# LilyPond

Il compositore tipografico per la musica

# Guida alla Notazione

## Il team di sviluppo di LilyPond

Questo manuale costituisce la guida di riferimento per tutti gli aspetti relativi alla notazione musicale in LilyPond versione 2.21.82. Si presuppone che il lettore conosca il materiale esposto nel Sezione “Manuale di Apprendimento” in *Manuale di Apprendimento*.

Questo manuale è disponibile in altri formati ed è integrato col resto della documentazione. Maggiori informazioni in Sezione “Manuali” in *Informazioni generali*.

La documentazione completa si trova all’indirizzo `http://lilypond.org/`.

Copyright © 1999–2020 degli autori.*La traduzione della seguente nota di copyright è gentilmente offerta alle persone che non parlano inglese, ma solo la nota in inglese ha valore legale.*

*The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.*

> È garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della GNU Free Documentation License, versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation; senza alcuna sezione non modificabile. Una copia della licenza si trova nella sezione intitolata "GNU Free Documentation License".
>
> 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”.

Per la versione di LilyPond 2.21.82

# Sommario


**1 Notazione musicale ... 1**

1.1 Altezze ... 1

1.1.1 Inserimento delle altezze ... 1

Ottava assoluta ... 1

Ottava relativa ... 2

Alterazioni ... 6

Nomi delle note in altre lingue ... 8

1.1.2 Modifica di più altezze ... 10

Controlli di ottava ... 10

Trasposizione ... 11

Inversione ... 14

Retrogradazione ... 15

Trasposizioni modali ... 15

1.1.3 Aspetto delle altezze ... 17

Chiave ... 17

Armatura di chiave ... 22

Segni di ottavazione ... 24

Trasporto strumentale ... 27

Alterazioni automatiche ... 28

Ambitus ... 36

1.1.4 Teste di nota ... 39

Teste di nota speciali ... 39

Testa di nota con nome della nota ... 40

Teste di nota a forma variabile ... 42

Improvvisazione ... 45

1.2 Ritmi ... 45

1.2.1 Inserimento delle durate ... 46

Durata ... 46

Gruppi irregolari ... 49

Scalare le durate ... 53

Legature di valore ... 55

1.2.2 Inserimento delle pause ... 59

Pause ... 59

Pause invisibili ... 61

Pause d'intero ... 62

1.2.3 Aspetto dei ritmi ... 67

Indicazione di tempo ... 67

Indicazioni metronomiche ... 71

Anacrusi ... 75

Musica in tempo libero ... 76

Notazione polimetrica ... 78

Divisione automatica delle note ... 81

Mostrare i ritmi della melodia ... 83

1.2.4 Travature ... 85

Travature automatiche ... 85

Impostare il comportamento delle travature automatiche ... 88

Travature manuali ... 97

Travature a raggiera ... 100

1.2.5 Battute . . . 101
Stanghette . . . 101
Numeri di battuta . . . 108
Controlli di battuta e del numero di battuta . . . 113
Segni di chiamata . . . 114
1.2.6 Questioni ritmiche particolari . . . 116
Abbellimenti . . . 116
Allineamento sulle cadenze . . . 121
Gestione del tempo . . . 122
1.3 Segni di espressione . . . 123
1.3.1 Segni di espressione collegati alle note . . . 124
Articolazioni e abbellimenti . . . 124
Dinamiche . . . 126
Nuove indicazioni dinamiche . . . 133
1.3.2 Indicazioni espressive curvilinee . . . 135
Legature di portamento . . . 135
Legature di frase . . . 138
Respiri . . . 139
Portamenti indeterminati discendenti (cadute) e ascendenti . . . 141
1.3.3 Indicazioni espressive lineari . . . 142
Glissando . . . 142
Arpeggio . . . 147
Trilli . . . 150
1.4 Ripetizioni . . . 152
1.4.1 Ripetizioni lunghe . . . 153
Ripetizioni normali . . . 153
Indicazioni di ripetizione manuali . . . 160
Ripetizioni ricopiate . . . 162
1.4.2 Ripetizioni brevi . . . 164
Ripetizioni con percentuale . . . 164
Ripetizioni con tremolo . . . 167
1.5 Note simultanee . . . 169
1.5.1 Una voce . . . 169
Note in un accordo . . . 170
Ripetizione di un accordo . . . 171
Espressioni simultanee . . . 173
Cluster . . . 175
1.5.2 Più voci . . . 175
Polifonia su un solo rigo . . . 175
Stili di voce . . . 179
Risoluzione delle collisioni . . . 179
Accorpare le pause . . . 185
Combinazione automatica delle parti . . . 185
Scrivere la musica in parallelo . . . 191
1.6 Notazione del rigo . . . 193
1.6.1 Aspetto del rigo . . . 193
Istanziare nuovi righi . . . 194
Raggruppare i righi . . . 195
Gruppi di righi annidati . . . 198
Separare i sistemi . . . 200
1.6.2 Modificare singoli righi . . . 201
Simbolo del rigo . . . 201
Righi ossia . . . 204
Nascondere i righi . . . 208

1.6.3 Scrittura delle parti ... 210
Nomi degli strumenti ... 210
Citare altre voci ... 213
Formattazione delle notine ... 217
1.7 Note editoriali ... 222
1.7.1 Interne al rigo ... 222
Scelta della dimensione del tipo di carattere ... 222
Indicazioni di diteggiatura ... 226
Note nascoste ... 229
Colorare gli oggetti ... 229
Parentesi ... 231
Gambi ... 232
1.7.2 Esterne al rigo ... 233
Nuvoletta di aiuto ... 233
Linee della griglia ... 234
Parentesi analitiche ... 236
1.8 Testo ... 238
1.8.1 Inserimento del testo ... 239
Scritte ... 239
Estensori del testo ... 240
Indicazioni testuali ... 242
Testo separato ... 245
1.8.2 Formattazione del testo ... 246
Introduzione al testo a margine ... 247
Scelta del tipo di carattere e della dimensione ... 248
Allineamento del testo ... 251
Notazione grafica nel blocco markup ... 255
Notazione musicale nel blocco markup ... 257
Testo formattato su più pagine ... 260
1.8.3 Tipi di carattere ... 260
Tipi di carattere in dettaglio ... 261
Tipi di carattere per singolo oggetto ... 264
Tipi di carattere per l'intero documento ... 264

**2 Notazione specialistica ... 267**
2.1 Musica vocale ... 267
2.1.1 Notazione comune per la musica vocale ... 267
Riferimenti per la musica vocale ... 267
Inserimento del testo vocale ... 268
Allineamento del testo alla melodia ... 269
Durate automatiche delle sillabe ... 271
Durate manuali delle sillabe ... 274
Più sillabe in una nota ... 275
Più note in una sillaba ... 276
Estensori e trattini ... 279
2.1.2 Tecniche specifiche per il testo vocale ... 279
Lavorare con testo e variabili ... 279
Posizionamento verticale del testo ... 281
Posizionamento orizzontale delle sillabe ... 285
Testo e ripetizioni ... 287
Testi alternati ... 295
Polifonia con testo in comune ... 297
2.1.3 Strofe ... 298
Aggiungere i numeri di strofa ... 298

Aggiungere le dinamiche alle strofe . . . 299
Aggiungere i nomi dei cantanti alle strofe . . . 300
Strofe con ritmi diversi . . . 300
Stampare le strofe alla fine . . . 303
Stampare le strofe alla fine in molteplici colonne . . . 304
2.1.4 Canzoni . . . 306
Riferimenti per canzoni . . . 306
Canzonieri . . . 307
2.1.5 Musica corale . . . 307
Riferimenti per musica corale . . . 307
Struttura di una partitura corale . . . 308
2.1.6 Opera e musical . . . 310
Riferimenti per opera e musical . . . 311
Nomi dei personaggi . . . 311
Suggerimenti musicali . . . 313
Musica parlata . . . 317
Dialogo sopra la musica . . . 317
2.1.7 Canti salmi e inni . . . 318
Riferimenti per canti e salmi . . . 318
Impostare un canto . . . 319
Salmi . . . 325
Misure parziali nei motivi degli inni . . . 329
2.1.8 Musica vocale antica . . . 331
2.2 Tastiera e altri strumenti multirigo . . . 331
2.2.1 Notazione comune per tastiere . . . 332
Riferimenti per tastiere . . . 332
Modifica manuale del rigo . . . 333
Modifica automatica del rigo . . . 335
Linee del cambio rigo . . . 337
2.2.2 Pianoforte . . . 340
Pedali del pianoforte . . . 340
2.2.3 Fisarmonica . . . 341
Simboli di discanto . . . 341
2.2.4 Arpa . . . 342
Riferimenti per arpe . . . 342
Pedali dell'arpa . . . 342
2.3 Strumenti a corde senza tasti . . . 343
2.3.1 Notazione comune per strumenti a corde senza tasti . . . 344
Riferimenti per archi senza tasti . . . 344
Indicazioni di arcata . . . 344
Armonici . . . 345
Pizzicato alla Bartók . . . 346
2.4 Strumenti a corde con tasti . . . 346
2.4.1 Notazione comune per strumenti a corde con tasti . . . 347
Riferimenti per strumenti a corde con tasti . . . 347
Indicazioni di numero di corda . . . 348
Intavolature predefinite . . . 350
Intavolature personalizzate . . . 364
Diagrammi dei tasti . . . 368
Diagrammi dei tasti predefiniti . . . 378
Diagrammi dei tasti automatici . . . 388
Diteggiature della mano destra . . . 391
2.4.2 Chitarra . . . 393
Indicazione di posizione e barré . . . 393

Indicazione di armonici e note smorzate . . . 393
Indicazione di power chord . . . 395
2.4.3 Banjo . . . 396
Intavolature per banjo . . . 396
2.4.4 Liuto . . . 397
Intavolature per liuto . . . 397
2.5 Percussioni . . . 398
2.5.1 Notazione comune per le percussioni . . . 398
Riferimenti per percussioni . . . 398
Notazione di base per percussioni . . . 398
Rulli di tamburo . . . 399
Percussioni a suono determinato . . . 400
Righi delle percussioni . . . 400
Righi delle percussioni personalizzati . . . 402
Ghost note . . . 405
2.6 Strumenti aerofoni . . . 405
2.6.1 Notazione comune per gli strumenti aerofoni . . . 405
Riferimenti per strumenti aerofoni . . . 406
Diteggiature . . . 407
2.6.2 Cornamusa . . . 409
Definizioni per cornamusa . . . 409
Esempio per cornamusa . . . 409
2.6.3 Legni . . . 411
2.6.3.1 Diagrammi per legni . . . 411
2.7 Notazione per accordi . . . 419
2.7.1 Modalità accordo . . . 419
Panoramica sulla modalità accordo . . . 419
Accordi comuni . . . 420
Accordi estesi e alterati . . . 422
2.7.2 Visualizzazione accordi . . . 424
Stampa dei nomi degli accordi . . . 424
Personalizzazione dei nomi degli accordi . . . 427
2.7.3 Basso continuo . . . 432
Introduzione al basso continuo . . . 433
Inserimento del basso continuo . . . 433
Visualizzazione del basso continuo . . . 436
2.8 Musica contemporanea . . . 439
2.8.1 Altezza e armonica nella musica contemporanea . . . 439
Riferimenti per altezza e armonia nella musica contemporanea . . . 439
Notazione microtonale . . . 439
Armature di chiave e armonia contemporanee . . . 439
2.8.2 Approcci contemporanei al ritmo . . . 439
Riferimenti per approcci contemporanei al ritmo . . . 439
Gruppi irregolari nella musica contemporanea . . . 439
Indicazioni di tempo contemporanee . . . 439
Notazione polimetrica estesa . . . 439
Travature nella musica contemporanea . . . 439
Stanghette nella musica contemporanea . . . 439
2.8.3 Notazione grafica . . . 439
2.8.4 Tecniche di arrangiamento contemporaneo . . . 439
2.8.5 Nuove tecniche strumentali . . . 439
2.8.6 Letture consigliate e partiture rilevanti . . . 440
Libri e articoli sulla notazione musicale contemporanea . . . 440
Partiture e esempi musicali . . . 440

2.9 Notazione antica . . . 440
2.9.1 Panoramica degli stili supportati . . . 441
2.9.2 Notazione antica—funzionalità comuni . . . 442
Contesti predefiniti . . . 442
Legature . . . 442
Custodes . . . 443
2.9.3 Scrivere la musica mensurale . . . 444
Contesti mensurali . . . 444
Chiavi mensurali . . . 445
Indicazioni di tempo mensurali . . . 446
Teste di nota mensurali . . . 447
Code mensurali . . . 448
Pause mensurali . . . 448
Alterazioni e armature di chiave mensurali . . . 449
Alterazioni suggerite (*musica ficta*) . . . 450
Legature mensurali bianche . . . 450
2.9.4 Scrivere il canto gregoriano . . . 452
Contesti del canto gregoriano . . . 452
Chiavi gregoriane . . . 452
Alterazioni e armature di chiave gregoriane . . . 453
Divisiones . . . 454
Segni di articolazione gregoriani . . . 454
Punti di aumentazione (morae) . . . 455
Legature di neumi quadrati gregoriani . . . 456
2.9.5 Scrivere in notazione quadrata di Kiev . . . 462
Contesti della notazione di Kiev . . . 462
Chiavi della notazione di Kiev . . . 463
Note della notazione di Kiev . . . 463
Alterazioni della notazione di Kiev . . . 464
Stanghetta della notazione di Kiev . . . 464
Melismi della notazione di Kiev . . . 464
2.9.6 Lavorare con la musica antica—scenari e soluzioni . . . 465
Incipit . . . 466
Formattazione mensurstriche . . . 466
Trascrivere il canto gregoriano . . . 467
Antico e moderno da un unico sorgente . . . 470
Correzioni editoriali . . . 472
2.10 World music . . . 472
2.10.1 Notazione comune per la musica non occidentale . . . 472
Estensione dei sistemi di notazione e di accordatura . . . 472
2.10.2 Musica araba . . . 473
Referenze per la musica araba . . . 473
Nomi delle note in arabo . . . 474
Armature di chiave arabe . . . 474
Indicazioni di tempo arabe . . . 476
Esempio di musica araba . . . 477
Letture complementari sulla musica araba . . . 477
2.10.3 Musica classica turca . . . 478
Riferimenti per la musica classica turca . . . 478
Nomi delle note in turco . . . 478

**3 Input e output ... 480**

3.1 Struttura dell'input ... 480
3.1.1 Struttura di una partitura ... 480
3.1.2 Molteplici partiture in un libro ... 481
3.1.3 Molteplici file di output da un unico file di input ... 482
3.1.4 Nomi dei file di output ... 483
3.1.5 Struttura del file ... 484
3.2 Titoli e intestazioni ... 486
3.2.1 Creazione di titoli intestazioni e piè di pagina ... 486
Come funzionano i titoli ... 486
Formattazione predefinita dei titoli delle parti e dei brani ... 489
Formattazione predefinita delle intestazioni e dei piè di pagina ... 492
3.2.2 Titoli intestazioni e piè di pagina personalizzati ... 493
Titoli personalizzati ... 493
Formattazione personalizzata dei titoli ... 494
Formattazione personalizzata di intestazioni e piè di pagina ... 497
3.2.3 Creazione di metadati per i file di output ... 498
3.2.4 Creazione di note a piè di pagina ... 499
Note a piè di pagina nelle espressioni musicali ... 499
Note a piè di pagina nel testo separato ... 505
3.2.5 Riferimento ai numeri di pagina ... 508
3.2.6 Indice ... 509
3.3 Lavorare coi file di input ... 511
3.3.1 Inclusione di file LilyPond ... 511
3.3.2 Edizioni diverse da un unico sorgente ... 513
Uso delle variabili ... 513
Uso delle etichette ... 515
Impostazioni globali ... 519
3.3.3 Caratteri speciali ... 519
Codifica del testo ... 519
Unicode ... 519
Alias ASCII ... 520
3.4 Controllo dell'output ... 521
3.4.1 Estrarre frammenti musicali ... 521
3.4.2 Saltare la musica già corretta ... 522
3.4.3 Formati di output alternativi ... 523
Output SVG ... 523
3.4.4 Cambiare il tipo di carattere della notazione ... 523
3.5 Creazione dell'output MIDI ... 525
3.5.1 Notazione supportata nel MIDI ... 525
3.5.2 Notazione non supportata nel MIDI ... 526
3.5.3 Il blocco MIDI ... 526
3.5.4 Gestione delle dinamiche nel MIDI ... 527
Dinamiche nel MIDI ... 527
Impostazione del volume MIDI ... 528
Impostazione delle proprietà del blocco MIDI ... 531
3.5.5 Uso degli strumenti MIDI ... 532
3.5.6 Uso delle ripetizioni nel MIDI ... 532
3.5.7 Mappatura dei canali MIDI ... 533
3.5.8 Proprietà di contesto per gli effetti MIDI ... 535
3.5.9 Miglioramento dell'output MIDI ... 536
Lo script `articulate` ... 537
3.6 Estrazione dell'informazione musicale ... 537
3.6.1 Mostrare la notazione LilyPond ... 537

3.6.2 Mostrare le espressioni musicali Scheme ..... 538
3.6.3 Salvare eventi musicali in un file ..... 538

**4 Gestione dello spazio ..... 539**

4.1 Formattazione della pagina ..... 539
4.1.1 Il blocco `\paper` ..... 539
4.1.2 Formato carta e ridimensionamento automatico ..... 540
Impostare il formato carta ..... 540
Ridimensionamento automatico al formato carta ..... 541
4.1.3 Variabili `\paper` della spaziatura verticale fissa ..... 542
4.1.4 Variabili `\paper` della spaziatura verticale flessibile ..... 542
Struttura delle liste associative flessibili della spaziatura verticale ..... 543
Elenco delle variabili `\paper` flessibili della spaziatura verticale ..... 544
4.1.5 Variabili `\paper` della spaziatura orizzontale ..... 544
Variabili `\paper` per larghezze e margini ..... 545
Variabili `\paper` per la modalità due pagine per foglio ..... 546
Variabili `\paper` per spostamenti e indentazioni ..... 547
4.1.6 Altre variabili di `\paper` ..... 547
Variabili di `\paper` per l'interruzione di linea ..... 547
Variabili di `\paper` per l'interruzione di pagina ..... 548
Variabili di `\paper` per la numerazione delle pagine ..... 549
Svariate variabili di `\paper` ..... 549
4.2 Formattazione della partitura ..... 550
4.2.1 Il blocco `\layout` ..... 550
4.2.2 Impostare la dimensione del rigo ..... 552
4.3 Interruzioni ..... 554
4.3.1 Interruzioni di linea ..... 554
4.3.2 Interruzioni di pagina ..... 557
Interruzione di pagina manuale ..... 557
Interruzione di pagina ottimale ..... 559
Interruzione di pagina minimale ..... 559
Interruzione di pagina di una pagina ..... 559
Interruzione di pagina su una linea ..... 559
Interruzione di pagina su una linea con altezza automatica ..... 559
Voltata di pagina ottimale ..... 560
4.4 Spaziatura verticale ..... 561
4.4.1 Spaziatura verticale flessibile all'interno dei sistemi ..... 561
Proprietà della spaziatura dentro un sistema ..... 561
Spaziatura dei righi non raggruppati ..... 564
Spaziatura dei righi raggruppati ..... 565
Spaziatura delle linee che non sono righi ..... 567
4.4.2 Posizionamento esplicito di righi e sistemi ..... 568
4.4.3 Elusione delle collisioni verticali ..... 576
4.5 Spaziatura orizzontale ..... 577
4.5.1 Panoramica sulla spaziatura orizzontale ..... 577
4.5.2 Nuova spaziatura nel corso di un brano ..... 579
4.5.3 Modifica della spaziatura orizzontale ..... 580
Allungamento uniforme dei gruppi irregolari ..... 581
Spaziatura rigorosa della nota ..... 581
4.5.4 Larghezza della linea ..... 582
4.5.5 Notazione proporzionale ..... 582
4.6 Riduzione del numero di pagine di una partitura ..... 588
4.6.1 Visualizzare la spaziatura ..... 589
4.6.2 Modificare la spaziatura ..... 590

