aikenHeads 33
aikenHeadsMinor 33
ajout de texte 189
al niente 104
alias 443
align to objects 475
alignAboveContext 445
alignBelowContext 445
alignement des markups 202
alignement du numéro de mesure 88
alignement du texte 202
alignement du texte, commandes 206
alignement et cadence 97
alignement horizontal du texte 202
alignement vertical du texte 203
alist 558
allowPageTurn 587
altération 5
altération de précaution 5
altération de précaution style *modern voice* 23
altération de précaution, style *modern* 22
altération entre parenthèses 5
altération et liaison de tenue 5
altération, style 20
altération, style *dodecaphonic* 24
altération, style *modern* 22
altération, style *modern cautionary* 22
altération, style *no reset* 25
altération, style par défaut 20
altération, trilles avec hauteur explicite 119
altérations et accords 26
altérations et notes simultanées 26
altérations multivoix 22, 24
altérations style *piano* 23
altérations style *piano cautionary* 23
altérations, style *default* 21
altérations, style *forget* 25
altérations, style *modern* 22
altérations, style *modern-cautionary* 22
altérations, style *neo-modern* 23
altérations, style *teaching* 25
altérations, style *voice* 22
altered chords 310
alternative et liaison de prolongation 123
alternative et texte 126
alternative, reprise 120
Amazing Grace bagpipe example 298
ambitus 27
amplitude 27
anacrouse 61
anacrouse et reprise 121
analyse musicologique 188
angled hairpins 471
annotate-spacing 431
applyContext 587
applyMusic 587
applyOutput 587
appoggiatura 587
appoggiature 92
Arabic key signatures 353
Arabic music 352
Arabic music example 356
Arabic music template 356
Arabic note names 352
Arabic semi-flat symbol 353
Arabic time signatures 354
armure 5, 16
arpège 115
arpège, style parenthèse 118
arpège, symboles spéciaux 115
arpeggio 115
arpeggioArrowDown 115
arpeggioArrowUp 115
arpeggioBracket 115
arpeggioNormal 115
arpeggioParenthesis 115
arpeggioParenthesisDashed 115
arranger 365
arrow-head 207
articulation, liaison 108
articulation-event 172
articulations 99, 338
artificiels, harmoniques 247
assertBeamQuant 587
assertBeamSlope 587
association list 558
aug 309
auto-first-page-number 395
autobeam 71
autoBeaming 71
autoBeamOff 69
autoBeamOn 69
autochange 235
autochange 587
autochange et mode relatif 236
automatic chord diagrams 277
automatic fret diagrams 277
automaticBars 470
Automatique, changement de portée 235
automatiques, altérations accidentelles 20

B

backslashed digits 550
bagpipe 298
bagpipe example 298
Balloon_engraver 185
balloonGrobText 185
balloonGrobText 587
balloonLengthOff 185
balloonLengthOn 185
balloonText 185
balloonText 587
banjo tablature 248
banjo tablatures 285
banjo tunings 285

banjo-c-tuning 285
banjo-modal-tuning 285
banjo-open-d-tuning 285
banjo-open-dm-tuning 285
bar 81, 85
bar 587
bar lines, suppressing 470
barCheckSynchronize 89
BarNumber 86
barNumberCheck 90
barNumberCheck 587
barNumberVisibility 86
baroque, ornementation 99
barre de mesure invisible 81
barre de mesure manuelle 81
barre de mesure par défaut, modification du style 85
barre finale 81
barre indications 259
barres de mesure 81
barres de mesure, supression 61
barres de mesure, symboles au dessus de 193
barres de reprise 81
Bartók pizzicato 247
bartype 85
base-shortest-duration 420
baseMoment 71
bass note, for chords 312
Bass, figured 320
Bass, thorough 320
Basso continuo 320
battements par minute 58
batterie, portée 149
beamExceptions 71
beatStructure 71
bécarre 5
bémol 5
bendAfter 113
bendAfter 587
Bézier control points 478
Bézier curves 478
binding gutter 397
binding-offset 394
bisbigliando 244
Bison 560
blank-last-page-force 395
blank-page-force 395
blocs de texte 189
BNF 560
bold 200
bookOutputName 587
bookOutputSuffix 587
bookTitleMarkup 368
bottom-margin 391
bouché 99
box 206
bracket 106, 206, 239
break-align-symbols 475
break-visibility 467
breakable 70
breakbefore 365
breaking lines 399
breaking pages 424
breathe 112
breathe 588
breve 37, 47
broderies 92
bulle 185

C

cadence 61, 97
cadence et alignement 97
cadence et ligature 62
cadence et saut de ligne 62
cadence et saut de page 62
cadenza 97
cadenzaOff 61
cadenzaOn 61
cadrillage temporel 186
callback 558
calques 138
capo 263
caractères réservés, impression 199
caractères spéciaux en mode markup 199
caratères, taille 200
center-align 202
center-column 204
centering a column of text 524
centrage des nuances entre deux portées de piano 234
centrage du texte sur la page 204
césure 112
change 234
changement d'instrument 169
changement de fonte 200
changement de nom d'instrument 168
Changement de portée automatique 235
changement de portée forcé 234
changements de portée 237
changements de portée manuels 234
changing direction of text columns 525
changing properties 450
check-consistency 396
chevrons 133
chiffrage de mesure 54
chiffre indicateur de mesure 54
chœur, système 151
choral score 223
chorale, clef de tenor 13
chord chords 308
chord diagrams 259, 268
chord diagrams, automatic 277
chord inversions 312
chord mode 308
chord names 308, 313
Chord names in MIDI 383
chord names with fret diagrams 269
chord quality 309
chord shapes for fretted instruments 271
chord steps, altering 311
chord, modifying one note in 454
chordChanges 314
chordmode 3, 12, 269
chordNameExceptions 316
chordNameLowercaseMinor 316
ChordNames 269
chordNameSeparator 316
chordNoteNamer 316
chordPrefixSpacer 317