**5 Modifica delle impostazioni predefinite ... 592**
5.1 Contesti di interpretazione ... 592
5.1.1 Tutto sui contesti ... 592
Definizioni di output - gerarchia dei contesti ... 592
Score - il padre di tutti i contesti ... 593
Contesti del livello superiore - contenitori di righi ... 593
Contesti del livello intermedio - righi ... 593
Contesti del livello inferiore - voci ... 594
5.1.2 Creazione e citazione di un contesto ... 594
5.1.3 Conservazione di un contesto ... 598
5.1.4 Modifica dei componenti aggiuntivi di un contesto ... 600
5.1.5 Modifica delle impostazioni predefinite di un contesto ... 602
Modifica di tutti i contesti dello stesso tipo ... 603
Modifica di un solo contesto specifico ... 605
Ordine di precedenza ... 607
5.1.6 Definizione di nuovi contesti ... 607
5.1.7 Ordine di disposizione dei contesti ... 610
5.2 Come funziona la Guida al funzionamento interno ... 612
5.2.1 Navigazione nella guida al programma ... 612
5.2.2 Interfacce di formattazione ... 612
5.2.3 Determinazione della proprietà del grob ... 614
5.2.4 Convenzioni sui nomi ... 615
5.3 Modifica delle proprietà ... 615
5.3.1 Panoramica sulla modifica delle proprietà ... 615
5.3.2 Il comando `\set` ... 615
5.3.3 Il comando `\override` ... 617
5.3.4 Il comando `\tweak` ... 619
5.3.5 `\set` vs. `\override` ... 621
5.3.6 Il comando `\offset` ... 622
5.3.7 Modifica delle liste associative ... 627
5.4 Proprietà e concetti utili ... 629
5.4.1 Modalità di inserimento ... 629
5.4.2 Direzione e posizionamento ... 630
Indicatori di direzione delle articolazioni ... 630
La proprietà direction ... 631
5.4.3 Distanze e misurazioni ... 632
5.4.4 Dimensioni ... 632
5.4.5 Proprietà del simbolo del rigo ... 633
5.4.6 Estensori ... 633
Uso di `spanner-interface` ... 633
Uso di `line-spanner-interface` ... 636
5.4.7 Visibilità degli oggetti ... 638
Soppressione dello stampo ... 638
Rendere gli oggetti trasparenti ... 639
Dipingere gli oggetti di bianco ... 639
Uso di break-visibility ... 640
Considerazioni speciali ... 642
5.4.8 Stili della linea ... 644
5.4.9 Rotazione degli oggetti ... 645
Rotazione degli oggetti della formattazione ... 645
Rotazione del testo ... 646
5.5 Ritocchi avanzati ... 646
5.5.1 Allineamento degli oggetti ... 646
Impostazione diretta di `X-offset` e `Y-offset` ... 647

Uso di `side-position-interface` ..... 647
Uso di `self-alignment-interface` ..... 648
Uso di `break-alignable-interface` ..... 649
5.5.2 Raggruppamento verticale dei grob ..... 651
5.5.3 Modifica degli stampi ..... 651
5.5.4 Modifica delle forme ..... 652
Modifica di legature di valore e di portamento ..... 652
5.5.5 Modifica degli estensori spezzati ..... 656
Uso di `\alterBroken` ..... 656
5.5.6 Contenitori unpure-pure ..... 658
5.6 Uso delle funzioni musicali ..... 659
5.6.1 Sintassi della funzione di sostituzione ..... 660
5.6.2 Esempi della funzione di sostituzione ..... 660

**Appendice A Tabelle del manuale della notazione ..... 663**
A.1 Grafico dei nomi degli accordi ..... 663
A.2 Modificatori degli accordi comuni ..... 664
A.3 Accordature predefinite ..... 667
A.4 Diagrammi degli accordi predefiniti ..... 668
Diagrammi per chitarra ..... 668
Diagrammi per ukulele ..... 670
Diagrammi per mandolino ..... 672
A.5 Formati carta predefiniti ..... 674
A.6 Strumenti MIDI ..... 677
A.7 Elenco dei colori ..... 678
A.8 Il font Emmentaler ..... 679
Glifi della chiave ..... 680
Glifi delle indicazioni di tempo ..... 680
Glifi dei numeri ..... 680
Glifi delle alterazioni ..... 681
Glifi delle teste di nota predefinite ..... 682
Glifi delle teste di nota speciali ..... 682
Glifi delle teste di nota a forma variabile ..... 683
Glifi delle pause ..... 687
Glifi delle code ..... 687
Glifi dei punti ..... 688
Glifi delle dinamiche ..... 688
Glifi dei segni ..... 689
Glifi delle teste a forma di freccia ..... 691
Glifi delle estremità delle parentesi ..... 692
Glifi dei pedali ..... 692
Glifi della fisarmonica ..... 692
Glifi delle legature di valore ..... 692
Glifi della notazione vaticana ..... 693
Glifi della notazione medicea ..... 694
Glifi Hufnagel ..... 694
Glifi della notazione mensurale ..... 695
Glifi della notazione neomensurale ..... 698
Glifi Petrucci ..... 699
Glifi Solesmes ..... 700
Glifi della notazione di Kiev ..... 700
A.9 Stili delle teste di nota ..... 701
A.10 Stili della chiave ..... 702
Chiavi standard ..... 702

Chiave del rigo della percussione ... 703
Chiavi del rigo dell'intavolatura ... 703
Chiavi della musica antica ... 703
A.11 Comandi per *markup* ... 707
A.11.1 Font ... 707
A.11.2 Align ... 717
A.11.3 Graphic ... 732
A.11.4 Music ... 740
A.11.5 Instrument Specific Markup ... 746
A.11.6 Accordion Registers ... 749
A.11.7 Other ... 754
A.12 Comandi per una lista di *markup* ... 761
A.13 Elenco dei caratteri speciali ... 763
A.14 Elenco delle articolazioni ... 764
Articolazioni ... 764
Ornamenti ... 765
Punti coronati ... 765
Segni specifici per strumento ... 765
Segni di ripetizione ... 766
Segni antichi ... 766
A.15 Note percussive ... 766
A.16 Glossario tecnico ... 769
alist ... 769
callback ... 769
closure ... 769
glyph ... 769
grob ... 769
immutable ... 770
interface ... 770
lexer ... 770
mutable ... 770
output-def ... 770
parser ... 770
parser variable ... 771
prob ... 771
smob ... 771
stencil ... 771
A.17 Tutte le proprietà di contesto ... 772
A.18 Proprietà della formattazione ... 785
A.19 Funzioni musicali disponibili ... 806
A.20 Identificatori delle modifiche di contesto ... 817
A.21 Tipi di predicati predefiniti ... 818
R5RS primary predicates ... 818
R5RS secondary predicates ... 818
Guile predicates ... 819
LilyPond scheme predicates ... 819
LilyPond exported predicates ... 820
A.22 Funzioni Scheme ... 821

**Appendice B Schema riassuntivo ... 847**

**Appendice C GNU Free Documentation License ... 851**

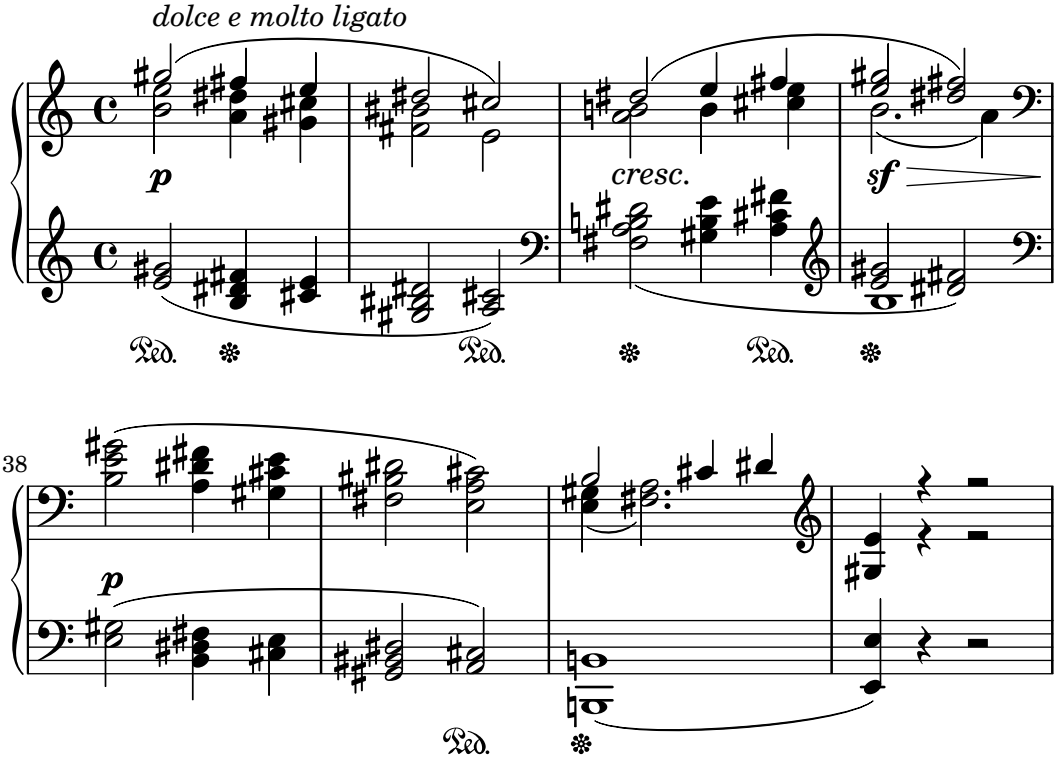
**Appendice D Indice dei comandi di LilyPond ..... 858**

**Appendice E Indice di LilyPond ..... 868**

# 1 Notazione musicale

Questo capitolo spiega come creare la notazione musicale.

## 1.1 Altezze




Questa sezione tratta il modo in cui si determina l'altezza delle note. Occorre considerare tre aspetti: input, modifica e output.
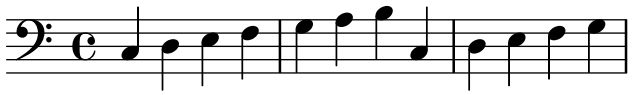
### 1.1.1 Inserimento delle altezze

Questa sezione spiega come indicare l'altezza delle note. Ci sono due modi di collocare le note in una determinata ottava: il modo assoluto e il modo relativo. Nella maggioranza dei casi il modo relativo è più funzionale.

#### Ottava assoluta
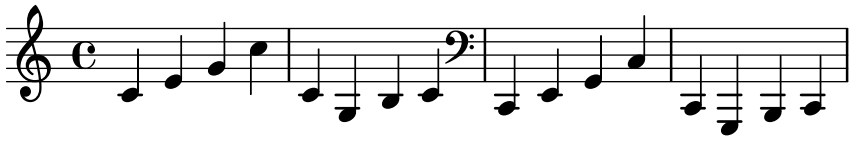
Le altezze, se non si adotta una lingua diversa, sono scritte in notazione olandese, che usa le lettere minuscole dalla `a` (La) alla `g` (Sol). Le note `c` (Do) e `b` (Si) vengono scritte un'ottava sotto il Do centrale.

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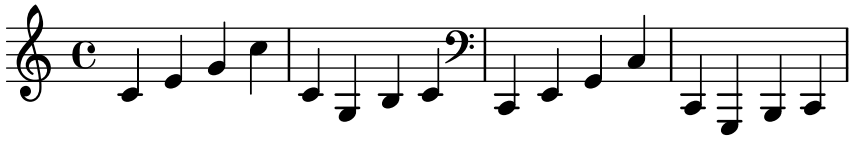


Si possono indicare altre ottave con l'apice singolo (') o la virgola (,). Ogni ' alza l'altezza di un'ottava; ogni , abbassa l'altezza di un'ottava.

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



I normali segni di ottava possono essere inseriti una sola volta se si imposta un'altezza di riferimento dopo `\fixed` e prima della musica. Le altezze inserite in un blocco `\fixed` hanno bisogno dei segni ' o , solo quando si trovano sopra o sotto l'ottava dell'altezza di riferimento.

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



Le altezze dell'espressione musicale che segue `\fixed` non cambiano se racchiuse da un blocco `\relative`, che vedremo tra poco.

## Vedi anche

Glossario musicale: Sezione "Nomi delle altezze" in *Glossario Musicale*.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

## Ottava relativa

L'inserimento delle note con l'ottava assoluta costringe a specificare l'ottava di ogni singola nota. Al contrario, se si usa l'ottava relativa, ogni ottava è determinata dall'ultima nota: se si cambia l'ottava di una nota, cambieranno anche le ottave di tutte le note successive.

La modalità relativa deve essere impostata in modo esplicito col comando `\relative`:

`\relative` *altezza_di_riferimento espressione_musicale*

In modalità relativa ogni nota è collocata il più vicino possibile a quella precedente. Questo significa che l'ottava di ogni altezza all'interno di *espressione_musicale* viene calcolata nel modo seguente:

- In assenza di segni di cambiamento d'ottava, l'ottava di un'altezza viene calcolata in modo che l'intervallo con la nota precedente sia inferiore a una quinta. Tale intervallo è determinato senza considerare gli accidenti.
- Si può aggiungere un segno di cambiamento d'ottava `'` o `,` per collocare l'altezza di una nota all'ottava superiore o inferiore a quella di riferimento.
- È possibile usare più di un segno di cambiamento d'ottava. Per esempio, `''` e `,,` modificano l'altezza di due ottave.
- L'altezza della prima nota è relativa a *altezza_di_riferimento*. *altezza_di_riferimento* è specificato nel modo di ottava assoluta. Quale di queste opzioni è la più conveniente?

  un'ottava del `c`
  : Identificare il Do centrale con `c'` è molto semplice, quindi trovare le ottave del `c` (Do) sarà altrettanto semplice. Se la musica inizia con `gis` sopra `c'''`, si scriverà qualcosa simile a `\relative { gis''' ... }`
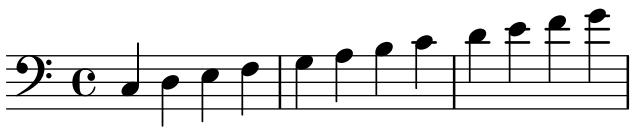
  un'ottava della prima nota
  : Scrivere `\relative { gis''' ... }` è un modo semplice per determinare l'altezza assoluta della prima nota dell'espressione musicale.

  nessuna altezza di riferimento esplicita
  : La forma `\relative {gis''' ... }` è una versione compatta dell'opzione precedente: la prima nota dentro l'espressione musicale è scritta come altezza assoluta. In questo caso equivale a scegliere `f` come altezza di riferimento.

  La documentazione di solito usa l'ultima opzione.
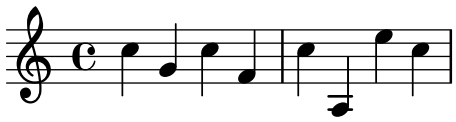
Ecco il modo relativo in azione:

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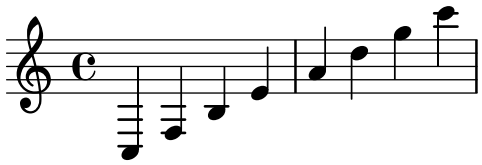


I segni di cambiamento d'ottava si impiegano per gli intervalli più ampi di quello di quarta:

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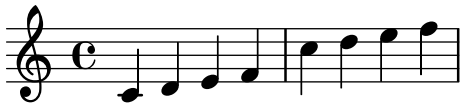


Una sequenza di note senza segni di ottava può tuttavia comprendere intervalli di grande estensione:

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



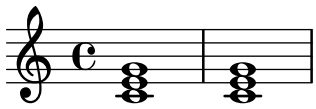
Nel caso di blocchi `\relative` annidati, il blocco `\relative` più interno inizia con la propria altezza di riferimento, indipendentemente dal `\relative` più esterno.

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



`\relative` non ha effetto sui blocchi `\chordmode`.
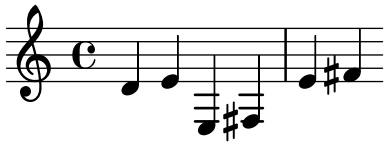
```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` non può essere inserito all'interno dei blocchi `\chordmode`.

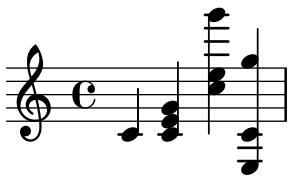
La musica all'interno di un blocco `\transpose` è considerata in notazione d'ottava assoluta, a meno che non sia incluso il blocco `\relative`.

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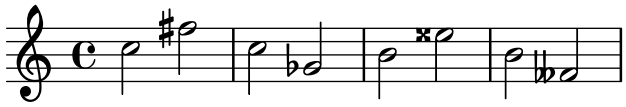


Se l'elemento precedente è un accordo, il posizionamento dell'ottava della nota o dell'accordo che segue è riferito alla prima nota dell'accordo stesso. All'interno degli accordi la nota successiva è sempre relativa a quella precedente. Esaminate con attenzione l'esempio seguente, e in particolare le note `c`.

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



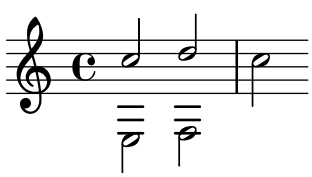
Come spiegato sopra, il riferimento delle altezze a un'ottava è calcolato in base ai soli nomi delle note, senza considerare le alterazioni. Dunque un Mi doppio diesis che segue un Si verrà posizionato sopra, mentre un Fa doppio bemolle sarà posizionato sotto. In altre parole, un intervallo di quarta aumentata due volte viene considerato più piccolo di una quinta diminuita due volte, indipendentemente dal numero di semitoni contenuto in ogni intervallo.

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



In situazioni complesse, può essere utile cambiare l'altezza di riferimento senza tener conto di ciò che è successo prima, usando `\resetRelativeOctave`:

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



### Vedi anche

Glossario musicale: Sezione “quinta” in *Glossario Musicale*, Sezione “intervallo” in *Glossario Musicale*, Sezione “Nomi delle altezze” in *Glossario Musicale*.

Guida alla notazione: [Controlli di ottava], pagina 10.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.
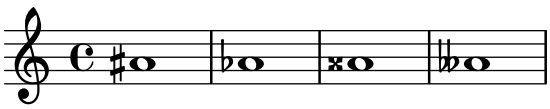
Guida al funzionamento interno: Sezione “RelativeOctaveMusic” in *Guida al Funzionamento Interno*.

## Alterazioni

> **Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note specificano le altezze; le armature e le chiavi determinano come queste altezze debbano essere rappresentate. Una nota non alterata come `c` significa ‘Do naturale’, indipendentemente dall’armatura o dalla chiave. Per maggiori informazioni si veda Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.
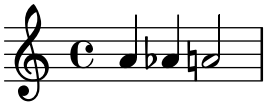
Nella modalità di notazione predefinita un *diesis* si ottiene aggiungendo `is` al nome della nota, un *bemolle* aggiungendo `es`. Come potete immaginare, un *doppio diesis* o *doppio bemolle* si ottengono aggiungendo `isis` o `eses`. Questa sintassi è desunta dalla notazione olandese. Per usare altri nomi per le alterazioni, si veda [Nomi delle note in altre lingue], pagina 8.

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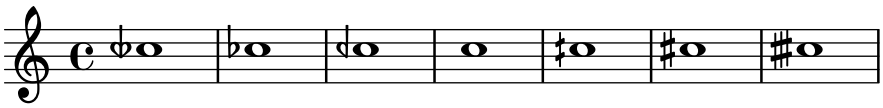


Un’altezza naturale è indicata con il semplice nome della nota; non è richiesto alcun suffisso. Un segno di bequadro apparirà automaticamente quando occorre cancellare l’armatura di chiave o l’effetto di un’alterazione precedente.

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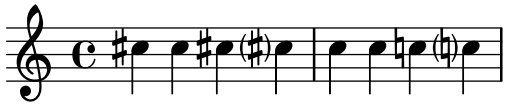


È possibile indicare alterazioni di quarti di tono. Ecco una serie di Do con altezza crescente:

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



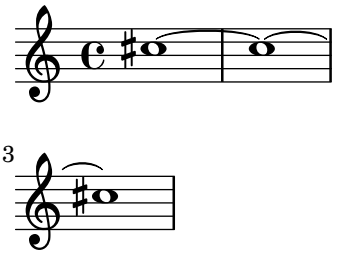
Di norma le alterazioni vengono mostrate automaticamente, ma è possibile anche inserirle manualmente. Si può forzare l’inserimento di un’alterazione di sicurezza aggiungendo il punto esclamativo `!` dopo l’altezza. Un’alterazione di cortesia (ovvero un’alterazione compresa tra parentesi) si ottiene aggiungendo il punto interrogativo `?` dopo l’altezza.

```
\relative c'' { cis cis cis! cis? c c c! c? }
```



Se una nota è prolungata attraverso una legatura di valore, l'alterazione viene ripetuta solo all'inizio di un nuovo sistema:

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

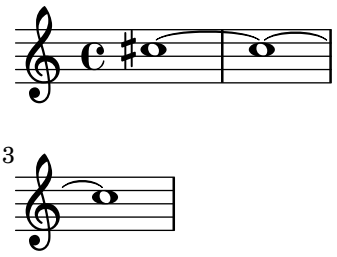


## Frammenti di codice selezionati

*Nascondere le alterazioni delle note con legatura di valore*all'inizio di un nuovo sistema

Questo frammento mostra come nascondere le alterazioni delle note unite alla figura precedente mediante una legatura di valore all'inizio di un nuovo sistema
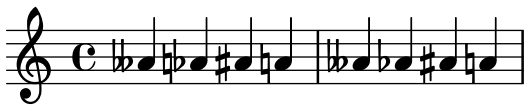
```
\relative c'' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```



*Impedire l'inserimento automatico dei bequadri supplementari*

Secondo le norme tipografiche tradizionali, un segno di bequadro viene inserito prima di un diesis o di un bemolle se un precedente doppio diesis o bemolle sulla stessa nota è cancellato. Per cambiare questo comportamento e seguire la pratica contemporanea, si imposta la proprietà `extraNatural` su `f` (falso) nel contesto `Staff`.

```
\relative c'' {
  aeses4 aes ais a
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



## Vedi anche

Glossario musicale: Sezione “diesis” in *Glossario Musicale*, Sezione “bemolle” in *Glossario Musicale*, Sezione “doppio diesis” in *Glossario Musicale*, Sezione “doppio bemolle” in *Glossario Musicale*, Sezione “Nomi delle altezze” in *Glossario Musicale*, Sezione “quarto di tono” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.

Guida alla notazione: [Alterazioni automatiche], pagina 28, [Alterazioni suggerite (musica ficta)], pagina 450, [Nomi delle note in altre lingue], pagina 8.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Accidental_engraver” in *Guida al Funzionamento Interno*, Sezione “Accidental” in *Guida al Funzionamento Interno*, Sezione “AccidentalCautionary” in *Guida al Funzionamento Interno*, Sezione “accidental-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti
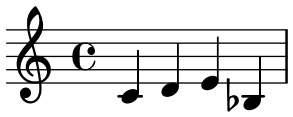
Poiché non esistono standard universalmente accettati per indicare le alterazioni di quarto di tono, i simboli impiegati da LilyPond non si riferiscono ad alcuno standard.

## Nomi delle note in altre lingue

LilyPond comprende insiemi predefiniti di nomi di note e alterazioni in altre lingue. La scelta della lingua si fa solitamente all’inizio del file; l’esempio seguente è scritto in notazione italiana:

```
\language "italiano"

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



Le lingue disponibili e i tipi di notazione che definiscono sono:

| **Lingua** | **Nomi delle note** |
|---|---|
| nederlands | c d e f g a bes b |
| català o catalan | do re mi fa sol la sib si |
| deutsch | c d e f g a b h |
| english | c d e f g a bf/b-flat b |
| español o espanol | do re mi fa sol la sib si |
| français | do ré/re mi fa sol la sib si |
| italiano | do re mi fa sol la sib si |
| norsk | c d e f g a b h |
| português o portugues | do re mi fa sol la sib si |
| suomi | c d e f g a b h |
| svenska | c d e f g a b h |
| vlaams | do re mi fa sol la sib si |
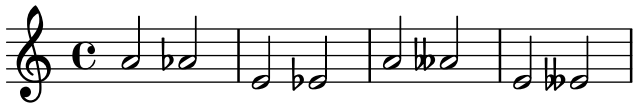
Oltre ai nomi delle note, anche i suffissi per le alterazioni possono variare a seconda della lingua adottata:

| Lingua | diesis | bemolle | doppio diesis | doppio bemolle |
|---|---|---|---|---|
| nederlands | is | es | isis | eses |
| català o catalan | d/s | b | dd/ss | bb |
| deutsch | is | es | isis | eses |
| english | s/-sharp | f/-flat | ss/x/-sharpsharp | ff/-flatflat |
| español o espanol | s | b | ss/x | bb |
| français | d | b | dd/x | bb |
| italiano | d | b | dd | bb |
| norsk | iss/is | ess/es | ississ/isis | essess/eses |
| português o portugues | s | b | ss | bb |
| suomi | is | es | isis | eses |
| svenska | iss | ess | ississ | essess |
| vlaams | k | b | kk | bb |

In olandese, norvegese e finlandese `aes` viene contratto in `as`; in olandese e norvegese, tuttavia, entrambe le forme sono accettate da LilyPond. Lo stesso vale per `es` e `ees`, `aeses` e `ases`, e infine `eeses` e `eses`.

In tedesco e finlandese, LilyPond fornisce anche la forma più frequente `asas` per `ases`.

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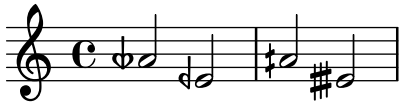


In alcune forme musicali vengono usati i microtoni, le cui alterazioni sono frazioni di un 'normale' diesis o bemolle. La seguente tabella elenca i nomi delle note per le alterazioni di un quarto di tono in tutte le lingue; i prefissi *semi-* e *sesqui-* significano rispettivamente 'metà' e 'uno e mezzo'.

| Lingua | semi-diesis | semi-bemolle | sesqui-diesis | sesqui-bemolle |
|---|---|---|---|---|
| nederlands | ih | eh | isih | eseh |
| català o catalan | qd/qs | qb | tqd/tqs | tqb |
| deutsch | ih | eh | isih | eseh |
| english | qs | qf | tqs | tqf |
| español o espanol | cs | cb | tcs | tcb |
| français | sd | sb | dsd | bsb |
| italiano | sd | sb | dsd | bsb |
| norsk | ih | eh | issih/isih | esseh/eseh |
| português o portugues | sqt | bqt | stqt | btqt |
| suomi | ih | eh | isih | eseh |
| svenska | ih | eh | issih | esseh |
| vlaams | hk | hb | khk | bhb |

In tedesco, esistono simili contrazioni dei nomi per i microtoni così come nelle normali altezze descritte sopra.

```
\language "deutsch"

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



Gran parte delle lingue presentate qui sono comunemente associate alla musica classica occidentale, nota anche come *Common Practice Period*. Sono tuttavia supportati anche altezze e sistemi di accordatura alternativi: si veda Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

### Vedi anche

Glossario musicale: Sezione "Nomi delle altezze" in *Glossario Musicale*, Sezione "Periodo di pratica comune" in *Glossario Musicale*.

Guida alla notazione: Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

File installati: `scm/define-note-names.scm`.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

## 1.1.2 Modifica di più altezze
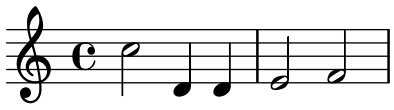
Questa sezione tratta il modo di modificare le altezze delle note.

### Controlli di ottava

In modalità relativa è facile dimenticare un segno di cambiamento d'ottava. I controlli di ottava permettono di rilevare questi errori più facilmente: infatti, generano un avviso e correggono l'ottava se una nota si trova in un'ottava diversa dal previsto.

Per controllare l'ottava di una nota, occorre specificare l'ottava assoluta dopo il simbolo `=`. Questo esempio genererà un avviso (e cambierà l'altezza) perché la seconda nota è l'ottava assoluta `d''` invece di `d'`, come indicato dalla correzione di ottava.
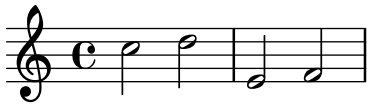
```
\relative {
  c''2 d='4 d
  e2 f
}
```



L'ottava in cui si trovano le note può essere controllata anche col comando `\octaveCheck` *`altezza_di_controllo`*. L'*`altezza_di_controllo`* è specificata in modo assoluto. Questo comando controlla che l'intervallo tra la nota precedente e l'*`altezza_di_controllo`* sia compresa in una quinta (ovvero secondo il normale calcolo della modalità relativa). Se il controllo fallisce, compare un avviso. Benché la nota precedente non sia modificata, le note successive sono relative al valore corretto.

```
\relative {
  c''2 d
```
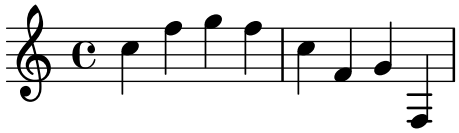
```
  \octaveCheck c'
  e2 f
}
```



Nelle due battute che seguono, il primo e il terzo `\octaveCheck` falliscono, mentre il secondo non fallisce.

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

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



## Vedi anche

Frammenti di codice: Sezione “Pitches” in *snippets*.

Guida al funzionamento interno: Sezione “RelativeOctaveCheck” in *Guida al Funzionamento Interno*.

## Trasposizione

Un’espressione musicale può essere trasposta con `\transpose`. La sintassi è

`\transpose` *`altezza_di_partenza altezza_di_arrivo espressione_musicale`*
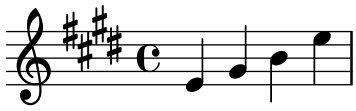
Significa che *`espressione_musicale`* viene trasposto dell’intervallo compreso tra le altezze *`altezza_di_partenza`* e *`altezza_di_arrivo`*: qualsiasi nota che presenti un’altezza corrispondente all’*`altezza_di_partenza`* viene modificata in *`altezza_di_arrivo`*, e qualsiasi altra nota viene trasposta dello stesso intervallo. Entrambe le altezze sono inserite in modalità assoluta.

> **Nota:** La musica all’interno di un blocco `\transpose` è assoluta a meno che il blocco non includa un `\relative`.

Prendiamo come esempio un brano scritto in Re maggiore. Possiamo trasportarlo in Mi maggiore; si noti come anche l’armatura di chiave venga trasposta automaticamente.
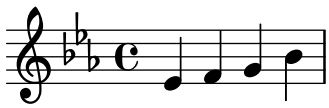
```
\transpose d e {
  \relative {
    \key d \major
    d'4 fis a d
  }
```

```
}
```



Se una parte scritta in Do (l'*intonazione reale* abituale) deve essere suonata su un clarinetto in La (per il quale un La viene rappresentato da un Do e dunque suona una terza minore più basso), la trasposizione sarà ottenuta con:
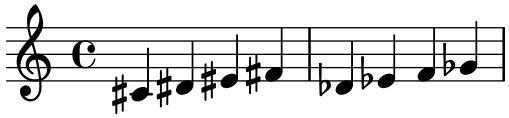
```
\transpose a c' {
  \relative {
    \key c \major
    c'4 d e g
  }
}
```



Si noti che `\key c \major` è specificato esplicitamente. Se non si specifica un'armatura di chiave, le note verranno trasposte ma non apparirà alcuna armatura.

`\transpose` fa distinzione tra altezze enarmoniche: sia `\transpose c cis` che `\transpose c des` traspongono un brano di un semitono più alto. La prima versione mostrerà i diesis e le note rimarranno sullo stesso grado della scala, mentre la seconda versione mostrerà i bemolle sul grado superiore della scala.

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



`\transpose` può essere usato anche in un altro modo, ovvero per inserire note scritte per uno strumento traspositore. Gli esempi precedenti mostrano come inserire altezze in Do (o *intonazione reale*) e mostrare le note di uno strumento traspositore, ma è possibile anche il contrario: per esempio, se da un insieme di parti strumentali si volesse ricavare una partitura per il direttore. Così, per inserire la parte per una tromba in Si bemolle che inizia con un Mi (intonazione reale Re), si può scrivere:

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

Per stampare questa musica in Fa (ad esempio per riarrangiarla per corno) si può avvolgere la musica esistente in un altro `\transpose`:

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

Per maggiori informazioni sugli strumenti traspositori, si veda [Trasporto strumentale], pagina 27.

## Frammenti di codice selezionati

*Trasposizione delle altezze con numero minimo di alterazioni*

Questo esempio usa del codice Scheme per imporre delle modifiche enarmoniche alle note che permettano di avere il numero minimo di alterazioni. In questo caso si applica la seguente regola:

Le doppie alterazioni devono essere eliminate

Si diesis -> Do

Mi diesis -> Fa

Do bemolle -> Si

Fa bemolle -> Mi

In questo modo vengono scelti i suoni enarmonici più semplici.

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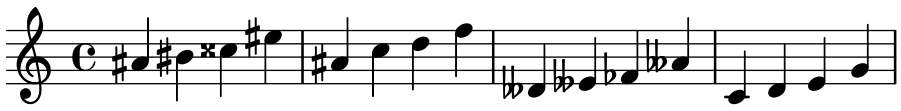
#(define (naturalize music)
   (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
     (if (pair? es)
         (ly:music-set-property!
          music 'elements
          (map naturalize es)))
     (if (ly:music?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

naturalizeMusic =
```

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

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

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



## Vedi anche

Guida alla notazione: [Trasporto strumentale], pagina 27, [Inversione], pagina 14, [Trasposizioni modali], pagina 15, [Ottava relativa], pagina 2, [Retrogradazione], pagina 15.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TransposedMusic” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

La conversione relativa non avrà effetto sulle sezioni `\transpose`, `\chordmode` e `\relative` comprese all’interno di un blocco `\relative`. Per usare la modalità relativa all’interno di musica trasposta, occorre inserire un ulteriore blocco `\relative` all’interno di `\transpose`.

Il comando `\transpose` impedisce di stampare le alterazioni triple. Le sostituisce con un’altezza ‘enarmonicamente equivalente’ (per esempio, Re bemolle al posto di Mi triplo bemolle).
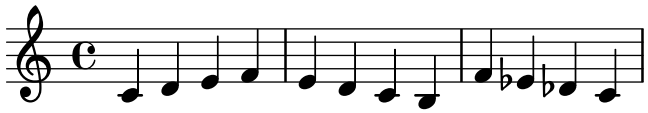
## Inversione

Un’espressione musicale può essere invertita e trasposta in una singola operazione con:

`\inversion` *`altezza-di-riferimento altezza-di-arrivo espressione_musicale`*

L’*`espressione_musicale`* viene invertita intervallo per intervallo intorno all’*`altezza-di-riferimento`* e poi trasposta in modo che ci sia una corrispondenza tra *`altezza-di-riferimento`* e *`altezza-di-arrivo`*.

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



**Nota:** I motivi da invertire devono essere scritti in forma assoluta oppure devono essere prima convertiti in forma assoluta racchiudendoli in un blocco `\relative`.

## Vedi anche
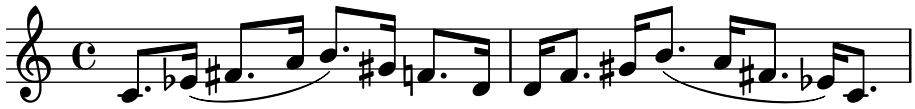
Guida alla notazione: [Trasposizioni modali], pagina 15, [Retrogradazione], pagina 15, [Trasposizione], pagina 11.

## Retrogradazione

Un'espressione musicale può essere invertita in modo da produrre il proprio retrogrado:

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

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



## Problemi noti e avvertimenti

`\retrograde` è uno strumento piuttosto semplice. Dato che molti eventi sono 'riflessi' (come in uno specchio) invece che scambiati, le modifiche manuali e i modificatori direzionali per gli estensori di apertura devono essere aggiunti ai corrispondenti estensori di chiusura: `^(` deve essere terminato da `^)`, ogni `\<` o `\cresc` deve essere terminato da `\!` o `\endcr`, ogni `\>` o `\decr` deve essere terminato da `\enddecr`. Comandi e modifiche che cambiano le proprietà con un effetto duraturo causeranno facilmente delle sorprese.

## Vedi anche

Guida alla notazione: [Inversione], pagina 14, [Trasposizioni modali], pagina 15, [Trasposizione], pagina 11.

## Trasposizioni modali

In una composizione musicale basata su una scala, un motivo viene frequentemente trasportato in differenti modi. Può essere *trasposto* per iniziare in punti diversi della scala o può essere *invertito* rispetto a un punto cardine della scala. Può anche essere rovesciato per produrre il *retrogrado*, si veda [Retrogradazione], pagina 15.

**Nota:** Le note che non si trovano all'interno della scala definita non vengono trasformate.

### *Trasposizione modale*
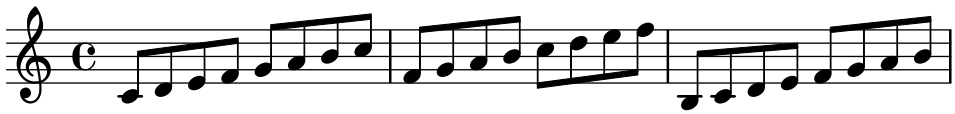
Un motivo può essere trasposto entro una certa scala con:

`\modalTranspose` *`altezza-di-partenza altezza-di-arrivo scala motif`*

Le note di *motif* vengono spostate, se all'interno della *scala*, del numero di gradi della scala dati dall'intervallo tra *altezza-di-arrivo* e *altezza-di-partenza*:

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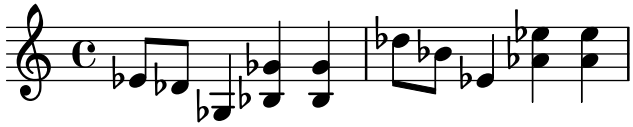
```
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```



È possibile indicare una scala ascendente di qualsiasi lunghezza e con qualsiasi intervallo:

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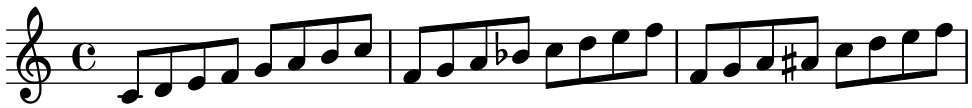
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```



Se usato con una scala cromatica, `\modalTranspose` ha un effetto simile a `\transpose`, con in più la possibilità di specificare i nomi delle note da usare:

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

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



### *Inversione modale*

Una sequenza di note può essere invertita all'interno di una data scala intorno a una determinata nota cardine e quindi trasposto, in un'unica operazione, con:
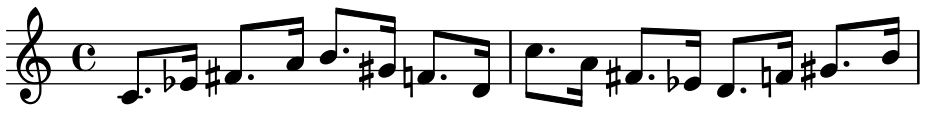
`\modalInversion` *`altezza-cardine altezza-di-arrivo scala motif`*

Le note di *motif* vengono spostate dello stesso numero di gradi dalla nota dell'*altezza-cardine* all'interno della *scala*, ma nella direzione opposta, e il risultato viene poi spostato all'interno della *scala* per il numero di gradi dato dall'intervallo tra *altezza-di-arrivo* e *altezza-cardine*.

Dunque, per invertire intorno a una particolare nota della scala, è necessario usare il medesimo valore per *altezza-cardine* e *altezza-di-arrivo*:

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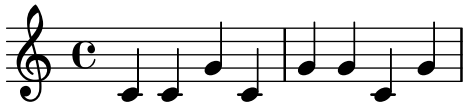
```
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Per invertire intorno a una nota cardine posta tra altre due note, si inverte intorno a una della note e poi si traspone di un grado della scala. Le due note specificate possono essere interpretate come parentesi del punto cardine:

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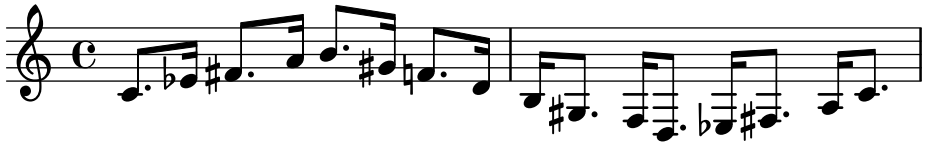
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



L'operazione combinata di inversione e retrogradazione produce la retrogradazione inversa:

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

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



## Vedi anche

Guida alla notazione: [Inversione], pagina 14, [Retrogradazione], pagina 15, [Trasposizione], pagina 11.
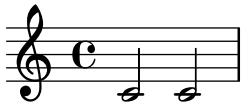
### 1.1.3 Aspetto delle altezze

Questa sezione tratta il modo di modificare l'aspetto delle altezze delle note.

## Chiave
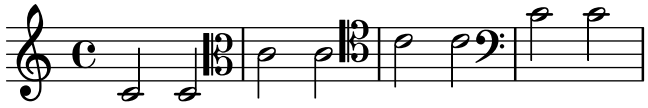
Senza un comando esplicito, la chiave predefinita in LilyPond è la chiave di violino (o di *Sol*).

```
c'2 c'
```



Per cambiare la chiave si usa il comando `\clef` seguito dal nome della chiave. In tutti gli esempi seguenti viene mostrato il *Do centrale*.

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

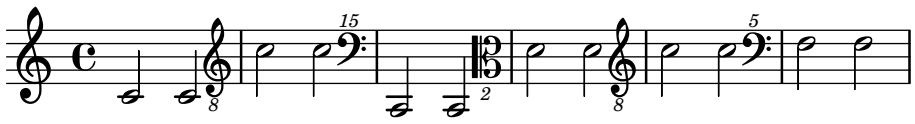


L'elenco completo di tutti i nomi di chiave possibili si trova in Sezione A.10 [Stili della chiave], pagina 702. Chiavi speciali, come quelle usate nella musica *antica*, sono descritte in [Chiavi mensurali], pagina 445, e [Chiavi gregoriane], pagina 452. La musica che utilizza le chiavi dell'intavolatura è discussa in [Intavolature predefinite], pagina 350, e [Intavolature personalizzate], pagina 364.