chordRootNamer ... 316
chords ... 313
chords, jazz ... 315
chords, power ... 284
chords, suppressing repeated ... 314
Christian Harmony, tête de note ... 33
chutes ... 113
circle ... 206
circling text ... 538
circulus ... 555
citation ... 170
citation, fin ... 175
clavier, portées pour instrument à ... 233
claviers, centrage des nuances ... 234
clé d'ut ... 12
clé de fa ... 12
clé de sol ... 12
clef ... 5
clef ... 12
clef ... 588
clef et transposition ... 13
clef, moderntab ... 258
clef, percussion ... 286
clef, tab ... 258
clef, visibility following explicit change ... 469
clefs ... 330, 337
clefs, visibility of octavation ... 470
clés, portées pour instrument à ... 233
closure ... 558
cluster ... 136
coda ... 91, 99, 555
coda sur une barre de mesure ... 193
collision de notes ... 140
collision et numéro de mesure ... 89
collisions ... 140
colonnes de texte ... 204
color ... 182
coloration d'objets ... 182
coloration de notes ... 182
colorier des voix ... 139
coloring objects ... 466
coloring text ... 554
colorisation d'objets ... 182
colorisation de notes ... 182
Colors, list of ... 495
column ... 204
combine ... 207
comma intervals ... 357
commandes d'alignement du texte ... 206
commentaire textuel ... 198
common-shortest-duration ... 420
Completion_heads_engraver ... 66
composer ... 365
composite, métrique ... 63
compressFullBarRests ... 51, 52
concatenating text ... 525
condenser les silences ... 54
consists ... 443
Context, creating ... 437
contexts, defining new ... 443
contexts, keeping alive ... 438
contexts, layout order ... 460
contexts, lifetime ... 438
contexts, nested ... 445
control points, tweaking ... 455
contrôle des hauteurs ... 8
controlling general text alignment ... 526
controlpitch ... 8
copyright ... 365
copyright ... 368
copyright sign ... 378
corde à vide, indication ... 246
cordes d'orchestre ... 245
cordes, écriture pour ... 245
couches ... 138
couleur rgb ... 182
couleur rvb ... 182
couleur, notes d'un accord ... 183
couleurs ... 182
couleurs x11 ... 182
coup d'archet ... 99
courbes ... 108
cr ... 102
creating contexts ... 438
creating empty text objects ... 552
creating horizontal spaces in text ... 528
creating text fractions ... 550
creating vertical spaces in text ... 535, 553
crescendo ... 102
crescHairpin ... 103
crescTextCresc ... 103
crochet de nolet, positionnement ... 39
crochet de regroupement de notes ... 188
crochet vertical ... 151
crochets ... 188
crochets de phrasé ... 188
croix, tête de note ... 29
cross ... 29
cross-staff ... 237
cross-staff tremolo ... 133
cueDuring ... 174
cueDuring ... 588
currentBarNumber ... 86, 97
custodes ... 329
custom fret diagrams ... 259
custom fret diagrams, adding ... 270
customized fret diagram ... 265
customizing chord names ... 315
custos ... 329

D

D.S al Fine ... 91
dampened notes on fretted instruments ... 282
deadNote ... 588
début de portée ... 158
décélage de voix ... 140
décoration du texte ... 206
decr ... 102
decrescendo ... 102
dedication ... 365
default ... 20, 21
default, style d'altérations ... 21
defaultBarType ... 85
defaultNoteHeads ... 588
defaultTimeSignature ... 55
défaut, durée ... 37
délimitation, systèmes imbriqués ... 155
demi-bémol ... 8
demi-bémols ... 5

demi-dièse ... 8
demi-dièses ... 5
denies ... 443
Devnull, contexte ... 225
diagram, fret, customized ... 265
diagrams, chord for fretted instruments ... 259
diagrams, fret ... 259
diagrams, fret, transposing ... 269
dièse ... 5
dim ... 309
dimHairpin ... 103
diminuendo ... 102
dimTextDecr ... 103
dimTextDecresc ... 103
dimTextDim ... 103
displayLilyMusic ... 588
displayMusic ... 588
disponibilité des fontes ... 214
distance between staves ... 407
distance entre deux portées de piano ... 237
distances, absolute ... 460
distances, scaled ... 460
divisio ... 338
division de notes ... 66
divisiones ... 338
dodecaphonic ... 24
dodecaphonic, style d'altérations ... 24
doigté ... 179
doigtés des accords ... 179
doigtés et silences multimesure ... 54
dorian ... 16
dorien ... 16
dotsDown ... 38
dotsNeutral ... 38
dotsUp ... 38
double barre ... 81
double bémol ... 5
double dièse ... 5
double pause ... 47
double point ... 37
doublement pointée, note ... 37
downbow ... 555
downmordent ... 555
downprall ... 555
draw-circle ... 207
draw-line ... 207
drawing beams within text ... 537
drawing boxes with rounded corners ... 539
drawing boxes with rounded corners around text ... 542
drawing circles within text ... 538
drawing lines within text ... 538
drawing paths ... 540
drawing solid boxes within text ... 539
drawing staff symbol ... 460
drawing triangles within text ... 542
drummode ... 149
drums ... 286, 288
DrumStaff ... 149
durée automatique des syllabes ... 220
durée d'un silence ... 47
durée par défaut ... 37
durées ... 37
durées, échelonnement ... 43
dynamic ... 106
dynamic-event ... 172
dynamicDown ... 103
DynamicLineSpanner ... 103
dynamicNeutral ... 103
dynamics, vertical positioning ... 103
dynamicUp ... 103

E

easy play, têtes de note ... 31
easyHeadsOff ... 31
easyHeadsOn ... 31
échelonnement de musique ... 43
échelonnement des durées ... 43
écrire la musique en parallèle ... 147
embedding graphic objects ... 206
encadrement du texte ... 206
encapsulated postscript output ... 379
enclosing text in a box with rounded corners ... 542
enclosing text within a box ... 515
endSpanners ... 588
engravers, including in contexts ... 443
éolien ... 16
EPS output ... 379
epsfile ... 207
espacement autour du texte ... 206
espacement des paroles ... 226
espaces, dans les paroles ... 216, 217
espressivo ... 99
espressivo ... 102
espressivo ... 555
étiquette ... 198
Étiquette de texte ... 189
étiquette et silence multimesure ... 51
étiquette textuelle ... 198
evenFooterMarkup ... 368
evenHeaderMarkup ... 368
exceptions, chord names ... 317
expandFullBarRests ... 51, 52
expansion de reprise ... 127
explicitClefVisibility ... 469
explicitKeySignatureVisibility ... 469
exposant ... 201
expressions markup ... 198
extended chords ... 310
extenseur ... 191, 224
extenseur, mise en forme ... 191

F

f ... 101
fantômes, notes ... 183
featherDurations ... 80, 588
fermata ... 555
fermataMarkup ... 51, 52
Feta font ... 496
ff ... 101
fff ... 101
ffff ... 101
fffff ... 101
Figured bass ... 320
figured bass alignment ... 325
figured bass extender lines ... 323
fill-line ... 204