Per mischiare le chiavi quando si usano le notine, leggere come si usano i comandi `\cueClef` e `\cueDuringWithClef` descritti in [Formattazione delle notine], pagina 217.
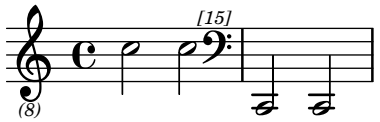
Aggiungendo `_8` o `^8` al nome della chiave, la sua adozione comporta il trasporto all'ottava rispettivamente inferiore o superiore, mentre `_15` e `^15` traspongono di due ottave. È possibile usare altri numeri interi, se necessario. I nomi di chiave contenenti caratteri non alfabetici devono essere racchiusi tra virgolette

```
\clef treble
c'2 c'
\clef "treble_8"
c'2 c'
\clef "bass^15"
c'2 c'
\clef "alto_2"
c'2 c'
\clef "G_8"
c'2 c'
\clef "F^5"
c'2 c'
```



L'ottavazione opzionale si può ottenere racchiudendo l'argomento numerico tra parentesi tonde o quadre:

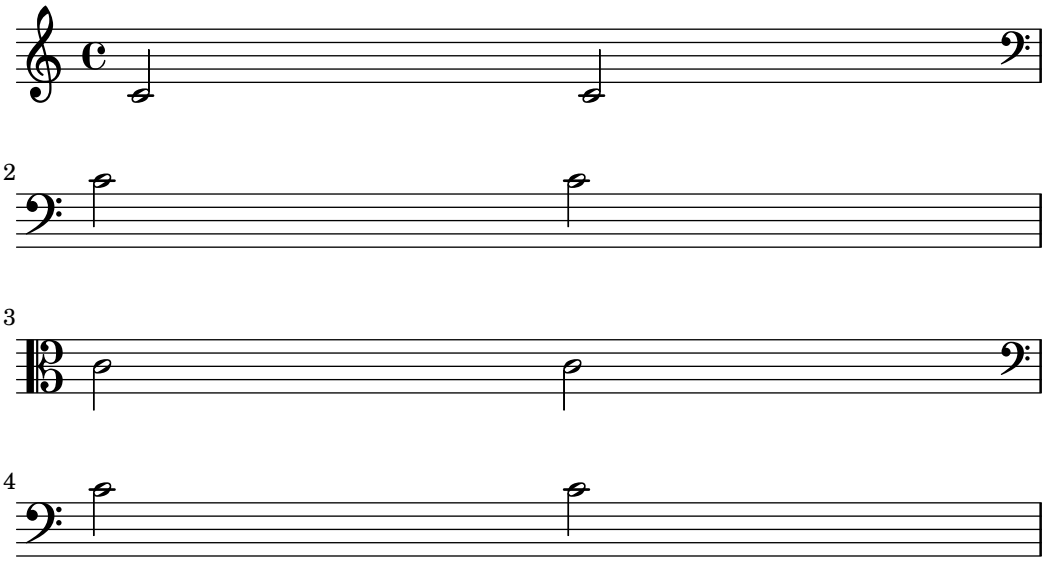
```
\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'
```




Le altezze vengono mostrate come se l'argomento numerico fosse inserito senza parentesi.
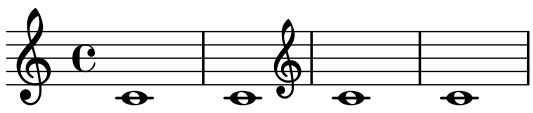
Se c'è un cambio di chiave quando si interrompe la linea, il simbolo della nuova chiave viene ripetuto alla fine della linea precedente, come chiave di *avviso*, e all'inizio di quella successiva. Tale chiave di *precauzione* può essere soppressa.

```
\clef treble { c'2 c' } \break
\clef bass { c'2 c' } \break
\clef alto
  \set Staff.explicitClefVisibility = #end-of-line-invisible
  { c'2 c' } \break
  \unset Staff.explicitClefVisibility
\clef bass { c'2 c' } \break
```




Una chiave che è già stata visualizzata non viene ristampata se viene ripetuto lo stesso comando `\clef` e verrà quindi ignorata. Si può cambiare tale comportamento predefinito col comando `\set Staff.forceClef = ##t`.
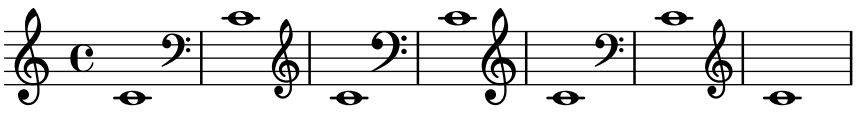
```
\clef treble
c'1
\clef treble
c'1
\set Staff.forceClef = ##t
c'1
\clef treble
c'1
```



Per essere più precisi, non è proprio il comando `\clef` a stampare la chiave. Questo comando imposta o cambia una proprietà dell'incisore `Clef_engraver`, cui spetta la decisione se stampare o meno una chiave nel rigo attuale. La proprietà `forceClef` scavalca questa decisione localmente in modo da ristampare la chiave una volta.

Quando c'è un cambio di chiave manuale, il glifo della chiave cambiata sarà più piccolo del normale. Tale comportamento può essere modificato.

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



## Frammenti di codice selezionati

*Modifiche manuali della proprietà della chiave*

Cambiando il glifo della chiave, la sua posizione o l'ottavazione non cambia la posizione delle note successive nel rigo. Per far sì che le armature di chiave si trovino sulle linee del rigo corrette, bisogna specificare anche `middleCPosition`, con valori positivi o negativi che spostano il `Do centrale` rispettivamente su o giù in senso relativo alla linea centrale del rigo.

Per esempio, `\clef "treble_8"` equivale a impostare `clefGlyph`, `clefPosition` (che regola la posizione verticale della chiave), `middleCPosition` e `clefTransposition`. Viene stampata una chiave quando cambia una di queste proprietà, eccetto `middleCPosition`.

Gli esempi seguenti mostrano le possibilità date dall'impostazione manuale di tali proprietà. Sulla prima linea le modifiche manuali preservano il posizionamento relativo standard di chiavi e note, mentre sulla seconda linea non lo fanno.

```
{
  % The default treble clef
  \key f \major
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  \set Staff.middleCPosition = #6
  \set Staff.middleCClefPosition = #6
  \key g \major
  c'1
  % The baritone clef
  \set Staff.clefGlyph = #"clefs.C"
  \set Staff.clefPosition = #4
  \set Staff.middleCPosition = #4
```

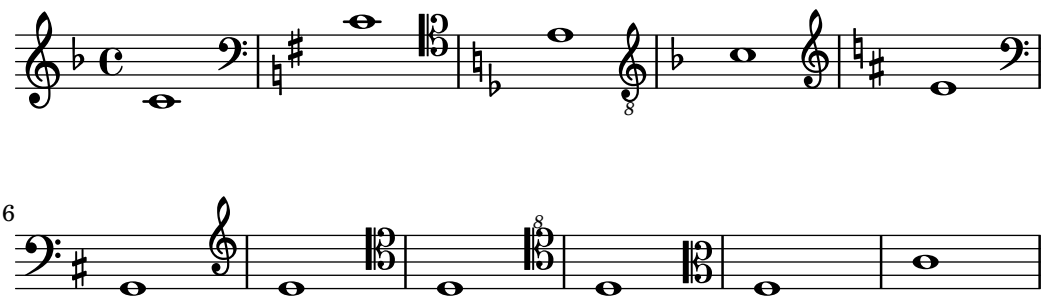
```
  \set Staff.middleCClefPosition = #4
  \key f \major
  c'1
  % The standard choral tenor clef
  \set Staff.clefGlyph = #"clefs.G"
  \set Staff.clefPosition = #-2
  \set Staff.clefTransposition = #-7
  \set Staff.middleCPosition = #1
  \set Staff.middleCClefPosition = #1
  \key f \major
  c'1
  %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

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

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

  % Return to the normal clef:

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



### Vedi anche

Guida alla notazione: [Chiavi mensurali], pagina 445, [Chiavi gregoriane], pagina 452, [Intavolature predefinite], pagina 350, [Intavolature personalizzate], pagina 364, [Formattazione delle notine], pagina 217.

File installati: `scm/parser-clef.scm`.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

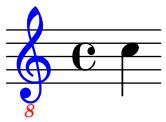
Guida al funzionamento interno: Sezione “Clef_engraver” in *Guida al Funzionamento Interno*, Sezione “Clef” in *Guida al Funzionamento Interno*, Sezione “ClefModifier” in *Guida al Funzionamento Interno*, Sezione “clef-interface” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

I numeri di ottavazione assegnati alle chiavi sono trattati come oggetti grafici separati. Quindi qualsiasi `\override` all’oggetto *Clef* dovrà essere applicato, con un altro `\override`, all’oggetto *ClefModifier*.

```
\new Staff \with {
  \override Clef.color = #blue
  \override ClefModifier.color = #red
}

\clef "treble_8" c'4
```



### Armatura di chiave
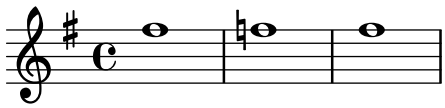
> **Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note costituiscono l’input grezzo; le armature e le chiavi determinano come questo venga mostrato. Una nota non alterata come `c` significa ‘Do naturale’, indipendentemente dall’armatura o dalla chiave. Per maggiori informazioni si veda Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.

L’armatura di chiave indica la tonalità di un brano. È costituita da un insieme di alterazioni (bemolle o diesis) all’inizio del rigo. L’armatura di chiave può essere modificata:

```
\key altezza modo
```

*modo* deve essere `\major` o `\minor` per ottenere rispettivamente un’armatura di *altezza*-maggiore o *altezza*-minore. È anche possibile usare i nomi tradizionali dei modi, chiamati anche *modi ecclesiastici*: `\ionian`, `\dorian`, `\phrygian`, `\lydian`, `\mixolydian`, `\aeolian` e `\locrian`.
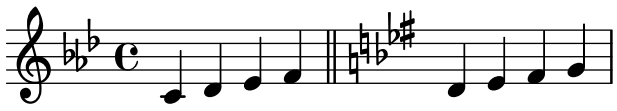
```
\relative {
  \key g \major
  fis''1
  f
  fis
}
```



Si possono definire ulteriori modi elencando le alterazioni per ogni grado della scala quando il modo inizia col Do.

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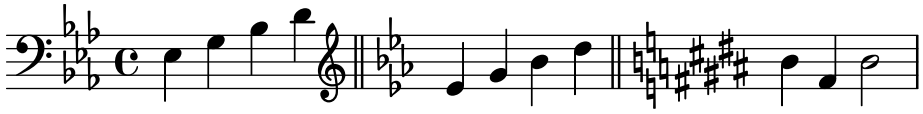
\relative {
  \key c \freygish c'4 des e f
  \bar "||" \key d \freygish d es fis g
}
```



Le alterazioni dell'armatura di chiave possono essere collocate in posizioni diverse da quelle tradizionali o anche in più di un'ottava, usando le proprietà `flat-positions` e `sharp-positions` di `KeySignature`. I valori di queste proprietà specificano l'estensione delle posizioni del rigo in cui potranno comparire le alterazioni. Se viene specificata una sola posizione, le alterazioni vengono collocate entro l'ottava che finisce in quella posizione del rigo.

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

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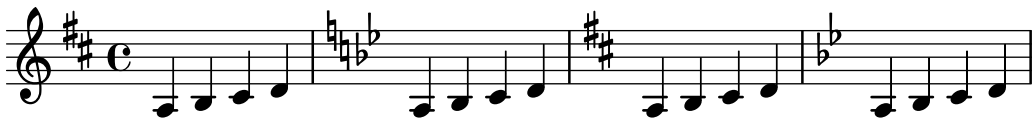


## Frammenti di codice selezionati

*Impedire l'inserimento dei segni di bequadro quando cambia l'armatura di chiave*

Quando l'armatura di chiave cambia, vengono inseriti automaticamente i segni di bequadro per annulare le alterazioni di precedenti armature. Si può evitare questo comportamento impostando su `f` (falso) la proprietà `printKeyCancellation` nel contesto `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
}
```



*Armature di chiave non tradizionali*

Il comando `\key` comunemente usato imposta la proprietà `keyAlterations` del contesto `Staff`. Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente.
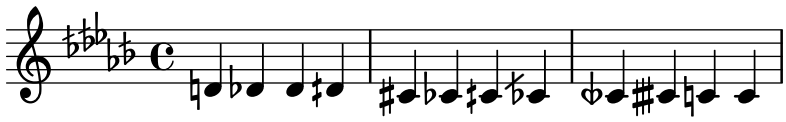
Il formato di questo comando è una lista:

`\set Staff.keyAlterations = #`(((ottava . grado) . alterazione) ((ottava . grado) . alterazione) ...)`

dove, per ogni elemento della lista, `ottava` indica l'ottava (0 è l'ottava dal Do centrale al Si precedente), `grado` indica la nota all'interno dell'ottava (0 significa Do e 6 significa Si) e `alterazione` può essere `,SHARP ,FLAT ,DOUBLE-SHARP` etc.

Altrimenti, usando il formato breve per ogni elemento della lista, `(grado . alterazione)`, ciò indica che la stessa alterazione deve essere presente in tutte le ottave. Per le scale microtonalidove un "diesis" non è 100 centesimi, `alterazione` si riferisce alla proporzione di un duecentesimo di tono intero.

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



## Vedi anche

Glossario musicale: Sezione "church mode" in *Glossario Musicale*, Sezione "scordatura" in *Glossario Musicale*.

Manuale di apprendimento: Sezione "Altezze e armature di chiave" in *Manuale di Apprendimento*.

Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

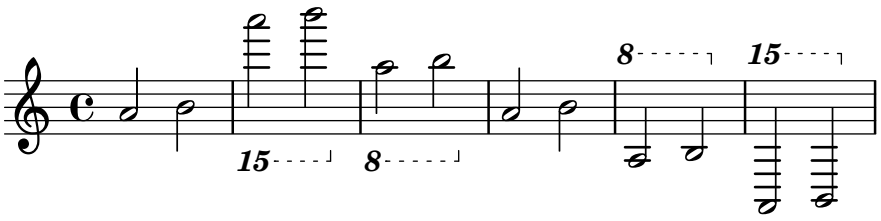
Guida al funzionamento interno: Sezione "KeyChangeEvent" in *Guida al Funzionamento Interno*, Sezione "Key_engraver" in *Guida al Funzionamento Interno*, Sezione "Key_performer" in *Guida al Funzionamento Interno*, Sezione "KeyCancellation" in *Guida al Funzionamento Interno*, Sezione "KeySignature" in *Guida al Funzionamento Interno*, Sezione "key-signature-interface" in *Guida al Funzionamento Interno*.

## Segni di ottavazione

I *segni di ottavazione* introducono un'ulteriore trasposizione di ottava nel rigo:

```
\relative a' {
  a2 b
  \ottava #-2
```

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



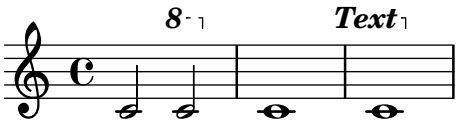

## Frammenti di codice selezionati

*Changing ottava text*

Internally, `\ottava` sets the properties `ottavation` (for example, to `8va` or `8vb`) and `middleCPosition`. To override the text of the bracket, set `ottavation` after invoking `\ottava`.

Short text is especially useful when a brief ottava is used.

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

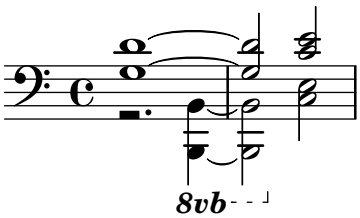



*Aggiungere un segno di ottava a una sola voce*

Se il rigo ha più di una voce, l'ottavazione in una voce trasporrà la posizione delle note in tutte le voci per la durata della parentesi dell'ottava. Se si intende applicare l'ottavazione a una sola voce, si possono impostare esplicitamente middleCPosition e la parentesi di ottava. In questo frammento, la chiave di basso ha di norma il MiddleCPosition impostato su 6, ovvero sei posizioni sopra la linea centrale, dunque nella porzione con l'ottava il MiddleCPosition è più alto di sette posizioni (un'ottava).

```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
```

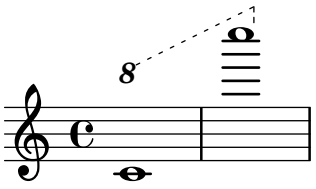
```
    {
      r2.
      \set Staff.ottavation = #"8vb"
      \once \override Staff.OttavaBracket.direction = #DOWN
      \set Voice.middleCPosition = #(+ 6 7)
      <b,,, b,,>4 ~ |
      q2
      \unset Staff.ottavation
      \unset Voice.middleCPosition
      <c e>2
    }
  >>
}
```




*Modificare l'inclinazione dell'estensore dell'ottava*

È possibile cambiare l'inclinazione dell'estensore dell'ottava.

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



### Vedi anche

Glossario musicale: Sezione “ottavazione” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Ottava_spanner_engraver” in *Guida al Funzionamento Interno*, Sezione “OttavaBracket” in *Guida al Funzionamento Interno*, Sezione “ottava-bracket-interface” in *Guida al Funzionamento Interno*.

## Trasporto strumentale
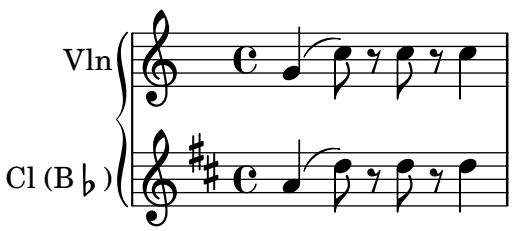
Quando si scrivono partiture che comprendono strumenti traspositori, alcune parti possono essere scritte a un'altezza diversa dall'*intonazione reale*. In questi casi, è necessario specificare la chiave dello *strumento trasportore*, altrimenti l'output MIDI e le citazioni in altre parti produrranno altezze errate. Per maggiori informazioni sulle citazioni, si veda [Citare altre voci], pagina 213.

`\transposition` *altezza*

L'altezza da usare per `\transposition` deve corrispondere al suono effettivamente prodotto quando un `c'` scritto sul rigo viene suonato dallo strumento trasportore. Tale altezza viene inserita in modalità assoluta; dunque, uno strumento che produce un suono reale un tono sopra la notazione deve usare `\transposition d'`. `\transposition` va usato *soltanto* se le altezze *non* sono scritte in intonazione reale.

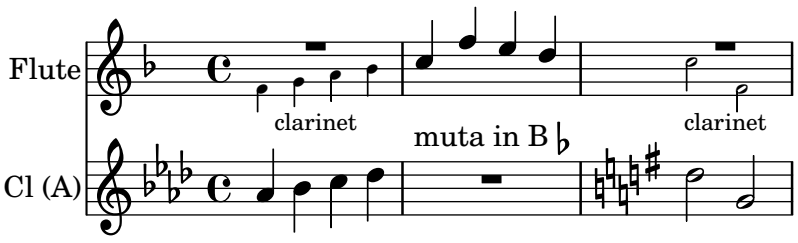
Ecco un frammento per violino e clarinetto in Si bemolle, le cui parti sono inserite usando le note e l'armatura di chiave che appaiono nei rispettivi righi sulla partitura del direttore. I due strumenti suonano all'unisono.

```
\new GrandStaff <<
  \new Staff = "violin" \with {
      instrumentName = "Vln"
      midiInstrument = "violin"
  }
  \relative c'' {
    % non strettamente necessario, ma meglio ricordarlo
    \transposition c'
    \key c \major
    g4( c8) r c r c4
  }
  \new Staff = "clarinet" \with {
    instrumentName = \markup { Cl (B\flat) }
    midiInstrument = "clarinet"
  }
  \relative c'' {
    \transposition bes
    \key d \major
    a4( d8) r d r d4
  }
>>
```



`\transposition` può essere modificato nel corso di un brano. Ad esempio, un clarinettista potrebbe essere costretto a passare da un clarinetto in La a uno in Si bemolle.

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




## Vedi anche

Glossario musicale: Sezione “intonazione reale” in *Glossario Musicale*, Sezione “strumento traspositore” in *Glossario Musicale*.

Guida alla notazione: [Citare altre voci], pagina 213, [Trasposizione], pagina 11.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

## Alterazioni automatiche

Esistono diverse convenzioni sul modo di scrivere le alterazioni. LilyPond ha una funzione per specificare lo stile di gestione delle alterazioni adottato. Questa funzione viene richiamata nel modo seguente:

```
\new Staff <<
  \accidentalStyle voice
  { ... }
>>
```

La gestione delle alterazioni si applica di norma all'attuale `Staff` (con l'eccezione degli stili `piano` e `piano-cautionary`, che sono spiegati dopo). Questa funzione accetta un secondo argomento opzionale che determina in quale ambito debba essere cambiato lo stile. Ad esempio, per usare lo stesso stile in tutti i righi dell'attuale `StaffGroup`, si usa:

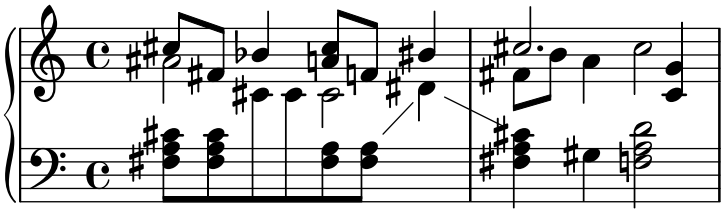
```
\accidentalStyle StaffGroup.voice
```

Sono supportati i seguenti modi di gestire le alterazioni. Il seguente esempio mostra tutti gli stili:

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

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

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

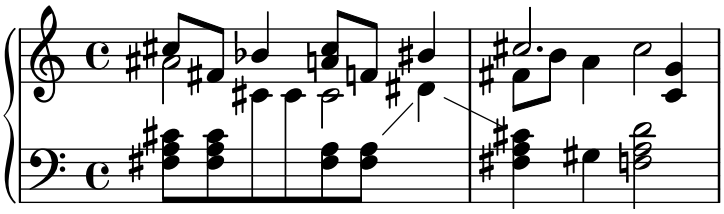


Si noti che le ultime linee di questo esempio possono essere sostituite dal seguente frammento, se si vuole usare lo stesso stile in entrambi i righi.

```
\new PianoStaff {
  <<
    \context Staff = "up" {
      %%% change the next line as desired:
      \accidentalStyle Score.default
      \musicA
    }
    \context Staff = "down" {
      \musicB
    }
  >>
}
```

`default`

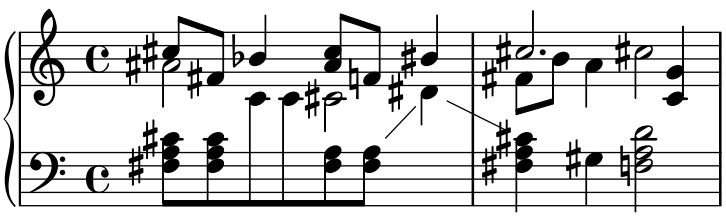
Questo è il comportamento predefinito del compositore tipografico. Corrisponde alla pratica comunemente impiegata dal diciottesimo secolo: le alterazioni vengono ricordate fino alla fine della misura in cui si trovano, limitatamente all'ottava di appartenenza. Quindi, nell'esempio seguente non compare alcun segno di bequadro prima del `b` nella seconda misura o prima dell'ultimo `c`:



`voice`

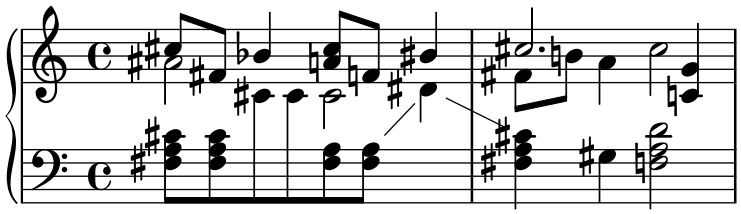
Normalmente le alterazioni mantengono la propria validità a livello di `Staff`. Tuttavia in questo stile le alterazioni vengono gestite individualmente per ogni voce. Al di fuori di quest'aspetto, lo stile è analogo a `default`.

Di conseguenza, le alterazioni relative a una voce non vengono cancellate nelle altre voci. Un risultato spesso non desiderabile: nell'esempio seguente è difficile capire se il secondo `a` sia naturale o diesis. L'opzione `voice` deve essere quindi usata solo se ogni voce è destinata a un esecutore diverso. Se la partitura deve essere letta da un unico musicista (come nel caso della partitura del direttore, o di uno spartito per pianoforte), allora è preferibile usare `modern` o `modern-cautionary`.



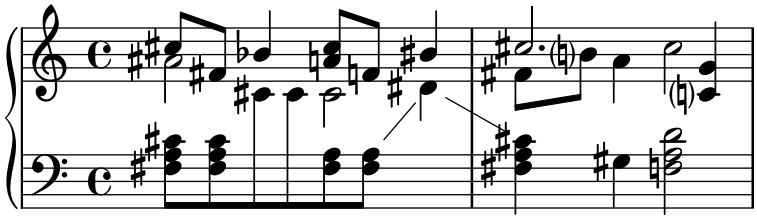
`modern`

Questa regola corrisponde alla pratica comune del ventesimo secolo. Omette i segni di bequadro supplementari che in passato erano di norma anteposti al diesis che segue un doppio diesis o a un bemolle che segue un doppio bemolle. La regola `modern` presenta le stesse alterazioni di `default`, con due aggiunte che servono a evitare ambiguità: i segni di annullamento delle alterazioni temporanee sono anteposti alle note sulla stessa ottava della misura successiva e alle note in ottave diverse nella stessa misura. In questo esempio, dunque, i bequadri del `b` e del `c` nella seconda misura del rigo superiore:



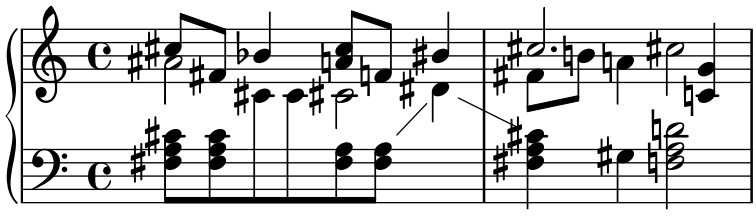
`modern-cautionary`

Questa regola è simile a `modern`, ma le alterazioni 'supplementari' sono segnate come alterazioni di precauzione (con parentesi). La loro dimensione può essere cambiata attraverso la proprietà `font-size` di `AccidentalCautionary`.



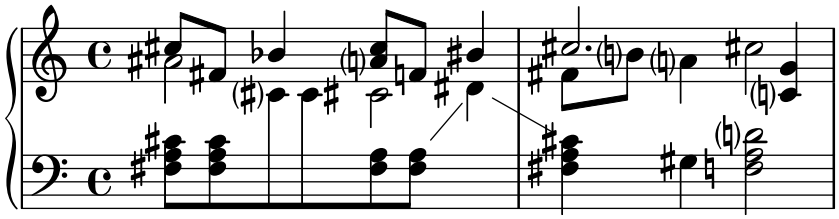
`modern-voice`

Questa regola viene usata per le alterazioni su più voci destinate sia agli esecutori che suonano una singola voce sia a quelli che suonano tutte le voci. Le alterazioni sono mostrate su tutte le voci, ma *sono annullate* su ogni voce dello stesso rigo (`Staff`). Quindi, l'alterazione dell' `a` nell'ultima misura viene annullata perché l'annullamento precedente si trovava in una voce diversa, mentre quella del `d` nel rigo inferiore viene annullata a causa dell'alterazione in un'altra voce della misura precedente:



`modern-voice-cautionary`

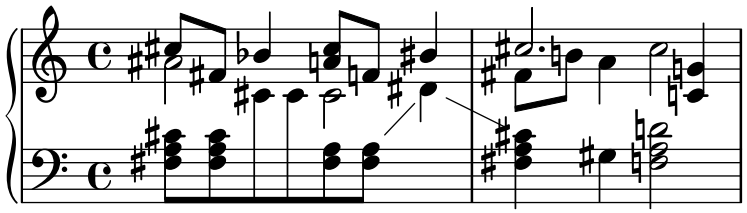
Questa regola è analoga a `modern-voice`, ma con le alterazioni supplementari (quelle non mostrate da `voice`) segnate come alterazioni di precauzione. Tutte le alterazioni mostrate da `default` *sono* mostrate con questa regola, ma alcune di esse sono indicate come alterazioni di precauzione.



`piano`

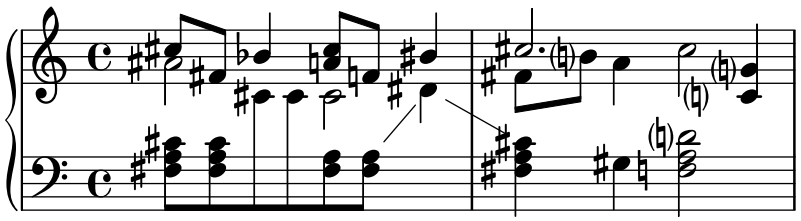
Questa regola riflette la pratica del ventesimo secolo per la notazione per pianoforte. Il suo comportamento è molto simile allo stile `modern`, ma in questo caso le alterazioni vengono annullate in tutti i righi che si trovano nello stesso `GrandStaff` o `PianoStaff`, dunque tutte gli annullamenti delle note finali.

È lo stile predefinito per gli attuali `GrandStaff` e `PianoStaff`.



`piano-cautionary`

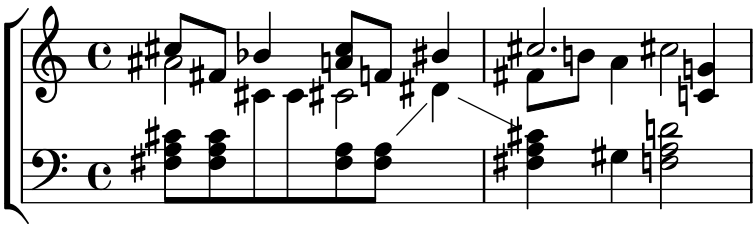
È uguale a `piano` ma con le alterazioni supplementari mostrate come alterazioni di precauzione.



`choral`

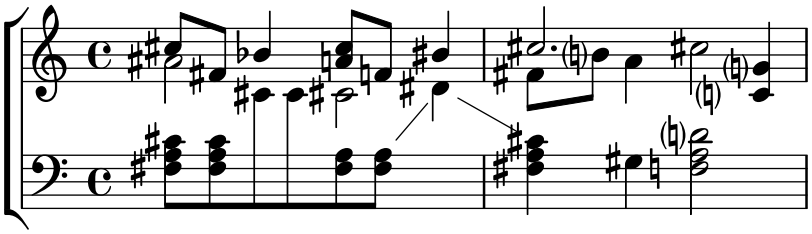
Questa regola è una combinazione degli stili `modern-voice` e `piano`. Mostra tutte le alterazioni richieste dai cantanti che seguono solo la propria voce, e mostra anche le ulteriori alterazioni per tutti i lettori che seguono tutte le voci di un intero `ChoirStaff` contemporaneamente.

Questo stile di alterazioni si applica al `ChoirStaff` corrente per impostazione predefinita.



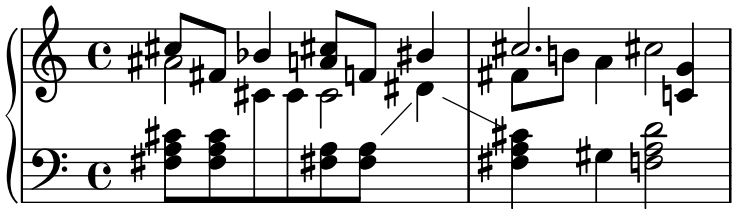
`choral-cautionary`

Questo stile è identico a `choral` ma con le alterazioni extra in forma di alterazioni di cortesia.



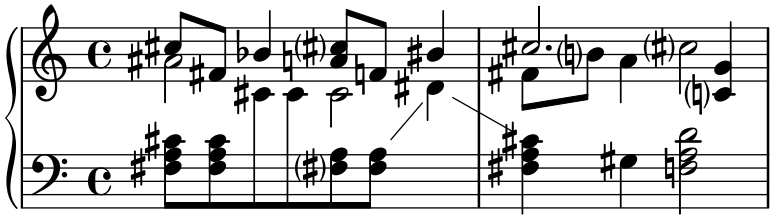
`neo-modern`

Questa regola si riferisce a una pratica tipica della musica contemporanea: le alterazioni sono mostrate come in `modern`, ma vengono ripetute se la stessa nota appare in seguito nella stessa misura – a meno che la seconda occorrenza non segua direttamente la prima.



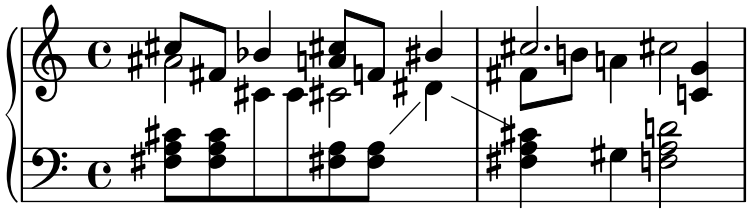
`neo-modern-cautionary`

Questa regola è simile a `neo-modern`, ma le alterazioni 'supplementari' sono mostrate come alterazioni di precauzione (con parentesi). La loro dimensione può essere modificata attraverso la proprietà `font-size` di `AccidentalCautionary`.



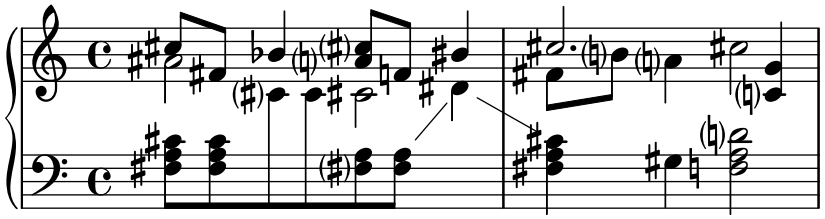
`neo-modern-voice`

Questa regola viene usata per le alterazioni su più di una voce che devono essere lette sia da musicisti che suonano una singola voce sia da musicisti che suonano tutte le voci. Le alterazioni per ogni voce sono mostrate come nello stile `neo-modern`, ma vengono annullate attraverso le voci nello stesso rigo (`Staff`).



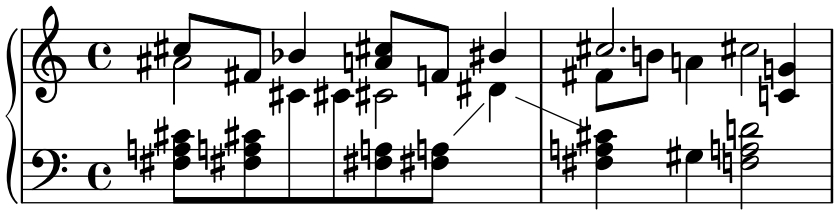
`neo-modern-voice-cautionary`

Questa regola è simile a `neo-modern-voice`, ma le alterazioni supplementari sono indicate come alterazioni di precauzione.



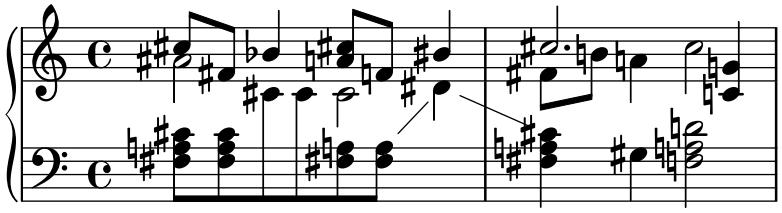
`dodecaphonic`

Questa regola riflette una regola introdotta dai compositori all'inizio del ventesimo secolo nel tentativo di abolire la gerarchia tra suoni naturali e non naturali. Con questo stile, *ogni* nota presenta un segno di alterazione, anche i suoni naturali.



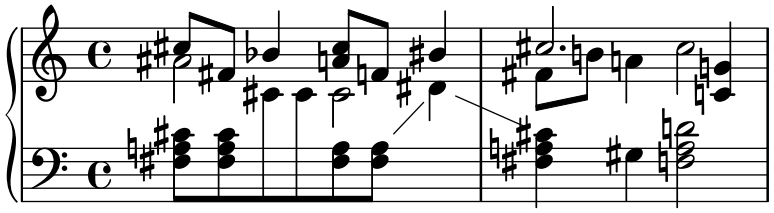
`dodecaphonic-no-repeat`

Come nello stile delle alterazioni dodecafonico *ogni* nota ha un segno di alterazione, ma le alterazioni sono soppresse per tutte le altezze ripetute immediatamente nello stesso rigo.



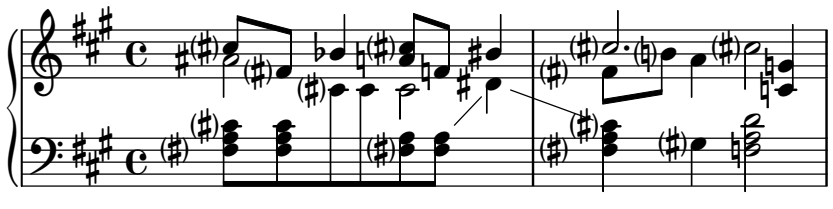
`dodecaphonic-first`

In modo analogo allo stile delle alterazioni dodecafonico *ogni* altezza ha un segno di alterazione, ma solo la prima volta che si incontra in una misura. Le alterazioni vengono ricordate solo per l'ottava corrente ma in tutte le voci.



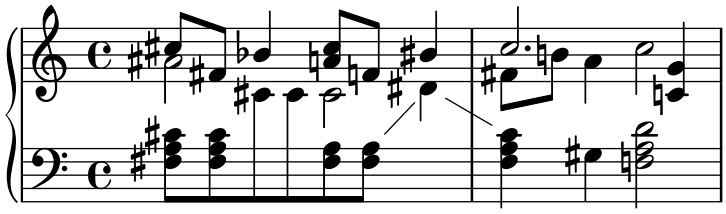
`teaching`

Questa regola è pensata per gli studenti: permette di generare facilmente degli spartiti di scale con le alterazioni di precauzione inserite in modo automatico. Alle alterazioni, indicate come nello stile `modern`, vengono aggiunte ulteriori segni di precauzione per tutti i diesis e bemolle specificati dall'armatura di chiave, fuorché nel caso di ripetizioni immediatamente successive di una stessa nota.



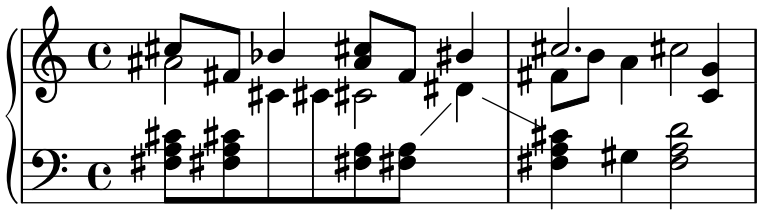
`no-reset`

È identico a `default`, ma le alterazioni mantengono la propria validità 'per sempre', non solo all'interno della singola misura:



`forget`

È il contrario di `no-reset`: le alterazioni non vengono ricordate affatto – pertanto, tutte le alterazioni si riferiscono all'armatura di chiave, indipendentemente dal materiale musicale precedente.



## Vedi anche

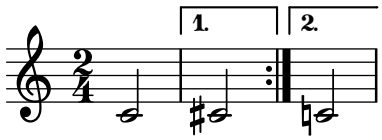
Frammenti di codice: Sezione "Altezze" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Accidental" in *Guida al Funzionamento Interno*, Sezione "Accidental_engraver" in *Guida al Funzionamento Interno*, Sezione "GrandStaff" in *Guida al Funzionamento Interno*, Sezione "PianoStaff" in *Guida al Funzionamento Interno*, Sezione "Staff" in *Guida al Funzionamento Interno*, Sezione "AccidentalSuggestion" in *Guida al Funzionamento Interno*, Sezione "AccidentalPlacement" in *Guida al Funzionamento Interno*, Sezione "accidental-suggestion-interface" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le note simultanee non vengono considerate nell'individuazione automatica delle alterazioni; vengono prese come riferimento solo le note precedenti e l'armatura di chiave. Se la stessa nota occorre simultaneamente con alterazioni diverse, può essere necessario forzare le alterazioni con `!` o `?`: '`<f! fis!>`'.

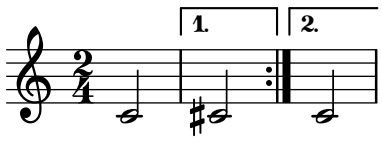
L'annullamento di precauzione delle alterazioni avviene in relazione alla misura precedente. Tuttavia, nel blocco `\alternative` che segue una sezione `\repeat volta N`, è auspicabile che l'annullamento sia calcolato in base alla precedente misura *eseguita*, non alla precedente misura *stampata*. Nell'esempio seguente il Do naturale della seconda volta non richiede il segno di bequadro:




Si può usare il seguente espediente: si definisce una funzione che imposti localmente lo stile delle alterazioni su `forget`:

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

```
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
     cis'
     \forget c'
  }
}
```
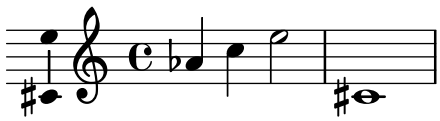



## Ambitus

Il termine *ambitus* (pl. ambitus) indica l'ambito di altezze di una determinata voce all'interno di una composizione musicale. Può indicare anche l'estensione di uno strumento musicale, ovvero l'intera gamma di suoni che può produrre. L'ambitus viene usato nelle parti vocali in modo che gli esecutori possano capire facilmente se siano adeguate alle loro possibilità.

L'ambitus viene indicato all'inizio del brano, prima della chiave iniziale. L'intervallo è individuato graficamente da due teste di nota che rappresentano l'altezza più bassa e più alta. Le alterazioni sono mostrate solo se non fanno parte dell'armatura di chiave.

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

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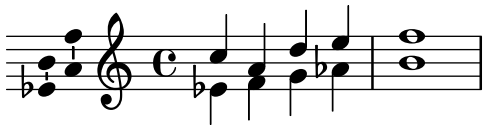


## Frammenti di codice selezionati

*Un ambitus per voce*

L'ambitus può essere specificato per voce. In tal caso occorre spostarlo manualmente per evitare collisioni.

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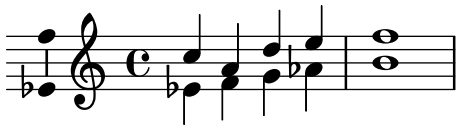
```
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



*Ambitus su più voci*

Se si aggiunge l'incisore `Ambitus_engraver` al contesto `Staff` viene creato un solo ambitus per il rigo, anche nel caso di righi che hanno più voci.

```
\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
  }
>>
```



*Modifica dell'intervallo dell'ambitus*

È possibile cambiare le impostazioni predefinite dell'intervallo tra le teste di nota dell'ambitus e la linea che le collega.