`filled-box` ... 207
fin alternative ... 120
fin de citation ... 175
fin de portée ... 158
fin de réplique ... 175
finalis ... 338
`finger` ... 179
fingering vs. string numbers ... 249
fingerings, adding to fret diagrams ... 279
fingerings, right hand for fretted instruments ... 280
`first-page-number` ... 396
`flag-style` ... 237
flageolet ... 99, 555
flags ... 332
flat, double ... 5
Flex ... 559
fois, première ... 120
`followVoice` ... 237
font ... 558
font size, setting ... 398
Font, Feta ... 496
`font-interface` ... 178, 212
`font-size` ... 177, 178
fonte, changement ... 200
fontes disponibles ... 214
fontes et markup ... 212
fontes, choix par défaut ... 214
fontes, familles ... 201
fontes, généralités ... 212
`fontsize` ... 200
`fontSize` ... 177
footer ... 368
`forget` ... 25
forget, style d'altérations ... 25
format de repère ... 91
formatage du numéro de mesure ... 87
four bar music ... 399
`four-string-banjo` ... 285
`fp` ... 101
Frenched scores ... 163
fret ... 251
fret diagram, customized ... 265
fret diagrams ... 259, 268
fret diagrams with chord names ... 269
fret diagrams, adding custom ... 270
fret diagrams, adding fingerings ... 279
fret diagrams, automatic ... 277
fret diagrams, custom ... 259
fret diagrams, transposing ... 269
fret diagrams, ukulele ... 268
`fret-diagram` ... 260
fret-diagram markup ... 260
`fret-diagram-interface` ... 265
`fret-diagram-terse` ... 262
fret-diagram-terse markup ... 262
`fret-diagram-verbose` ... 263
fret-diagram-verbose markup ... 263
`FretBoards` ... 268
fretted instruments, chord shapes ... 271
fretted instruments, dampened notes ... 282
fretted instruments, harmonics ... 282
fretted instruments, indicating position and barring ... 282
fretted instruments, predefined string tunings ... 257
fretted instruments, right hand fingerings ... 280
Funk, tête de note ... 33
`funkHeads` ... 33
`funkHeadsMinor` ... 33
fusion de notes ... 140

G

`general-align` ... 203
`glissando` ... 114
global variable ... 560
glyph ... 558
glyphes musicales ... 91
`grace` ... 92
`grace` ... 588
grace notes ... 298
grammar for LilyPond ... 560
graphical object interfaces ... 559
graphical objects ... 559
graphiques, intégration ... 207
graphiques, tracé d'objets ... 206
graphisme dans la notation ... 207
graphisme, tracé ... 206
gras ... 200
Gregorian square neumes ligatures ... 340
`GregorianTranscriptionStaff` ... 149
`Grid_line_span_engraver` ... 186
`Grid_point_engraver` ... 186
`gridInterval` ... 186
grob ... 447, 559
grob properties ... 451
grob-interface ... 559
grobs, overwriting ... 466
grobs, visibility of ... 466
groupement de mesures ... 75
groupement de pulsations ... 75
groupements de note manuels ... 77
`grow-direction` ... 80
grupetto ... 99
guillemets en mode markup ... 199
guillemets, dans les paroles ... 216, 217
guitar chord charts ... 67
guitar strumming rhythms, showing ... 67
guitar tablature ... 248
guitare, tête de note ... 29
gutter ... 397

H

hairpins, angled ... 471
Hal Leonard ... 31
halfopen ... 555
`halign` ... 202
hampe barrée ... 94
hampe, enjambement portées ... 237
hampes ... 184
hampes invisibles ... 184
harmonic indications in tablature notation ... 255
Harmonica Sacra, tête de note ... 33
`harmonicNote` ... 588
harmonics on fretted instruments ... 282
`harmonicsOn` ... 588
harmonique, tête de note ... 29
harmoniques artificiels ... 247
harmoniques naturels ... 246

harpe sacrée, tête de note ... 33
harpe, diagrammes de pédales ... 244
harpe, pédales ... 244
harpes ... 244
hauteurs ... 1
`hbracket` ... 206
header ... 368
`hideKeySignature` ... 298
`hideNotes` ... 181
`hideStaffSwitch` ... 237
horizontal spacing ... 419
`horizontal-shift` ... 394
`Horizontal_bracket_engraver` ... 188
horizontally centering text ... 524
hufnagel ... 327
`huge` ... 177, 202

I

ictus ... 555
images, intégration ... 207
imbrication de reprises ... 125
imbrication de systèmes ... 155
immutable objects ... 559
immutable properties ... 559
importing stencils into text ... 553
impression de caractères réservés ... 199
impression de caractères spéciaux ... 199
improvisation ... 35
`improvisationOff` ... 35, 67
`improvisationOn` ... 35, 67
including files ... 371
`indent` ... 168, 394, 424
indicating No Chord in ChordNames ... 314
indicating position and barring for fretted instruments ... 282
indication d'octave relative ... 2
indication de corde à vide ... 246
indication de nuance, personnalisation ... 106
indication de tempo et nolet ... 42
indication textuelle ... 193
indications d'archet ... 246
indice ... 201
info-bulle ... 185
inlining an Encapsulated PostScript image ... 539
`inner-margin` ... 394
inserting music into text ... 544
inserting PostScript directly into text ... 541
inserting URL links into text ... 543
`instrument` ... 365
instrument names ... 381
instrument, changement ... 169
instrument, changement de nom ... 168
instrument, noms ... 167
instrument, noms abrégés ... 167
instruments transpositeurs ... 10
`instrumentSwitch` ... 169
`instrumentSwitch` ... 588
intégration de graphiques ... 207
inter-portée, ligatures ... 234
inter-portée, notes ... 234
interface ... 559
interface, layout ... 447
Internals Reference ... 435
invisible, barre de mesure ... 81
invisibles, notes ... 181
`ionian` ... 16
ionien ... 16
`italic` ... 200
italique ... 200

J

jazz chords ... 315
justifié, texte ... 205
`justified-lines` ... 211
`justify` ... 205
justifying lines of text ... 555
justifying text ... 529

K

keep tagged music ... 374
`keepWithTag` ... 588
`key` ... 16, 33
key signature ... 334, 337
key signature, visibility following explicit change ... 469
`killCues` ... 175
`killCues` ... 588
kirchenpausen ... 52