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

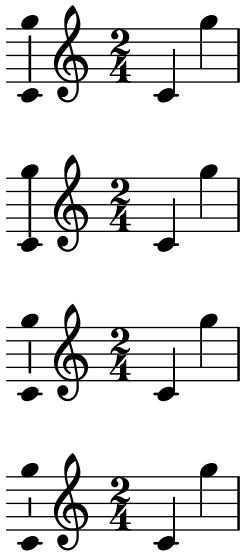
\new Staff {
  \time 2/4
```

```
  % 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''
}
```



## Vedi anche

Glossario musicale: Sezione “ambitus” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Ambitus_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “Ambitus” in *Guida al Funzionamento Interno*, Sezione “AmbitusAccidental” in *Guida al Funzionamento Interno*, Sezione “AmbitusLine” in *Guida al Funzionamento Interno*, Sezione “AmbitusNoteHead” in *Guida al Funzionamento Interno*, Sezione “ambitus-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le collisioni non vengono gestite in presenza di un ambitus multiplo su più di una voce.
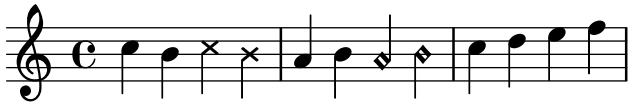
### 1.1.4 Teste di nota

Questa sezione suggerisce i modi in cui modificare la testa di una nota.

#### Teste di nota speciali

L'aspetto delle teste delle note può essere modificato:

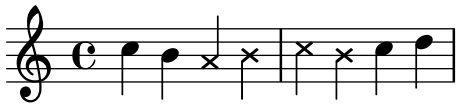
```
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
  \override NoteHead.style = #'harmonic
  a b
  \revert NoteHead.style
  c4 d e f
}
```



L'elenco di tutti gli stili per le teste di nota è in Sezione A.9 [Stili delle teste di nota], pagina 701.

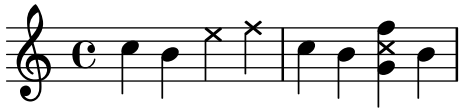
Lo stile barrato (`cross`) viene usato per rappresentare varie intenzioni musicali. I seguenti comandi generici predefiniti modificano la testa della nota nei contesti del rigo e dell'intavolatura e possono essere usati per rappresentare qualsiasi significato musicale:

```
\relative {
  c''4 b
  \xNotesOn
   a b c4 b
  \xNotesOff
  c4 d
}
```



Questo comando può essere usato all'interno e all'esterno degli accordi per generare teste barrate sia nel contesto del rigo che in quello dell'intavolatura:
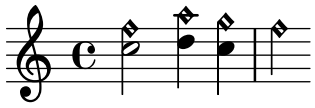
```
\relative {
  c''4 b
  \xNote { e f }
  c b < g \xNote c f > b
}
```



Potete utilizzare, al posto di `\xNote`, `\xNotesOn` e `\xNotesOff`, i comandi `\deadNote`, `\deadNotesOn` e `\deadNotesOff`. Il termine *dead note* è di uso comune tra i chitarristi.

Esiste anche una scorciatoia simile per le forme a diamante:

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



## Comandi predefiniti

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

## Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

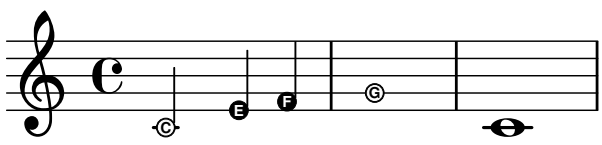
Guida alla notazione: Sezione A.9 [Stili delle teste di nota], pagina 701, [Note in un accordo], pagina 170, [Indicazione di armonici e note smorzate], pagina 393.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Ledger_line_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “LedgerLineSpanner” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*, Sezione “ledger-line-spanner-interface” in *Guida al Funzionamento Interno*.

## Testa di nota con nome della nota

La nota ‘easy play’ inserisce il nome della nota dentro la testa. Viene usata nella musica per principianti. Per rendere le lettere leggibili, occorrerebbe usare un carattere più grande. A questo proposito si veda Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

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



## Comandi predefiniti

`\easyHeadsOn`, `\easyHeadsOff`.

## Frammenti di codice selezionati

*Numeri dentro le teste di nota*

Le teste di nota con nome della nota usano la proprietà `note-names` dell'oggetto `NoteHead` per determinare cosa appaia all'interno della testa. È possibile sovrascrivere questa proprietà e mostrare numeri corrispondenti ai gradi della scala.

Si può creare un semplice incisore che faccia questo per ogni oggetto testa di nota che incontra.

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

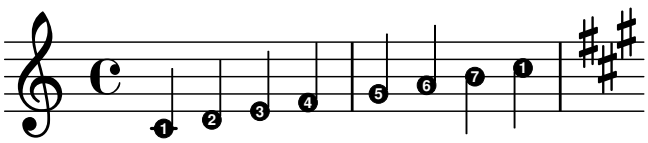
#(set-global-staff-size 26)

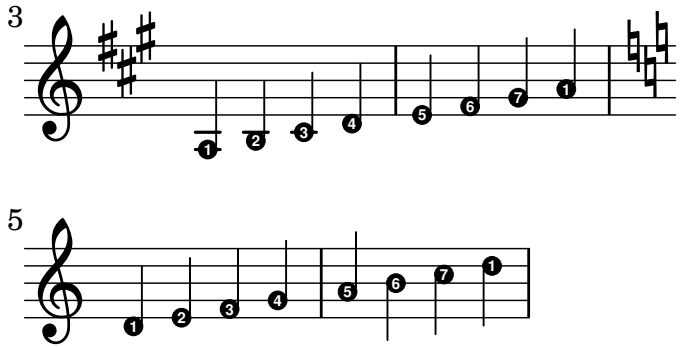
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

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

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

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





## Vedi anche

Guida alla notazione: Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.
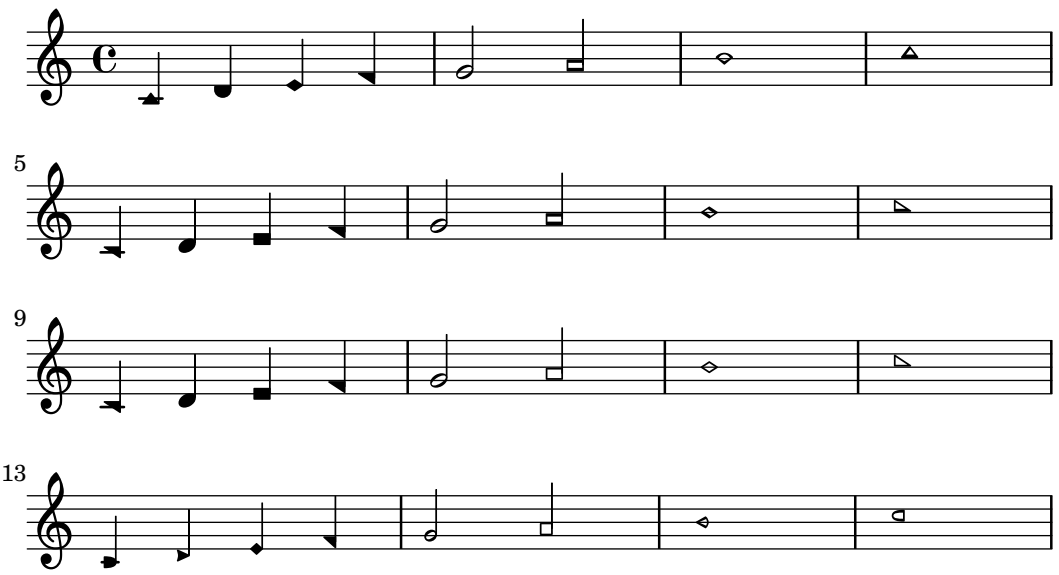
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

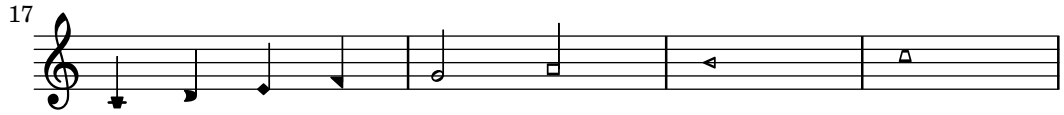
Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

## Teste di nota a forma variabile

In alcune notazioni, la forma della testa della nota corrisponde alla funzione armonica di una nota nella scala. Questa notazione era comune nei canzonieri americani del diciannovesimo secolo. Gli stili possibili sono Sacred Harp, Southern Harmony, Funk (Harmonia Sacra), Walker e Aiken (Christian Harmony):
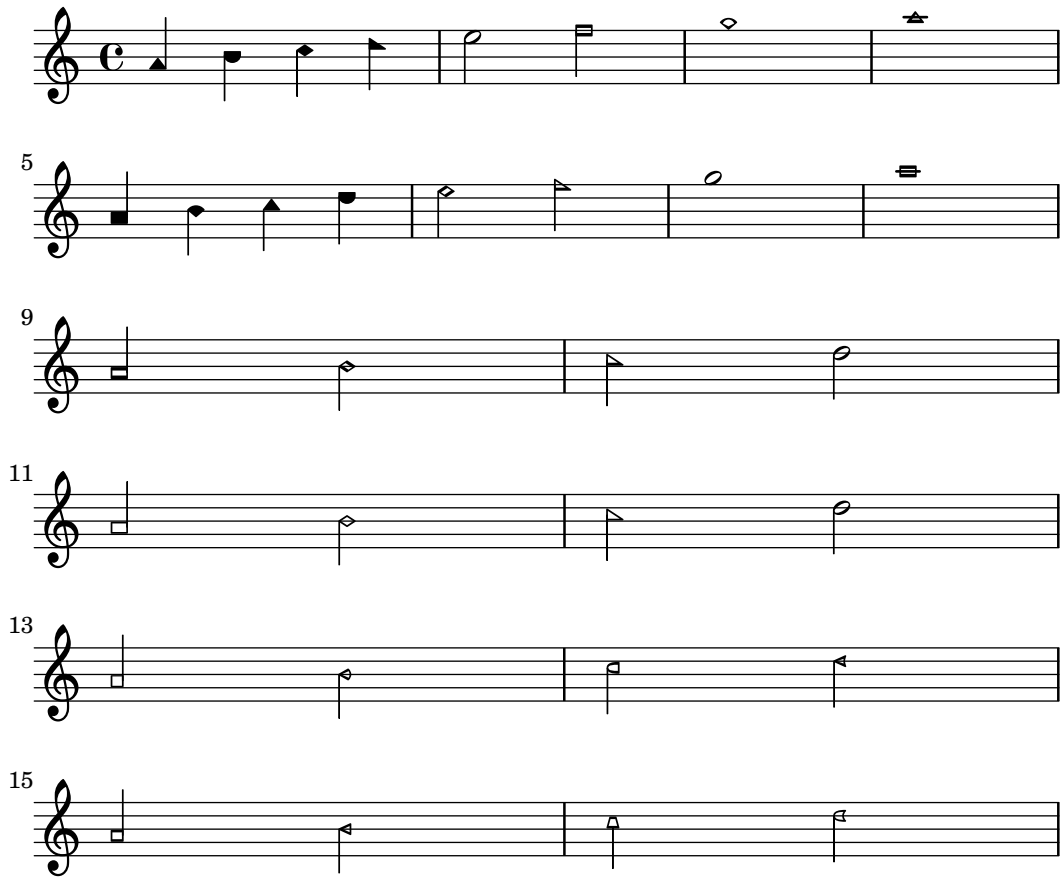
```
\relative c'' {
  \aikenHeads
  c, d e f g2 a b1 c \break
  \sacredHarpHeads
  c,4 d e f g2 a b1 c \break
  \southernHarmonyHeads
  c,4 d e f g2 a b1 c \break
  \funkHeads
  c,4 d e f g2 a b1 c \break
  \walkerHeads
  c,4 d e f g2 a b1 c \break
}
```





Le forme variano in base al grado della scala; la scala è determinata dal comando `\key`. Se si scrive in tonalità minore, il grado della scala può essere determinato in base alla relativa maggiore:

```
\relative c'' {
  \key a \minor
  \aikenHeads
  a b c d e2 f g1 a \break
  \aikenHeadsMinor
  a,4 b c d e2 f g1 a \break
  \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
}
```



## Comandi predefiniti

`\aikenHeads`, `\aikenHeadsMinor`, `\funkHeads`, `\funkHeadsMinor`, `\sacredHarpHeads`, `\sacredHarpHeadsMinor`, `\southernHarmonyHeads`, `\southernHarmonyHeadsMinor`, `\walkerHeads`, `\walkerHeadsMinor`.

## Frammenti di codice selezionati

*Applicazione degli stili delle teste di nota in base al grado della scala*

La proprietà `shapeNoteStyles` può essere usata per definire vari stili di teste di nota per ogni grado della scala (definita dall'armatura di chiave o dalla proprietà `tonic`). Questa proprietà richiede un insieme di simboli, che può essere puramente arbitrario (sono permesse espressioni geometriche come `triangle`, `cross` e `xcircle`) o basato sull'antica tradizione tipografica americana (sono consentiti anche alcuni nomi di nota latini).

Detto questo, per imitare gli antichi canzionieri americani, ci sono vari stili predefiniti disponibili attraverso dei comodi comandi come `\aikenHeads` o `\sacredHarpHeads`.
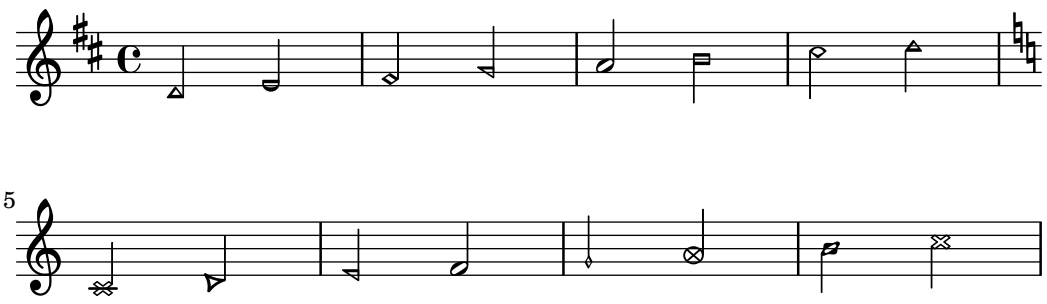
Questo esempio mostra modi diversi di ottenere teste di nota di varie forme e illustra la possibilità di trasporre una melodia senza perdere la corrispondenza tra le funzioni armoniche e gli stili delle teste.

```
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
  }
}
```




La lista completa di tutti gli stili delle teste si trova in Sezione A.9 [Stili delle teste di nota], pagina 701.

### Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida alla notazione: Sezione A.9 [Stili delle teste di nota], pagina 701.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

### Improvvisazione

L’improvvisazione viene talvolta indicata con teste tagliate: l’esecutore può scegliere qualsiasi nota ma deve seguire il ritmo indicato. Si possono creare queste teste:

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



### Comandi predefiniti

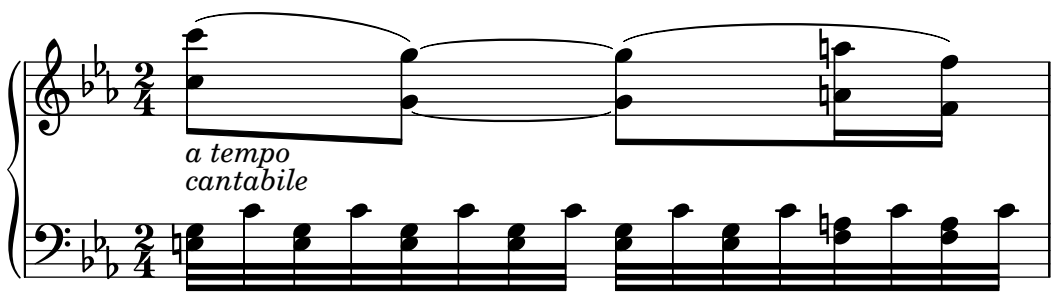
`\improvisationOn`, `\improvisationOff`.
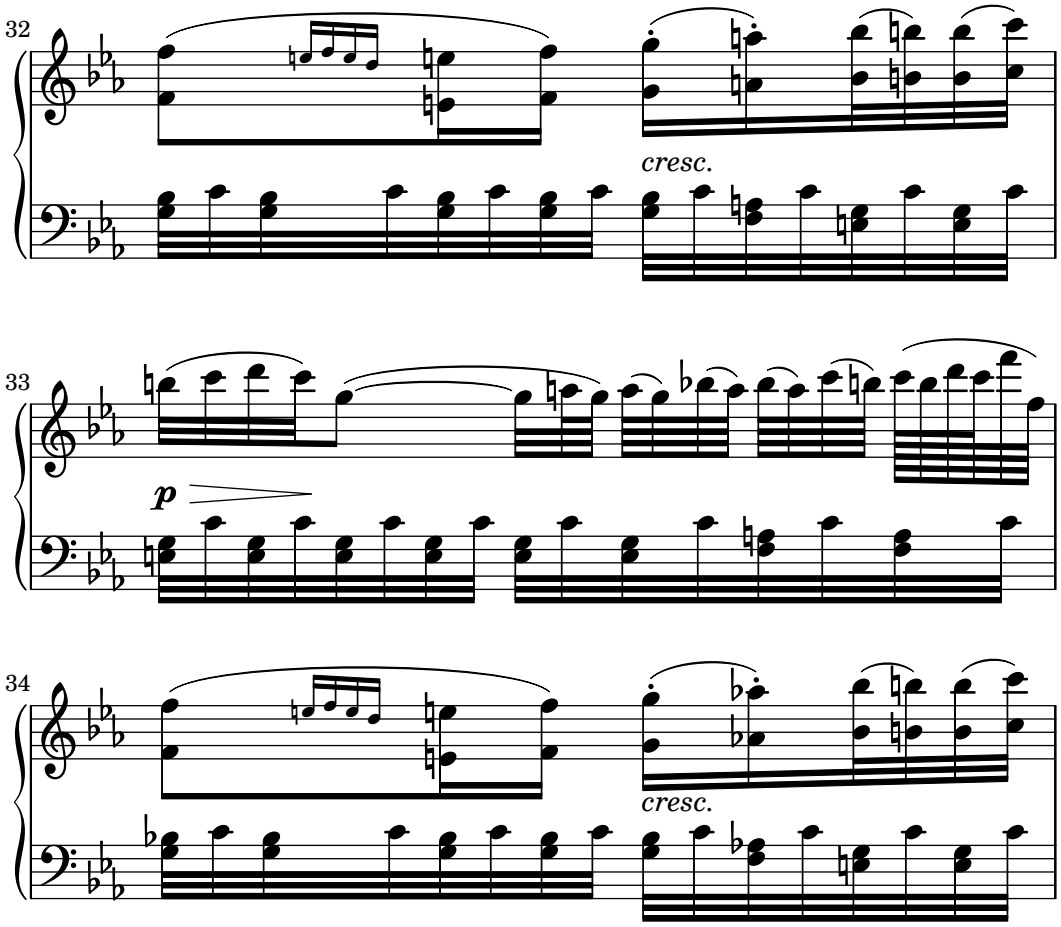
### Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*.

## 1.2 Ritmi






Questa sezione tratta i ritmi, le pause, le durate, la disposizione delle travature e le battute.
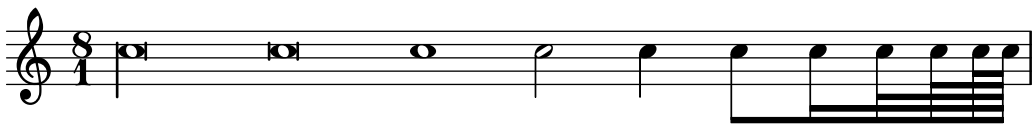
### 1.2.1 Inserimento delle durate

#### Durata

Le durate delle note si inseriscono usando numeri e punti. Il numero è basato sul valore reciproco della lunghezza della nota. Per esempio, una nota di un quarto si indica col valore numerico di `4` perché il suo valore è 1/4, mentre una minima si indica col `2`, la croma con l'`8` e così via. La minor durata esprimibile per una nota indipendente è di `128`; è possibile inserire anche valori inferiori, ma solo all'interno di travature. Vedi anche Sezione 1.2.4 [Travature], pagina 85.

Per le note più lunghe di un intero si usano i comandi `\longa` (due volte una breve) e `\breve`. Solo nella notazione per musica antica è possibile specificare una nota che dura quattro volte una breve, attraverso il comando `\maxima`. Vedi Sezione 2.9 [Notazione antica], pagina 440.
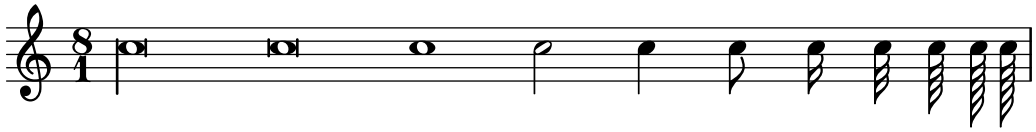
```
\relative {
  \time 8/1
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```



Ecco gli stessi valori con la disposizione automatica delle travature disabilitata.
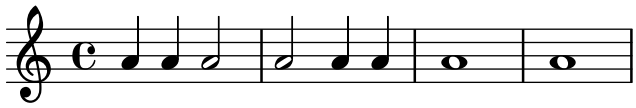
```
\relative {
  \time 8/1
```

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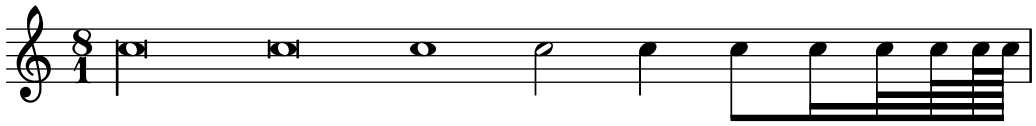


Le durate isolate, ovvero le durate prive di altezza, in una sequenza musicale prenderanno la loro altezza dalla nota o accordo precedenti.

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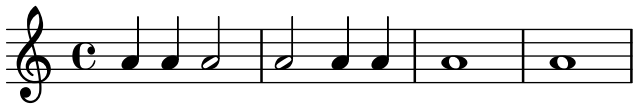


Le durate isolate prendono l'altezza dalla nota o accordo precedenti.