L

`label` ... 588
laissez vibrer ... 45
`laissezVibrer` ... 45
landscape ... 390
`language` ... 588
langues, noms de note ... 7
`large` ... 177, 202
`larger` ... 200, 202
`last-bottom-spacing` ... 393
layers ... 466
`layout file` ... 398
layout interface ... 447
layout objects ... 559
left aligning text ... 530
`left-align` ... 202
`left-margin` ... 394
legato ... 108
`length` ... 237
levée ... 61
levée dans une reprise ... 121
lexer ... 559
lheel ... 555
liaison d'articulation ... 108
liaison de prolongation ... 44
liaison de prolongation et alternative ... 123
liaison de prolongation et reprise avec alternative ... 44
liaison de prolongation, apparence ... 45
liaison de prolongation, pointillés ... 45
liaison de prolongation, tirets ... 45
liaison de prolongationt et reprise ... 123
liaison de tenue et accords ... 44
liaison de tenue et altération ... 5
liaison de tenue et répétition ... 44

liaison et reprise 125
liaison, au-dessous des notes 108
liaison, au-dessus des notes 108
liaison, densité des tirets 109, 111
liaison, laissez vibrer 45
liaison, orientation manuelle 108
liaison, style de trait 108, 111
liaison, tirets 108, 111
liaison, tirets et trait continu 108, 111
liaison, trait plein 108, 111
liaison, trait pointillé 108, 111
liaisons d'articulation multiples 108
liaisons d'articulation simultanées 108
liaisons de phrasé 110
liaisons de phrasé multiples 111
liaisons de phrasé simultanées 111
liaisons de prolongation, positionnement 45
liaisons, dans les paroles 217, 221
ligature en fin de partition 76
ligature en fin de voix polyphonique 76
ligature en soufflet 80
ligature et cadence 62
ligature et mélisme 69
ligature et métrique 55
ligature et musique non mesurée 62
ligature et musique polymétrique 63
ligature et saut de ligne 70
ligature inter-portée 234
ligature manuelle et notes d'ornement 77
ligature manuelle, orientation 77
Ligatures 328
ligatures automatiques, génération de 71
ligatures automatiques, personnalisation 71
ligatures in text 525
ligatures manuelles 69, 77
ligatures, définition de règles 69
ligatures, personnalisation 69
ligatures, subdivision 74
ligne supplémentaire 157
lignes 114
lignes de portée, épaisseur 157
lignes de portée, nombre de 157
line breaks 399
`line-width` 394, 424
lineprall 555
List of colors 495
`locrian` 16
locrien 16
`longa` 37, 47
longfermata 555
longueur de note 37
losange, tête de note 29, 246
`lower` 203
lowering text 531
ltoe 555
`ly:minimal-breaking` 403
`ly:optimal-breaking` 402
`ly:page-turn-breaking` 402
`lydian` 16
lydien 16
lyrics, aligning with sporadic melody 439

M

`m` 309
`magnify` 200
magnifying text 518
`magstep` 177, 460
`maj` 309
majeur 16
`major` 16
`major seven symbols` 317
`majorSevenSymbol` 316
makam 357
makamlar 357
`make-dynamic-script` 106
`make-pango-font-tree` 214
`makeClusters` 136
`makeClusters` 588
manuelle, barre de mesure 81
`Manuels` 1
maqam 352
maqams 352
marcato 99, 555
marges, texte qui dépasse 190
`mark` 90, 193
`markup` 193, 197, 198
markup et fontes 212
markup multi-ligne 204
markup multi-page 211
markup, centrage sur la page 204
markup, commandes d'alignement du texte 206
markup, encadrement du texte 206
markup, expressions 198
markup, inclusion de musique 208
markup, inclusion de partition 210
markup, ornementation du texte 206
markup, rembourrage du texte 206
markup, syntaxe 198
markup, texte justifié 205
markup, texte multi-page 211
markup, texte wordwrapped 205
`markup-markup-spacing` 392
`markup-system-spacing` 392
`markuplines` 197, 211
markups, alignement 202
Masquage de portée 163
masquage des hampes 184
masquées, notes 181
`max-systems-per-page` 396
`maxima` 37, 47
`measureLength` 71, 97
`measurePosition` 61, 97
Medicaea, Editio 327
medium intervals 352
mélisme 222, 224
mélisme et ligature 69
mélodie d'une portée à une autre 237
mélodie, affichage du rythme seul 66
mensural 327
Mensural ligatures 335
`MensuralStaff` 149
MensuralStaffContext 329
MensuralVoiceContext 329
mensuration sign 331
`mergeDifferentlyDottedOff` 140
`mergeDifferentlyDottedOn` 140

`mergeDifferentlyHeadedOff` ... 140
`mergeDifferentlyHeadedOn` ... 140
merging text ... 525
mesure de silence ... 47
mesure entière de silence ... 50
mesure incomplète ... 61
mesure, changement de longueur ... 61
mesure, numéro de ... 86, 97
mesure, répétition de ... 129
mesure, subdivision ... 75
mesures à compter ... 50
mesures, vérification des limites ... 89
`meter` ... 365
métrique ... 54
métrique composite ... 63
métrique double ... 63
métrique et ligature ... 55
métrique par défaut ... 55
métrique polymétrique ... 63
métrique, retour au propriétés par défaut ... 56
métrique, styles ... 55
métrique, visibilité ... 54
métriques composites ... 57
métriques décomposées ... 57
métronome, indication ... 58
`mf` ... 101
microtonalités ... 8
Microtones in MIDI ... 383
MIDI ... 380
MIDI block ... 382
MIDI context definitions ... 383
MIDI et transposition ... 18
MIDI, chord names ... 383
MIDI, microtones ... 383
MIDI, Pitches ... 383
MIDI, quarter tones ... 383
MIDI, Rhythms ... 383
`min-systems-per-page` ... 396
mineur ... 16
`minimumFret` ... 251, 279
`minimumPageTurnLength` ... 402
`minimumRepeatLengthForPageTurn` ... 402
`minor` ... 16
mirroring markup ... 542
`mixed` ... 239
`mixolydian` ... 16
mixolydien ... 16
mode ... 560
mode markup et caractères spéciaux ... 199
mode markup et guillemets ... 199
mode relatif et accord ... 134
mode relatif et autochange ... 236
`modern` ... 22
modern accidental style ... 22
modern cautionary, style d'altération ... 22
modern voice, styme d'altération de précaution ... 23
modern, style d'altération ... 22
modern, style d'altération de précaution ... 22
`modern-cautionary` ... 22
modern-cautionary, style d'altération ... 22
`modern-voice` ... 22
`modern-voice-cautionary` ... 23
moderntab clef ... 258
modes ... 16
modes anciens ... 16
modification du style par défaut des barres de mesure ... 85
modifiers, in chords ... 309
mordant ... 99
mordent ... 555
movements, multiple ... 360
`mp` ... 101
multi-ligne, markup ... 204
multi-ligne, texte ... 204
multi-page, markup ... 211
`MultiMeasureRestText` ... 51
multiples voix ... 140
multiples, liaisons d'articulation ... 108
multiples, liaisons de phrasé ... 111
multivoix, altérations ... 22, 24
Musica ficta ... 334
`musicglyph` ... 91
`musicMap` ... 588
musicQuotes ... 560
musique ancienne, masquage de portée ... 165
musique ancienne, transcription ... 154
musique dans un markup ... 208
musique échelonnée ... 43
musique en parallèle ... 147
musique entremêlée ... 147
musique non mesurée ... 61, 97
musique non mesurée et ligature ... 62
musique non mesurée et saut de ligne ... 62
musique non mesurée et saut de page ... 62
musique répétitive ... 127
mutable objects ... 559
mutable properties ... 559

N

N.C. symbol ... 314
`name` ... 443
naturel ... 5
naturels, harmoniques ... 246
`neo-modern` ... 23
neo-modern, style d'altérations ... 23
`neo-modern-cautionary` ... 24
neo-modern-cautionary, style d'altérations ... 24
`neo-modern-voice` ... 24
neo-modern-voice accidental style ... 24
neo-modern-voice, style d'altération ... 24
`neo-modern-voice-cautionary` ... 24
neomensural ... 327
nested contexts ... 445
new contexts ... 437
niente, al ... 104
no chord symbol ... 314
no reset, style d'altérations ... 25
`no-reset` ... 25
`noBeam` ... 77
nolet, modification du chiffre ... 40
nolets ... 38
nolets, formatage ... 39
nom d'instrument ... 167
nom de personnage ... 228
nom du chanteur ... 228
nombre de portées ... 160
noms d'instrument, autres contextes ... 168
noms d'instrument, centrés ... 167

noms de note ... 1
noms de note, hollandais ... 5
noms de note, langues ... 7
noms de note, par défaut ... 5
non mesurée, musique ... 97
non musicaux, symboles ... 207
non-ASCII characters ... 376
noPageBreak ... 588
noPageTurn ... 589
normalsize ... 177, 202
notation dans un markup ... 208
notation et graphisme ... 207
notation facile ... 31
notation, explication ... 185
note d'ornement, synchronisation ... 96
note head styles ... 514
note heads, ancient ... 332
note-event ... 172
Note_heads_engraver ... 66
notes colorisées dans un accord ... 183
notes d'ornement ... 92
notes d'ornement en fin de note ... 93
notes d'ornement et crochet de nolet ... 42
notes d'ornement et retouches ... 94
notes d'ornement, mise en forme ... 94
notes en couleur ... 182
notes fantômes ... 183
notes inter-portée ... 234
notes penchées ... 35
notes pointées ... 37
notes profilées ... 33
notes simultanées et altérations ... 26
notes within text by log and dot-count ... 544
notes within text by string ... 544
notes, division ... 66
notes, longueur ... 37
notes, noms selon la langue ... 7
notes, petites ... 176
notes, taille ... 177
notes, taille standard ... 178
nouvelle portée ... 149
nuances ... 101
nuances éditoriales ... 106
nuances entre les portées d'un système pianistique ... 234
nuances entre parenthèses ... 106
nuances personnalisées ... 106
nuances successives sur une note tenue ... 102
nuances suggérées ... 106
null ... 203
numericTimeSignature ... 55
numéro de couplet ... 227
numéro de mesure ... 97
numéro de mesure à intervalle régulier ... 86
numéro de mesure et collision ... 89
numéro de mesure et reprises ... 125
numéro de mesure, alignement ... 88
numéro de mesure, formatage ... 87
numéros de mesure ... 86
numérotation des mesures, supression ... 61

O

objects, coloring ... 466
objects, overwriting ... 466
objects, rotating ... 471
objects, visibility of ... 466
objets en couleur ... 182
objets graphiques, tracé ... 206
octavated clefs, visibility of ... 470
octave et clef ... 13
octave relative et accords ... 4
octave relative et transposition ... 3
octave relative, indication ... 2
octave, spécification ... 1
octave, vérification ... 8
octaveCheck ... 8
octaveCheck ... 589
octaves absolues ... 1
octaviation ... 17
oddFooterMarkup ... 368
oddHeaderMarkup ... 368
oneVoice ... 137
open ... 555
opus ... 365
orchestre, cordes ... 245
orgue, marque de pédale ... 99
orientation ... 390
ornementation baroque ... 99
ornementation, symboles ... 99
ornements ... 92
ossia ... 160, 165, 445
ottava ... 17
ottava ... 589
Ottoman music ... 357
outer-margin ... 394
output-count ... 560
output-def ... 560
output-suffix ... 560
outside-staff-horizontal-padding ... 418
outside-staff-padding ... 418
outside-staff-priority ... 418
ouvert ... 99
overrideProperty ... 589
overrides, reverting ... 452
overrideTimeSignatureSettings ... 589
overriding for only one moment ... 453
overriding properties within text markup ... 552
overwriting objects ... 466

P

p ... 101
pad-around ... 206
pad-markup ... 206
pad-to-box ... 206
pad-x ... 206
padding ... 448
padding text ... 532
padding text horizontally ... 532
page breaks ... 424
page breaks, forcing ... 365
page layout ... 368, 424
page size ... 390
page-breaking-system-system-spacing ... 396
page-count ... 396
page-spacing-weight ... 396
pageBreak ... 589