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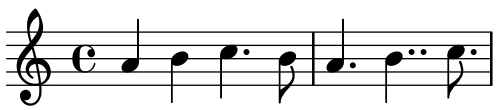


Le altezze isolate, ovvero le altezze prive di durata, in una sequenza musicale prenderanno la loro durata dalla nota o accordo precedenti. Se non c'è alcuna durata precedente, il valore predefinito è sempre 4, una nota di un quarto.

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



Inserire un punto (`.`) dopo la durata per ottenere note "puntate". Le figure con doppio punto si indicano aggiungendo due punti, e così via.

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



Per evitare collisioni con le linee del rigo, di norma i punti delle note sono spostati in su. Tuttavia all'interno di passaggi polifonici possono essere posizionati manualmente, sopra o sotto il rigo a seconda delle necessità. Vedi Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Alcune durate non possono essere rappresentate usando solo durate e punti ma soltanto tramite legature di valore tra due o più note. Vedi [Legature di valore], pagina 55.

Per sapere come specificare durate che allineino le sillabe del testo alle note, si veda Sezione 2.1 [Musica vocale], pagina 267.

Le note possono essere distanziate in modo proporzionale alla loro durata. Vedi Sezione 4.5.5 [Notazione proporzionale], pagina 582.

## Comandi predefiniti

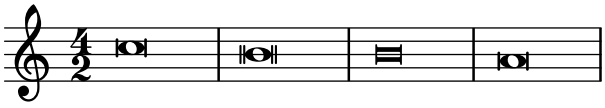
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

## Frammenti di codice selezionati

*Note brevi alternative*

Le note brevi sono disponibili anche con due linee verticali su ciascun lato della testa invece di una sola e in stile barocco.
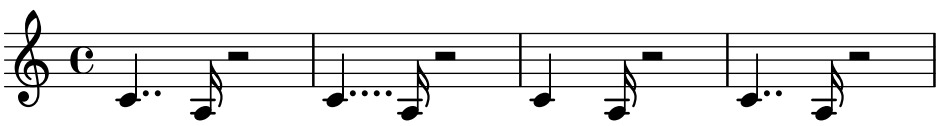
```
\relative c'' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}
```



*Modifica del numero di punti di aumentazione per nota*

Il numero di punti di aumentazione su una singola nota può essere modificato in modo indipendente dai punti posizionati dopo la nota.

```
\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 |
}
```



## Vedi anche

Glossario musicale: Sezione “breve” in *Glossario Musicale*, Sezione “longa” in *Glossario Musicale*, Sezione “maxima” in *Glossario Musicale*, Sezione “durata” in *Glossario Musicale*, Sezione “Nomi di durata delle note e delle pause” in *Glossario Musicale*.

Guida alla notazione: Sezione 1.2.4 [Travature], pagina 85, [Legature di valore], pagina 55, [Gambi], pagina 232, Sezione 1.2.1 [Inserimento delle durate], pagina 46, Sezione 1.2.2 [Inserimento delle pause], pagina 59, Sezione 2.1 [Musica vocale], pagina 267, Sezione 2.9 [Notazione antica], pagina 440, Sezione 4.5.5 [Notazione proporzionale], pagina 582.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Dots" in *Guida al Funzionamento Interno*, Sezione "DotColumn" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non c'è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato, dunque si possono indicare pause comprese tra `128` e `\maxima`.

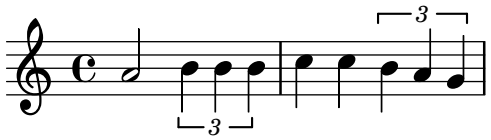
## Gruppi irregolari

I gruppi irregolari sono costituiti da un'espressione musicale introdotta dal comando `\tuplet`, che moltiplica la velocità dell'espressione musicale per una frazione:

```
\tuplet frazione { musica }
```
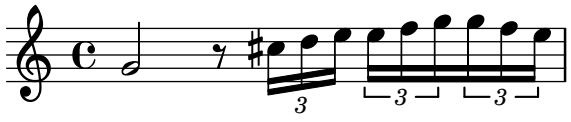
Il numeratore della frazione apparirà sopra o sotto le note; eventualmente, con l'aggiunta opzionale di una parentesi quadra. Il gruppo irregolare più comune è la terzina, in cui 3 note hanno la durata di 2:

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



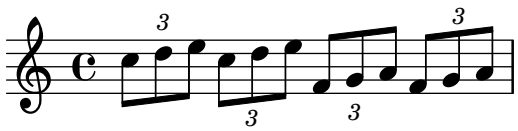

In caso di lunghi passaggi di gruppi irregolari, dover scrivere un comando `\tuplet` per ogni gruppo è scomodo. È possibile specificare direttamente la durata di un gruppo irregolare prima della musica per far sì che i gruppi siano suddivisi automaticamente:

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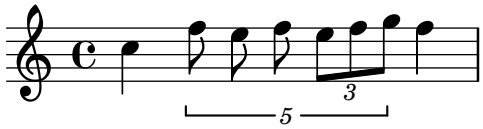



Le parentesi dei gruppi irregolari si possono posizionare manualmente sopra o sotto il rigo:

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



È possibile annidare i gruppi irregolari:

```
\relative {
  \autoBeamOff
  c''4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4
}
```




La modifica di gruppi irregolari annidati che iniziano simultaneamente richiede l'uso di `\tweak`.

Per modificare la durata delle note senza che appaia la parentesi quadra del gruppo irregolare, si veda [Scalare le durate], pagina 53.

## Comandi predefiniti

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

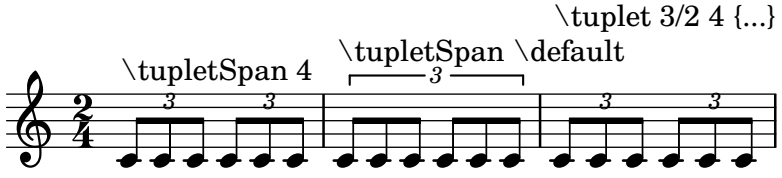
## Frammenti di codice selezionati

*Inserire vari gruppi irregolari usando una sola volta il comando* `\tuplet`

La proprietà `tupletSpannerDuration` imposta la durata di ognuno dei gruppi irregolari compresi tra parentesi dopo il comando `\tuplet`. In questo modo si possono inserire molti gruppi irregolari consecutivi all'interno di una singola espressione `\tuplet`, risparmiando così tempo e spazio.

Ci sono vari modi per impostare `tupletSpannerDuration`. Il comando `\tupletSpan` la imposta su una certa durata e poi la annulla quando invece di una durata viene specificato `\default`. Altrimenti si può usare un argomento opzionale con `\tuplet`.

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

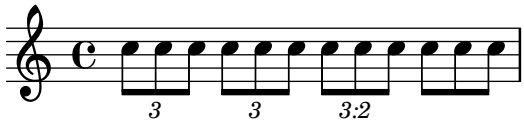



*Modifica del numero del gruppo irregolare*

Di norma compare sulla parentesi del gruppo irregolare solo il numeratore del numero del gruppo irregolare, ovvero il numeratore dell'argomento del comando `\tuplet`. Ma è possibile mostrare la frazione *num*:*den* del numero del gruppo irregolare oppure nascondere del tutto il numero.

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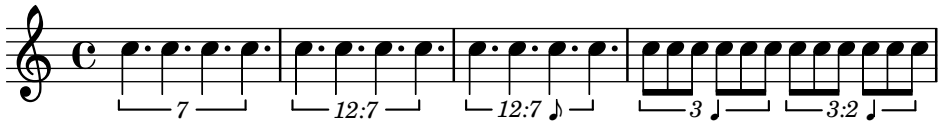
```
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```
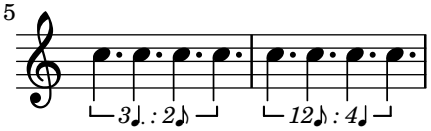



*Numeri non predefiniti per i gruppi irregolari*

LilyPond fornisce anche funzioni di formattazione che permettono di creare numeri di gruppi irregolari diversi dalla frazione vera e propria, così come di aggiungere un valore di nota al numero o alla frazione di un gruppo irregolare.

```
\relative c'' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7)
      (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::fraction-with-notes
      (ly:make-duration 2 1) (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-fraction-with-notes 12
      (ly:make-duration 3 0) 4 (ly:make-duration 2 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
}
```





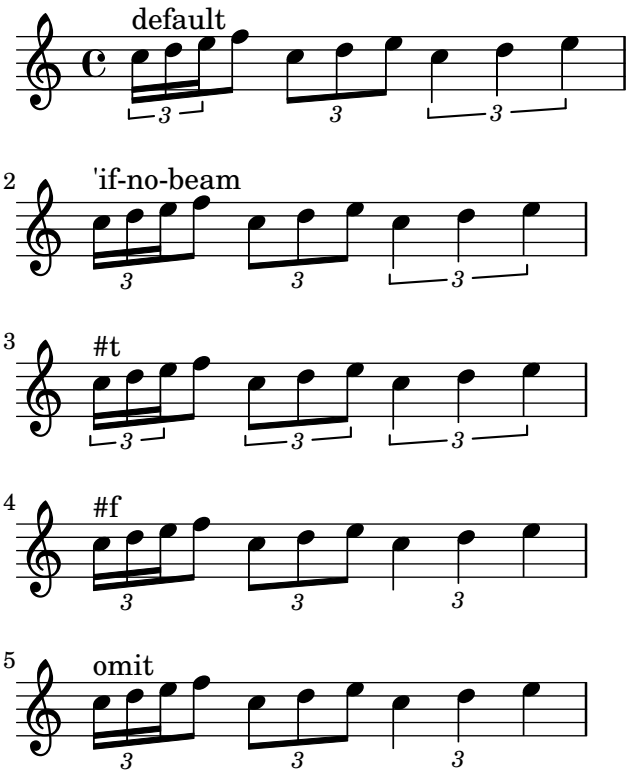
*Controllare la visibilità della parentesi del gruppo irregolare*

Il comportamento predefinito relativo alla visibilità della parentesi quadra del gruppo irregolare è di mostrare una parentesi a meno che non ci sia una travatura della stessa lunghezza del gruppo.

Per controllare la visibilità di tale parentesi, si imposta la proprietà `'bracket-visibility` su `#t` (mostra sempre la parentesi), `#'if-no-beam` (mostra la parentesi solo se non c'è una travatura, che è il comportamento predefinito) o `#f` (non mostrare mai la parentesi). L'ultima opzione equivale a omettere l'oggetto @code{TupletBracket} dall'output.

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

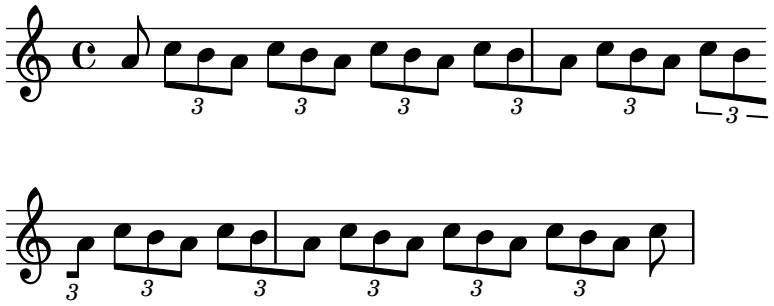
\new Voice {
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music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



*Consentire l'interruzione del rigo all'interno di gruppi irregolari con travature*

Questo esempio artificioso mostra come permettere interruzioni del rigo sia manuali che automatiche all'interno di un gruppo irregolare con travature. Si noti che le travature di questi gruppi irregolari fuori dal ritmo devono essere disposte manualmente.

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



## Vedi anche

Glossario musicale: Sezione "terzina" in *Glossario Musicale*, Sezione "gruppo irregolare" in *Glossario Musicale*, Sezione "polimetrico" in *Glossario Musicale*.

Manuale di apprendimento: Sezione "Metodi di modifica" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Gestione del tempo], pagina 122, [Scalare le durate], pagina 53, Sezione 5.3.4 [Il comando tweak], pagina 619, [Notazione polimetrica], pagina 78.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TupletBracket" in *Guida al Funzionamento Interno*, Sezione "TupletNumber" in *Guida al Funzionamento Interno*, Sezione "TimeScaledMusic" in *Guida al Funzionamento Interno*.
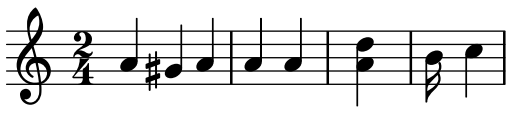
## Scalare le durate

La durata di singole note, pause o accordi può essere moltiplicata per una frazione `N/M` aggiungendo `*N/M` (o `*N` se `M` è 1). Questo non cambierà l'aspetto delle note o delle pause, ma la durata così alterata verrà utilizzata per calcolare la posizione all'interno della misura e per impostare la durata nel file MIDI. Si possono combinare molteplici fattori, come `*L*M/N`. I fattori fanno

parte della durata: quindi se non si specifica una durata per le note successive, la durata ripresa dalla nota precedente includerà il fattore di scalatura.

Nell'esempio seguente le prime tre note occupano esattamente due tempi, ma non sono indicate come gruppo irregolare.
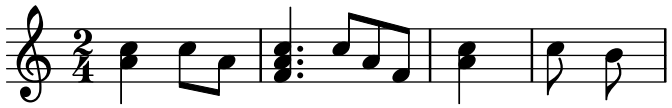
```
\relative {
  \time 2/4
  % Trasforma le durate in terzine
  a'4*2/3 gis a
  % Durate normali
  a4 a
  % Raddoppia la durata dell'accordo
  <a d>4*2
  % Durata di un quarto, ma appare come un sedicesimo
  b16*4 c4
}
```



Anche la durata delle pause spaziatrici può essere modificata con un moltiplicatore. Può essere utile per saltare molte misure; per esempio `s1*23`.

Frammenti musicali più lunghi possono essere compressi secondo la stessa proporzione, come moltiplicando ogni nota, accordo o pausa per una medesima frazione. In questo modo, l'aspetto della musica non cambia ma la durata interna delle note viene moltiplicata per la frazione *num*/*den*. Ecco un esempio che mostra come la musica possa essere compressa e espansa:

```
\relative {
  \time 2/4
  % Durate normali
  <c'' a>4 c8 a
  % Scala la musica di *2/3
  \scaleDurations 2/3 {
    <c a f>4. c8 a f
  }
  % Scala la musica di *2
  \scaleDurations 2/1 {
    <c' a>4 c8 b
  }
}
```



Questo comando torna utile nella notazione polimetrica, si veda [Notazione polimetrica], pagina 78.

## Vedi anche

Guida alla notazione: [Gruppi irregolari], pagina 49, [Pause invisibili], pagina 61, [Notazione polimetrica], pagina 78.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il calcolo della posizione in una misura deve considerare tutti i fattori di dimensionamento applicati alle note di quella misura e gli esigui residui delle misure precedenti. Questo calcolo viene fatto con numeri razionali. Se un numeratore o un denominatore intermedi in quel calcolo eccedono di 2^30, l'esecuzione e la composizione tipografica si arresteranno in quel punto senza indicare un errore.
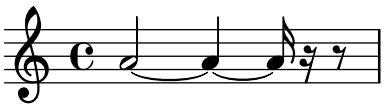
## Legature di valore

Una legatura di valore connette le teste di due note della stessa altezza successive. Dunque, la legatura di valore prolunga la durata di una nota.

> **Nota:** Le legature di valore non devono essere confuse con le *legature di portamento*, che articolano un passaggio, o con le *legature di frase*, che delimitano una frase musicale. Una legatura di valore serve semplicemente a prolungare la durata di una nota, in modo analogo al punto di valore.
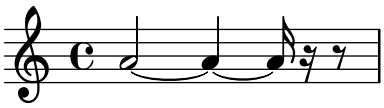
La legatura di valore si inserisce aggiungendo il simbolo tilde (~) alla prima di ogni coppia di note legate. Esso indica che la nota deve essere legata alla nota successiva, che deve essere della stessa altezza.

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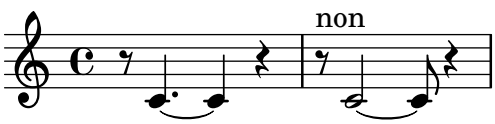


Le legature di valore possono avvantaggiarsi dell'interpretazione dell' 'ultima altezza esplicita' per le durate isolate:

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



Le legature di valore si usano per unire due note a cavallo di una stanghetta di battuta, oppure quando non si possono usare i punti per esprimere una particolare durata. Le legature si dovrebbero usare anche per unire note dalle durate superiori all'unità di suddivisione della misura:

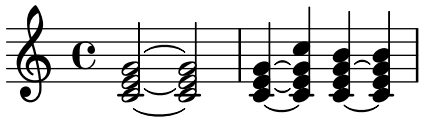
```
\relative {
  r8 c'4.~ 4 r4 |
  r8^"non" c2~ 8 r4
}
```




Per legare una successione di note la cui durata si prolunga per più misure intere, è più semplice ricorrere alla suddivisione automatica delle note, come è spiegato in [Divisione automatica delle note], pagina 81. Questo metodo divide automaticamente le note lunghe e le connette da misura a misura.

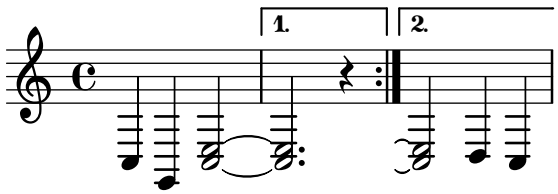
Quando si applica una legatura di valore a degli accordi, vengono legate tutte le teste delle note della stessa altezza. In assenza di altezze corrispondenti, non verrà creata alcuna legatura. Singoli suoni degli accordi possono essere legati inserendo la legatura all'interno dell'accordo stesso.

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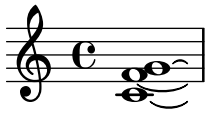


Quando la battuta della "seconda volta" di un ritornello inizia con una nota legata a quella precedente, occorre indicare la legatura nel modo seguente:

```
\relative {
  \repeat volta 2 { c g <c e>2~ }
  \alternative {
    % Prima volta: la nota seguente viene legata in modo normale
    { <c e>2. r4 }
    % Seconda volta: la nota seguente ha una legatura ripetuta
    { <c e>2\repeatTie d4 c }
  }
}
```




Le legature *L.v.* (*laissez vibrer*) indicano che le note non devono essere terminate nettamente. Si usa nella notazione per pianoforte, arpa e altri strumenti a corda e a percussione. Si inseriscono così:

```
<c' f' g'>1\laissezVibrer
```
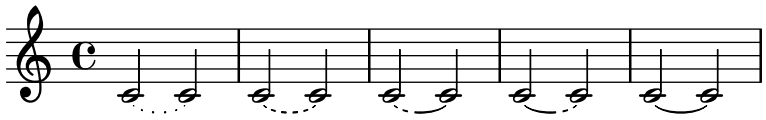


Le legature di valore possono essere impostate manualmente per avere la curva in su o in giù, come è spiegato in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Le legature di valore possono essere tratteggiate, punteggiate, oppure tracciate secondo una successione di tratti continui e tratti interrotti.
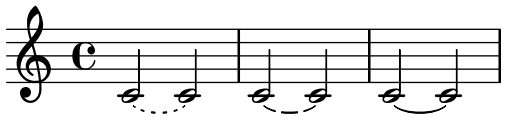
```
\relative c' {
  \tieDotted
  c2~ 2
  \tieDashed
  c2~ 2
```

```
  \tieHalfDashed
  c2~ 2
  \tieHalfSolid
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Si possono specificare modelli di tratteggiatura personalizzati:
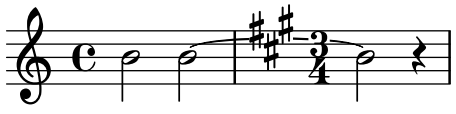
```
\relative c' {
  \tieDashPattern #0.3 #0.75
  c2~ 2
  \tieDashPattern #0.7 #1.5
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Le definizioni dei modelli di tratteggiatura delle legature di valore hanno la stessa struttura di quelle per le legature di portamento. I dettagli relativi ai modelli complessi di tratteggiatura sono trattati in [Legature di portamento], pagina 135.

Sovrascrivere le proprietà di formattazione *whiteout* e *layer* degli oggetti che devono creare uno spazio vuoto tra le legature di valore.

```
\relative {
  \override Tie.layer = #-2
  \override Staff.TimeSignature.layer = #-1
  \override Staff.KeySignature.layer = #-1
  \override Staff.TimeSignature.whiteout = ##t
  \override Staff.KeySignature.whiteout = ##t
  b'2 b~
  \time 3/4
  \key a \major
  b r4
}
```



## Comandi predefiniti

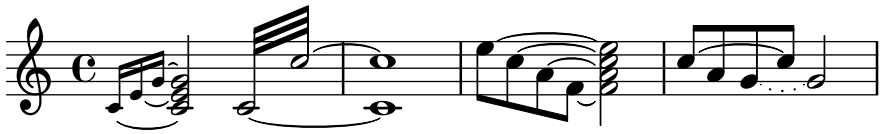
`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

## Frammenti di codice selezionati

*Usare le legature di valore con un arpeggio*

Le legature di valore vengono usate talvolta per scrivere un arpeggio. In questo caso, le due note da legare devono non essere consecutive. Per ottenere tale risultato occorre impostare la proprietà `tieWaitForNote` su `#t`. Questa funzionalità serve anche a legare un tremolo a un accordo e in generale qualsiasi coppia di note consecutive.