pageTurn ... 589
palmMute ... 589
palmMuteOn ... 589
Pango ... 212
paper size ... 390
paper-height ... 391
paper-width ... 394
parallelMusic ... 147
parallelMusic ... 589
parenthèses, altération entre ... 5
parenthèses, notes entre ... 183
parenthesize ... 183
parenthesize ... 589
parlato ... 216
parlato, tête de note ... 29
paroles ... 216
paroles assignées à une voix ... 137
paroles et ligatures ... 71
paroles et mélodies ... 220
paroles, blanc ... 49
paroles, gestion de l'espacement ... 226
paroles, identificateurs ... 219
paroles, saut de notes ... 49
paroles, variables ... 219
parser ... 560
parser variable ... 560
parseStringResult ... 560
partcombine ... 143
partcombine ... 589
partcombineForce ... 589
partCombineListener ... 560
partial ... 61
parties, combiner des ... 143
partition incluse dans un markup ... 210
paths, drawing ... 540
pédale sostenuto ... 239
pédale sustain ... 239
pédale sustain, style ... 239
pédale, indication combinée de ... 239
pédale, indication graphique de ... 239
pédale, indication textuelle de ... 239
pédale, styles d'indications de ... 239
pédales de harpe ... 244
pédales de piano ... 239
pédales, diagrammes pour harpe ... 244
pedalSustainStyle ... 239
percent ... 129
percussion ... 286, 288
percussion clef ... 286
percussion, portée ... 149
petite note ... 92
petites notes ... 170, 176
petites notes, formater des ... 174
Petrucci ... 327
phrasé, liaisons de ... 110
phrasé, pour des paroles ... 222
phrasingSlurDashed ... 111
phrasingSlurDashPattern ... 111, 589
phrasingSlurDotted ... 111
phrasingSlurDown ... 111
phrasingSlurHalfDashed ... 111
phrasingSlurHalfSolid ... 111
phrasingSlurNeutral ... 111
phrasingSlurSolid ... 111
phrasingSlurUp ... 111
phrygian ... 16
phrygien ... 16
piano accidentals ... 23
piano cautionary, style d'altérations ... 23
piano et altérations ... 23
piano et pédales ... 239
piano, nuances entre les portées ... 234
piano, style d'altérations ... 23
piano, système ... 151
piano, système pour ... 233
piano-cautionary ... 23
PianoStaff ... 233, 235
piece ... 365
pipeSymbol ... 89
Pitch_squash_engraver ... 67
pitchedTrill ... 119
pitchedTrill ... 590
Pitches in MIDI ... 383
pitchnames ... 560
pizzicato, Bartók ... 247
pizzicato, snap ... 247
placing horizontal brackets around text ... 539
placing parentheses around text ... 540
placing vertical brackets around text ... 538
plein, trait de liaison ... 108, 111
plusieurs lignes de texte ... 205
plusieurs pages de texte ... 211
poet ... 365
point ... 37
point d'arrêt ... 99, 112
point d'orgue ... 91, 99
point d'orgue et silence multimesure ... 51
point d'orgue sur une barre de mesure ... 193
pointAndClickOff ... 590
pointAndClickOn ... 590
pointée, note ... 37
pointillé, trait de liaison ... 108, 111
polices, choix par défaut ... 214
polymetric scores ... 441
polymétrie ... 63
polymétrie et ligature ... 63
polyphonie ... 140
polyphonie, portée simple ... 137
ponctuation ... 216
portato ... 99, 555
portée à la française ... 160
portée de batterie ... 149
portée de percussion ... 149
portée de tablature ... 149
portée multiple ... 151
portée rhytmique ... 149
portée simple ... 149
portée vide ... 163
portée, début ... 158
portée, fin ... 158
portée, initialisation ... 149
portée, instanciation ... 149
portée, lignes de ... 157
portée, nouvelle ... 149
portée, reprise ... 157
portée, suspension ... 157
portée, transcription de grégorien ... 149
portées pour instrument à clavier ... 233
portées pour instrument à clés ... 233
portées pour piano ... 233

portées rythmiques, masquage ... 165
portées, groupe de ... 151
portées, nombre variable de ... 160
portées, regroupement ... 151
portées, regroupement et imbrication ... 155
positionnement des silences multimesure ... 52
postscript ... 207
pouce ... 99
pouce, doigté ... 179
pourcent, répétition ... 129
poussé, indication d'archet ... 246
power chords ... 284
powerChords ... 284
pp ... 101
ppp ... 101
pppp ... 101
ppppp ... 101
prall ... 555
pralldown ... 555
prallmordent ... 555
prallprall ... 555
prallup ... 555
précaution, altération de ... 5
predefined string tunings for fretted instruments ... 257
predefinedFretboardsOff ... 278
predefinedFretboardsOn ... 278
première fois ... 120
présentation mensurale ... 154
print-all-headers ... 367, 396
print-first-page-number ... 396
print-page-number ... 396
printing chord names ... 313
printing order ... 466
prob ... 560
prolongateur ... 224
prolongation de texte ... 191
properties ... 450
properties, grob ... 451
property object ... 560
pulsation, regroupement ... 75
putting space around text ... 532

Q

quart de ton ... 7
Quarter tones in MIDI ... 383
quarts de ton ... 5
quotedCueEventTypes ... 172
quotedEventTypes ... 172
quoteDuring ... 170, 174
quoteDuring ... 590

R

r ... 47
R ... 50
ragged-bottom ... 396
ragged-last ... 396, 424
ragged-last-bottom ... 396
ragged-right ... 397
ragged-right ... 424
raise ... 203
raising text ... 533
Ratisbona, Editio ... 327
referencing page numbers in text ... 553
registres, symboles pour accordéon ... 240
réglage des ligatures automatiques ... 71
regroupement automatique de parties ... 143
regular line breaks ... 399
relatif ... 2
relative ... 2, 12, 236
removals, in chords ... 311
remove tagged music ... 374
removeWithTag ... 590
renaissance, musique ... 154
repeatCommands ... 125
repeats in MIDI ... 384
repeatTie ... 44
repère manuel ... 91
repère, format ... 91
repère, personnalisation ... 91
repère, style ... 91
Repères, indication de ... 90
répétition courte ... 129
répétition de mesure ... 129
répétition et liaison de tenue ... 44
répétition, pourcent ... 129
répétitions ... 83, 120
réplique, fin ... 175
répliques ... 170
répliques, formatage ... 174
reprise avec alternative ... 120
reprise avec levée ... 121
reprise courante ... 120
reprise de portée ... 157
reprise et anacrouse ... 121
reprise et liaison ... 125
reprise et liaison de prolongation ... 123
reprise manuelle ... 125
reprise, expansion ... 127
reprises ... 83, 120
reprises ambigües ... 125
reprises avec alternatives et liaisons de prolongation ... 44
reprises imbriquées ... 125
resetRelativeOctave ... 590
respiration, indication ... 112
rest ... 47
rest-event ... 172
rests, ancient ... 333
retouche des notes d'ornement ... 94
retour au propriétés par défaut de la métrique ... 56
reverseturn ... 555
reverting overrides ... 452
revertTimeSignatureSettings ... 590
rfz ... 101
rgb, couleur ... 182
rgb-color ... 182
rheel ... 555
RhythmicStaff ... 149
Rhythms in MIDI ... 383
right aligning text ... 533
right hand fingerings for fretted instruments ... 280
right-align ... 202
right-margin ... 394
rightHandFinger ... 280, 590
root of chord ... 309
rotating objects ... 471

rotating text ... 534
rounded-box ... 206
rtoe ... 555
rvb, couleur ... 182
rythmique d'une mélodie ... 66

S

s ... 49
sacredHarpHeads ... 33
sacredHarpHeadsMinor ... 33
SATB ... 223
saut de ligne et ligature ... 70
sauts ... 113
sauts de durée ... 49
sauts de ligne ... 81
sauts de ligne et cadences ... 62
sauts de ligne et musique non mesurée ... 62
sauts de page et cadences ... 62
sauts de page et musique non mesurée ... 62
scalable vector graphics output ... 379
scaleDurations ... 43, 63
scaleDurations ... 590
scaling markup ... 542
scaling text ... 534
Scheme object ... 561
Scheme variable ... 560
score-markup-spacing ... 392
score-system-spacing ... 392
scoreTitleMarkup ... 368
Scottish highland bagpipe ... 298
script et silence multimesure ... 51
seconde fois ... 120
segno ... 82, 91, 99, 555
segno sur une barre de mesure ... 193
selection de la taille (notation) ... 177
Semai form ... 354
Semi-flat symbol appearance ... 353
semicirculus ... 555
sesqui-bémol ... 8
sesqui-dièse ... 8
set ... 71
set-accidental-style ... 20
set-octavation ... 17
setting extent of text objects ... 554
setting horizontal text alignment ... 527
setting subscript in standard font size ... 518
setting superscript in standard font size ... 519
seventh chords ... 309
sf ... 101
sff ... 101
sfz ... 101
shared properties ... 559
shift note ... 140
shiftDurations ... 590
shiftOff ... 140
shiftOn ... 140
shiftOnn ... 140
shiftOnnn ... 140
short-indent ... 168, 394
shortfermata ... 555
show-available-fonts ... 214
showFirstLength ... 379
showFirstLength ... 560
showKeySignature ... 298
showLastLength ... 379
showLastLength ... 560
showStaffSwitch ... 237
signumcongruentiae ... 555
silence d'église ... 52
silence multi-mesure ... 47
silence multimesure et point d'orgue ... 51
silence multimesure, ajout de texte ... 51
silence multimesure, contraction ... 51
silence multimesure, étiquette ... 51
silence multimesure, expansion ... 51
silence multimesure, positionnement ... 52
silence multimesure, script ... 51
silence, spécification du positionnement vertical ... 47
silences ... 47
silences d'espacement ... 49
silences invisibles ... 49
silences multimesure et doigtés ... 54
silences multimesures ... 50
silences, collisions entre ... 54
silences, condenser les ... 54
silences, décalage automatique ... 140
silences, mesure entière ... 50
simple closure ... 558
simple text strings ... 520
simple text strings with tie characters ... 546
simultanées, liaisons d'articulation ... 108
simultanées, liaisons de phrasé ... 111
skip ... 49
skipTypesetting ... 379
slashed digits ... 553
slides in tablature notation ... 255
slurDashed ... 108
slurDashPattern ... 109
slurDashPattern ... 590
slurDotted ... 108
slurDown ... 108
slurHalfDashed ... 108
slurHalfSolid ... 108
slurNeutral ... 108
slurs, modifying ... 478
slurSolid ... 108
slurUp ... 109
small ... 177, 202
smaller ... 200, 202
smob ... 561
snap pizzicato ... 247
snappizzicato ... 555
Solesmes ... 327
solo ... 143
sos. ... 239
sostenuto, pédale ... 239
sostenutoOff ... 239
sostenutoOn ... 239
soufflet ... 102
soufflet, ligature ... 80
souligné ... 200
Sound ... 380
sourdine ... 99
southernHarmonyHeads ... 33
southernHarmonyHeadsMinor ... 33
sp ... 101
space between staves ... 407
space inside systems ... 407

`spacing` ... 420
spacing, display of layout ... 431
spacing, horizontal ... 419
spacing, vertical ... 407
`spacingTweaks` ... 590
spécification d'un repère ... 91
`spp` ... 101
Sprechgesang ... 216
Square neumes ligatures ... 340
staccatissimo ... 99, 555
staccato ... 99, 555
stacking text in a column ... 524
staff distance ... 407
staff size, setting ... 398
staff symbol, setting of ... 460
`staff-padding` ... 234
`Staff.midiInstrument` ... 381
`Staff_symbol_engraver` ... 163
standard, taille des notes ... 178
`start-repeat` ... 125
`startGroup` ... 188
`startStaff` ... 158, 160
`startTrillSpan` ... 118
`Stem` ... 237
stem, direction ... 184
stem, down ... 184
stem, neutral ... 184
stem, up ... 184
`stem-spacing-correction` ... 420
`stemDown` ... 184
`stemLeftBeamCount` ... 78
`stemNeutral` ... 184
`stemRightBeamCount` ... 78
`stemUp` ... 184
stencil ... 561
stencil, removing ... 466
`stopGroup` ... 188
stopped ... 555
`stopStaff` ... 158, 160, 163
`stopTrillSpan` ... 118
`storePredefinedDiagram` ... 271
`storePredefinedDiagram` ... 590
string numbers ... 249
string vs. fingering numbers ... 249
`stringTunings` ... 268
`StringTunings` ... 257
strumming rhythms, showing ... 67
style d'accidentelle ... 20
style d'altération *modern* ... 22
style d'altération *neo-modern-cautionary* ... 24
style d'altération *neo-modern-voice* ... 24
style de repère ... 91
style de trait, liaison ... 108, 111
`styledNoteHeads` ... 590
styles de métrique ... 55
styles de voix ... 139
styles, tête de note ... 29
`sub` ... 201
subdivision de ligature ... 74
subscript text ... 521
substitution de doigt ... 179
`subsubtitle` ... 365
`subtitle` ... 365
`suggestAccidentals` ... 334
`super` ... 201
superscript text ... 522
`sus` ... 312
suspension de portée ... 157
sustain, pédale ... 239
sustain, style de pédale ... 239
`sustainOff` ... 239
`sustainOn` ... 239
SVG output ... 379
symbole de portée ... 157
symboles d'ornementation ... 99
symboles non musicaux ... 207
synchronisation des notes d'ornement ... 96
syntaxe du mode markup ... 198
`system-count` ... 397
`system-separator-markup` ... 397
`system-system-spacing` ... 392
système ... 151
système choral ... 151
système pianistique ... 151
système, début de ... 151
système, grand ... 151
système, indicateur de séparation ... 156
systèmes imbriqués ... 155
`systems-per-page` ... 397

T

tab clef ... 258
`tabChordRepetition` ... 590
tablature ... 149, 248
tablature and harmonic indications ... 255
tablature and slides ... 255
tablature, banjo ... 248, 257, 285
tablature, bass guitar ... 257
tablature, guitar ... 248, 257
tablature, mandolin ... 257
tablature, predefined string tunings ... 257
tablature, ukulele ... 257
tablatures, basic ... 250
tablatures, custom ... 257
tablatures, default ... 250
`TabStaff` ... 149, 250
`TabVoice` ... 250
tag ... 374
`tag` ... 590
`tagline` ... 365
tagline ... 368
taille des notes ... 177
`taor` ... 298
taqasim ... 354
`teaching` ... 25
teaching, style d'altérations ... 25
`teeny` ... 177, 202
Template Arabic music ... 356
`tempo` ... 58
temporel, cadrillage ... 186
temps, gestion du ... 97
tenor, clef ... 13
tenue et nuances successives ... 102
tenuto ... 99, 555
tessiture ... 27
tête de note allongée ... 35
tête de note en losange ... 246
tête de note et improvisation ... 35

tête de note, Aiken ... 33
tête de note, allure ... 33
tête de note, apprentissage ... 31
tête de note, Christian Harmony ... 33
tête de note, Funk ... 33
tête de note, Harmonica Sacra ... 33
tête de note, harpe sacrée ... 33
tête de note, Walker ... 33
têtes de note ... 177
têtes de note, spéciales ... 29
têtes de note, styles ... 139
`text` ... 239
text columns, left-aligned ... 531
text columns, right-aligned ... 533
text, standalone ... 197
text, top-level ... 197
texte ajouté ... 198
texte en colonnes ... 204
texte en préambule ... 197
texte et alternative ... 126
texte et extenseur ... 191
texte et rembourrage ... 206
texte hors marges ... 190
texte indépendant ... 197
texte indiquant le nombre de mesures vides ... 51
texte isolé ... 197
texte justifié ... 205
texte multi-ligne ... 204
texte sur plusieurs lignes ... 205
texte, alignement ... 202
texte, alignement horizontal ... 202
texte, alignement vertical ... 203
Texte, autres langues ... 189
texte, centrage sur la page ... 204
texte, décoration ... 206
texte, encadrement ... 206
texte, maintien dans les marges ... 190
texte, mise en forme des extenseurs ... 191
texte, mise en forme des prolongations ... 191
texte, taille ... 200
`textLengthOff` ... 190
`textLengthOn` ... 52, 190
`textLenthOff` ... 52
`textSpannerDown` ... 191
`textSpannerNeutral` ... 191
`textSpannerUp` ... 191
Thorough bass ... 320
`thumb` ... 179
thumb ... 555
`tieDashed` ... 45
`tieDashPattern` ... 590
`tieDotted` ... 45
`tieDown` ... 45
`tieNeutral` ... 45
ties, modifying ... 478
`tieSolid` ... 45
`tieUp` ... 45
`time` ... 54, 71
time signatures ... 331
Time signatures, multiple ... 441
`times` ... 38, 63
`timeSignatureFraction` ... 63
`tiny` ... 177, 202
tiré, indication d'archet ... 246
tiret, trait de liaison ... 108, 111
tirettes d'accordéon, symboles ... 240
`title` ... 365
titles ... 368
`tocItem` ... 590
`Top` ... 1
top-level text ... 197
`top-margin` ... 391
`top-markup-spacing` ... 393
`top-system-spacing` ... 393
toplevel-bookparts ... 560
toplevel-scores ... 560
tracé d'objets graphiques ... 206
trait d'union ... 224
transcription de musique ancienne ... 154
`translate` ... 203
`translate-scaled` ... 203
translating text ... 534
transparent, making objects ... 466
transparentes, notes ... 181
`transpose` ... 2, 3, 9, 12
`transposedCueDuring` ... 175
`transposedCueDuring` ... 591
transposing fret diagrams ... 269
transpositeur, instrument ... 10
transposition ... 9
`transposition` ... 18, 170
`transposition` ... 591
transposition des hauteurs ... 9
transposition et clef ... 13
transposition et MIDI ... 18
transposition et mode octave relative ... 3
transposition, instrument ... 18
tre corde ... 239
`treCorde` ... 239
`tremolo` ... 131
tremolo, cross-staff ... 133
trémolo, indication de ... 132
trémolo, ligatures de ... 131
`tremoloFlags` ... 132
triads ... 309
`triangle` ... 207
`trill` ... 118
trill ... 555
trille ... 99
trilles ... 118
trilles avec hauteur explicite ... 119
trilles avec hauteur explicite et altération ... 119
triolets ... 38
triolets, formatage ... 39
tuning, non-Western ... 351
tunings, banjo ... 285
`tupletDown` ... 39
`tupletNeutral` ... 39
`TupletNumber` ... 40
`tupletNumberFormatFunction` ... 39
`tupletSpannerDuration` ... 39
`tupletUp` ... 39
Turkish music ... 357
Turkish note names ... 357
turn ... 555
`tweak` ... 591
tweaking ... 453
tweaking control points ... 455
tweaks in a variable ... 455
tweaks in lyrics ... 455

`two-sided` ... 397
`type` ... 443
typeface ... 558

U

U.C. ... 239
ukulele ... 260
una corda ... 239
`unaCorda` ... 239
`underline` ... 200
underlining text ... 523
une pause par mesure ... 50
`unfold` ... 127
`unfoldRepeats` ... 591
`unHideNotes` ... 181
Unicode ... 376
upbow ... 555
upmordent ... 555
upprall ... 555
UTF-8 ... 376

V

varcoda ... 99, 555
variables ... 364
variables, use of ... 373
variante ... 160
Vaticana, Editio ... 327
`VaticanaStaff` ... 149
VaticanaStaffContext ... 336
VaticanaVoiceContext ... 336
vérification d'octave ... 8
vérification des limites de mesure ... 89
vertical positioning of dynamics ... 103
vertical spacing ... 407, 424
vertically centering text ... 535
verylongfermata ... 555
visibilité des hampes ... 184
visibility of objects ... 466
visibility of octavated clefs ... 470
vocalise ... 222
`voice` ... 20, 22
`Voice` ... 137
voice, style d'altérations ... 22
`voiceOne` ... 137
voix ... 137
voix entre deux portées ... 237
voix multiples ... 140
voix multiples et altérations ... 22, 24
voix, décalage ... 140
voix, styles ... 139
volta ... 120

W

Walker shape, tête de note ... 33
`walkerHeads` ... 33
`walkerHeadsMinor` ... 33
`whichBar` ... 85
White mensural ligatures ... 335
whitespace ... 364
wind instruments ... 295
`with-color` ... 182
`withMusicProperty` ... 591
`wordwrap` ... 205
`wordwrap-lines` ... 211

X

x11, couleur ... 183
x11, couleurs ... 182
`x11-color` ... 182, 183
`xNote` ... 591
`xNotesOn` ... 591