```
\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
}
```
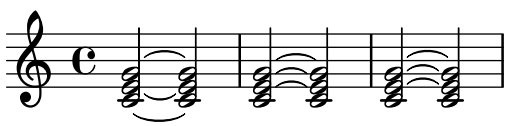


*Disegnare manualmente le legature di valore*

Le legature di valore possono essere disegnate a mano cambiando la proprietà `tie-configuration` dell'oggetto `TieColumn`. Il primo numero indica la distanza dal centro del rigo nell'unità di metà spazio rigo, mentre il secondo numero indica la direzione (1 = su, -1 = giù).

Si noti che LilyPond fa una distinzione tra valori precisi e imprecisi per il primo numero. Se si usa un valore preciso (ovvero un intero o una frazione come `(/ 4 5)`), il valore serve come posizione verticale approssimata che viene ulteriormente aggiustata da LilyPond per far sì che la legatura di valore eviti le linee del rigo. Se si usa un valore impreciso, come una virgola mobile, viene usato per la posizione verticale senza ulteriori regolazioni.

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



### Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*, Sezione “laissez vibrer” in *Glossario Musicale*.

Guida alla notazione: [Legature di portamento], pagina 135, [Divisione automatica delle note], pagina 81.

Frammenti di codice: Sezione “Expressive marks” in *Frammenti di codice*, Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “LaissezVibrerTie” in *Guida al Funzionamento Interno*, Sezione “LaissezVibrerTieColumn” in *Guida al Funzionamento Interno*, Sezione “TieColumn” in *Guida al Funzionamento Interno*, Sezione “Tie” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Cambiare rigo mentre una legatura di valore è attiva non produce una legatura obliqua.

Il cambio di chiave o di ottava durante una legatura di valore non è una situazione ben definita. In questi casi è preferibile usare una legatura di portamento.

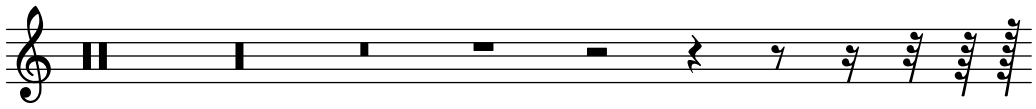
## 1.2.2 Inserimento delle pause

Le pause si inseriscono insieme alla musica contenuta nelle espressioni musicali.

### Pause

Le pause si inseriscono allo stesso modo delle note, ma con il carattere `r`. Le durate più lunghe di un intero usano i seguenti comandi predefiniti:
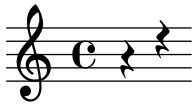
```
\new Staff {
  % Queste due linee servono solo ad abbellire questo esempio
  \time 16/1
  \omit Staff.TimeSignature
  % Mostra una pausa di maxima, equivalente a quattro brevi
  r\maxima
  % Mostra una pausa di longa, equivalente a due brevi
  r\longa
  % Mostra una pausa di breve
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Le pause d’intero, poste al centro della misura, devono essere inserite come pause multiple. Si possono usare sia per una sola misura sia su più misure, come è spiegato in [Pause d’intero], pagina 62.

Per indicare esplicitamente la posizione verticale di una pausa, si scrive la nota corrispondente seguita da `\rest`. Una pausa della durata della nota verrà collocata nella posizione della nota sul rigo. Questo permette una precisa formattazione manuale della musica polifonica, dato che il formattatore automatico che gestisce le collisioni tra pause non interviene su questo tipo di pause.

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



## Frammenti di codice selezionati

*Stili di pausa*

Esistono vari stili di pausa.

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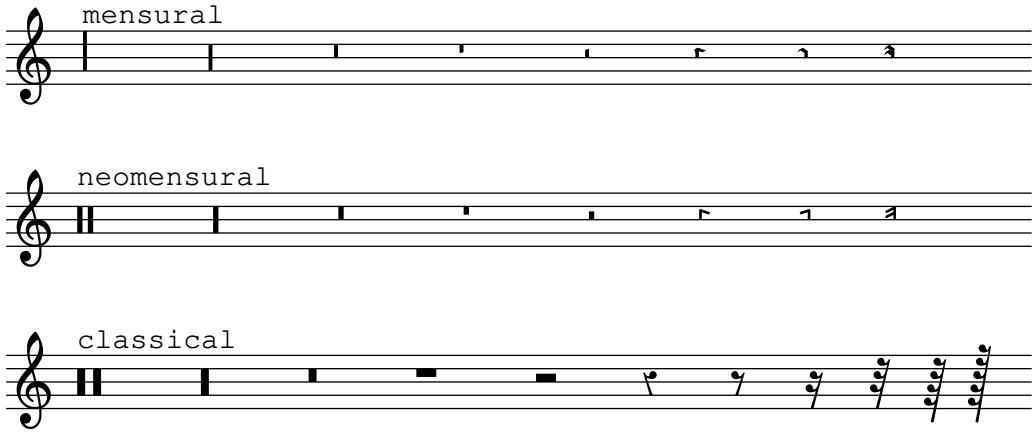
  \override Staff.Rest.style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

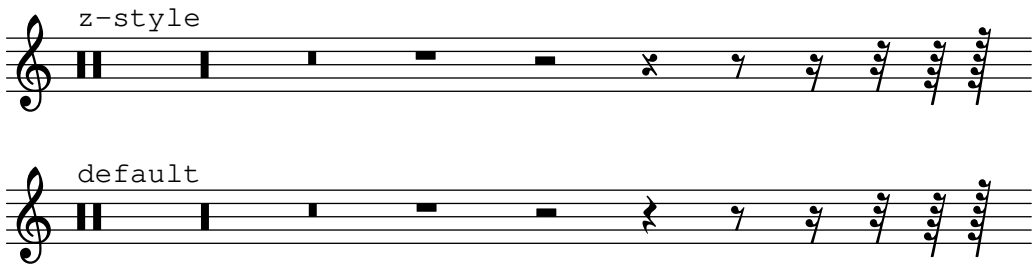
  \override Staff.Rest.style = #'neomensural
  r\maxima^\markup \typewriter { neomensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

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

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

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






## Vedi anche

Glossario musicale: Sezione “breve” in *Glossario Musicale*, Sezione “longa” in *Glossario Musicale*, Sezione “maxima” in *Glossario Musicale*.

Guida alla notazione: [Pause d’intero], pagina 62.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Rest” in *Guida al Funzionamento Interno*.
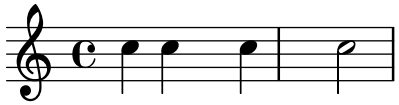
## Problemi noti e avvertimenti

Non c’è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato: si possono indicare pause da un centoventottesimo fino alla maxima (otto volte una semibreve).

## Pause invisibili
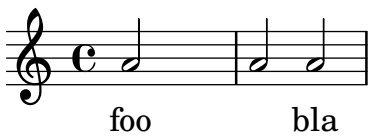
Una pausa invisibile (chiamata anche ‘pausa spaziatrice’) si inserisce come come una nota col nome `s`:

```
\relative c'' {
  c4 c s c |
  s2 c |
}
```
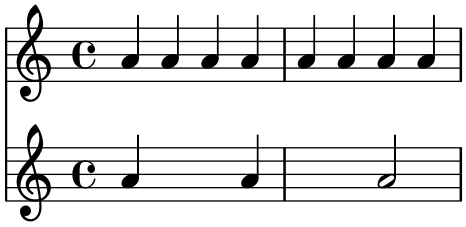


Le pause spaziatrici possono essere usate soltanto nella modalità note e nella modalità accordi. In altre situazioni, ad esempio quando si inserisce il testo vocale, si usa il comando `\skip` per saltare un valore musicale. `\skip` richiede una durata esplicita, ma questo requisito viene ignorato se il testo desume le proprie durate dalle note presenti in una melodia ad esso associata attraverso `\addlyrics` o `\lyricsto`.

```
<<
  {
    a'2 \skip2 a'2 a'2
  }
  \new Lyrics {
    \lyricmode {
      foo2 \skip 1 bla2
    }
  }
>>
```



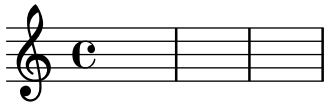
Dato che `\skip` è un comando, non modifica la durata predefinita delle note che seguono, diversamente da `s`.

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



Una pausa spaziatrice crea implicitamente i contesti `Staff` e `Voice` se non esistono già, proprio come accade per le note e le pause:

```
{ s1 s s }
```



`\skip` si limita a saltare un valore musicale, non crea nessun tipo di output.

```
% Questo input è corretto, ma non produce niente
\skip 1 \skip1 \skip 1
```

## Vedi anche

Manuale di apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: [Note nascoste], pagina 229, Sezione 5.4.7 [Visibilità degli oggetti], pagina 638.

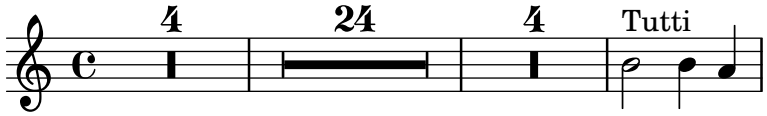
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SkipMusic” in *Guida al Funzionamento Interno*.

## Pause d’intero

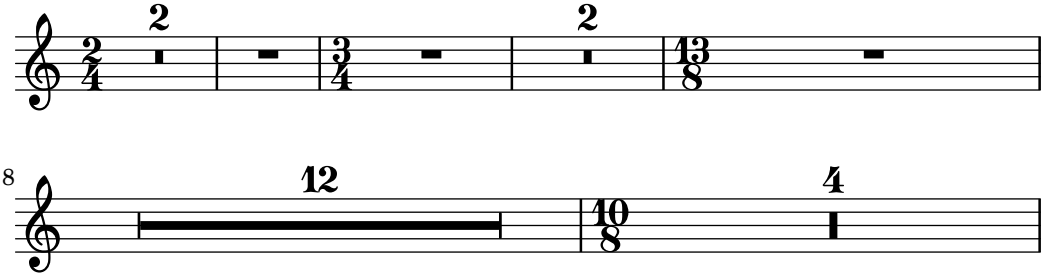
Le pause per una o più misure d’intero si inseriscono, come le note, col carattere maiuscolo `R`:

```
% L'insieme delle misure di pausa vengono riportate in una sola misura
\compressMMRests {
  R1*4
  R1*24
  R1*4
  b'2^"Tutti" b'4 a'4
}
```
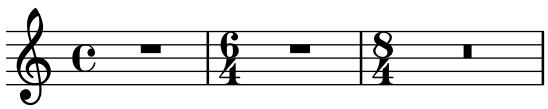



La durata delle pause multiple è identica alla notazione di durata usata per le note e deve essere sempre un numero intero di misure/lunghezze, quindi occorre spesso usare dei punti di aumentazione o delle frazioni:

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




Una pausa d'intero appare al centro della misura con la durata di una semibreve o di una breve, in base all'indicazione di tempo.

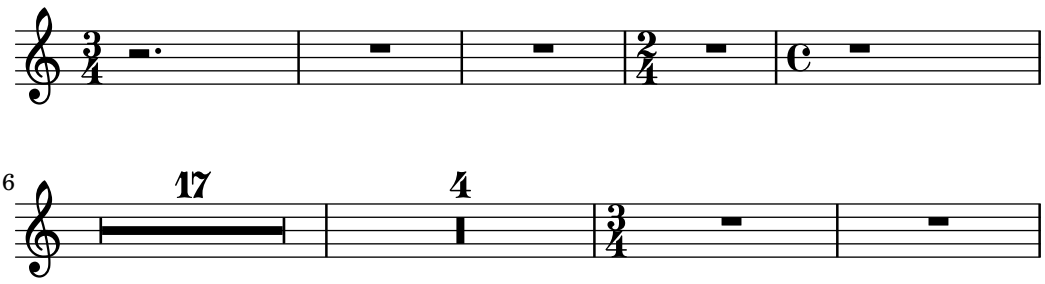
```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```



Di norma una pausa multipla viene scorporata sul pentagramma in modo da mostrare esplicitamente tutte le misure per cui si prolunga. Altrimenti, è possibile indicarla collocando in una sola misura un simbolo di pausa multipla, col numero di misure per cui la pausa si prolunga posto al di sopra della misura stessa:

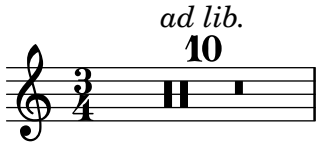
```
% Comportamento predefinito
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Tutte le misure di pausa sono riportate in una singola misura
\compressMMRests {
  r1 | R1*17 | R1*4 |
}
% Le misure della pausa multipla sono scorporate di nuovo
\time 3/4
```

```
R2.*2 |
```




Si possono aggiungere delle annotazioni alle pause multiple.

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




Quando una pausa multipla segue immediatamente un comando `\partial`, potrebbero non apparire i relativi avvertimenti del controllo battuta.

## Comandi predefiniti

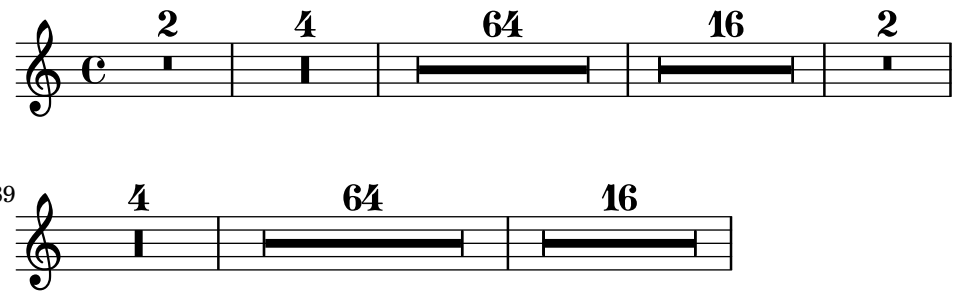
`\textLengthOn`, `\textLengthOff`, `\compressMMRests`,

## Frammenti di codice selezionati

*Regolazione della lunghezza delle pause multiple*

Le pause multiple hanno una lunghezza che dipende dalla loro durata totale e tale lunghezza è regolata da `MultiMeasureRest.space-increment`. Nota che il valore predefinito è `2.0`.

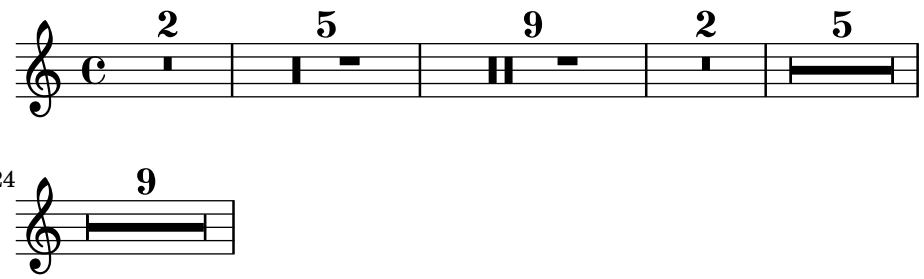
```
\relative c' {
  \compressEmptyMeasures
  R1*2 R1*4 R1*64 R1*16
  \override Staff.MultiMeasureRest.space-increment = 2.5
  R1*2 R1*4 R1*64 R1*16
}
```




*Modificare la forma delle pause multiple*

Se la pausa multipla dura dieci misure o un numero inferiore a dieci, nel rigo apparirà una serie di pause di longa e di breve (chiamate in tedesco “Kirchenpausen” - pause ecclesiastiche); altrimenti apparirà una semplice linea. Il numero predefinito di dieci può essere cambiato sovrascrivendo la proprietà `expand-limit`..

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







*Posizionamento delle pause multiple*

Diversamente dalle pause normali, non esiste un comando predefinito per cambiare la posizione sul rigo di un simbolo di pausa multipla di qualsiasi tipo connettendolo a una nota. Tuttavia, nella musica polifonica le pause multiple nelle voci dispari e pari sono separate verticalmente. Il posizionamento delle pause multiple si controlla nel modo seguente:

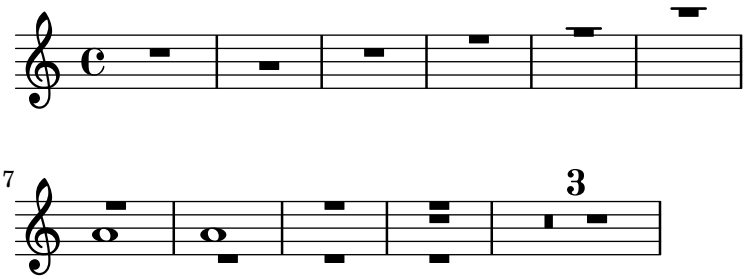
```
\relative c'' {
  % Multi-measure rests by default are set under the fourth line
  R1
  % They can be moved using an override
  \override MultiMeasureRest.staff-position = #-2
  R1
  \override MultiMeasureRest.staff-position = #0
  R1
  \override MultiMeasureRest.staff-position = #2
  R1
  \override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

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

  % Separating multi-measure rests in more than two voices
  % requires an override
```

```
  << { R1 } \\ { R1 } \\
     \once \override MultiMeasureRest.staff-position = #0
     { R1 }
  >>

  % Using compressed bars in multiple voices requires another override
  % in all voices to avoid multiple instances being printed
  \compressMMRests
  <<
   \revert MultiMeasureRest.direction
    { R1*3 }
    \\
   \revert MultiMeasureRest.direction
    { R1*3 }
  >>
}
```

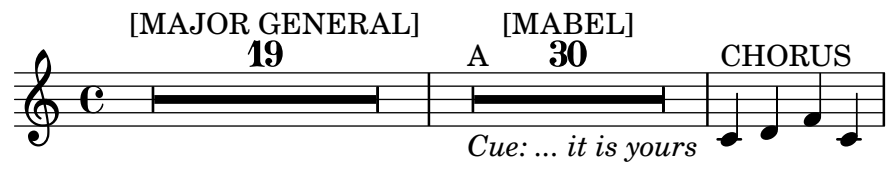



*Testo a margine delle pause multiple*

Il testo a margine di una pausa multipla viene centrato sopra o sotto di essa. Se il testo è lungo, la misura non si espanderà. Per espandere la pausa multipla in modo che si allinei col testo, conviene usare un accordo vuoto con del testo attaccato prima della pausa multipla.

Il testo così attaccato a una nota spaziatrice viene allineato a sinistra della posizione in cui la nota sarebbe posta nella misura, ma se la lunghezza della misura è determinata dalla lunghezza del testo, il testo verrà centrato.

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



### Vedi anche

Glossario musicale: Sezione “pausa multipla” in *Glossario Musicale*.

Guida alla notazione: [Durata], pagina 46, Sezione 1.8 [Testo], pagina 238, Sezione 1.8.2 [Formattazione del testo], pagina 246, [Scritte], pagina 239.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MultiMeasureRest” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestNumber” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestScript” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestText” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Se una diteggiatura viene posta su una pausa multipla (ad esempio `R1*10-4`), il numero della diteggiatura può collidere col numero del contatore delle battute.

Non è possibile condensare automaticamente molteplici pause normali in in una singola pausa multipla.
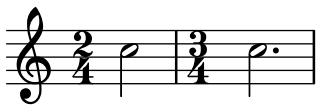
Le pause multiple non considerano le collisioni di pausa.

## 1.2.3 Aspetto dei ritmi

### Indicazione di tempo

L’indicazione di tempo si imposta così:

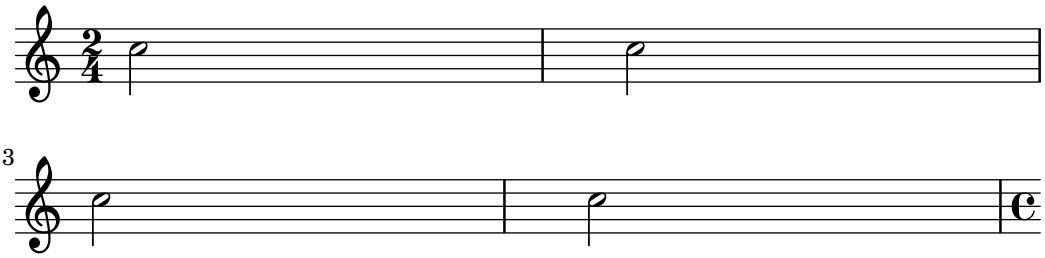
```
\time 2/4 c''2
\time 3/4 c''2.
```
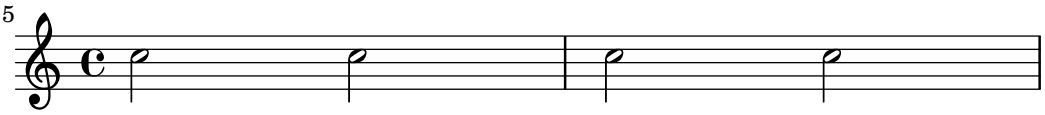


I cambi di indicazione di tempo a metà misura sono trattati in [Anacrusi], pagina 75.

Le indicazioni di tempo appaiono all’inizio di un brano e ogni volta che l’indicazione cambia. Se il cambio ha luogo alla fine di un rigo, appare un’indicazione di tempo di precauzione. Si può modificare questo comportamento predefinito, come è spiegato in Sezione 5.4.7 [Visibilità degli oggetti], pagina 638.
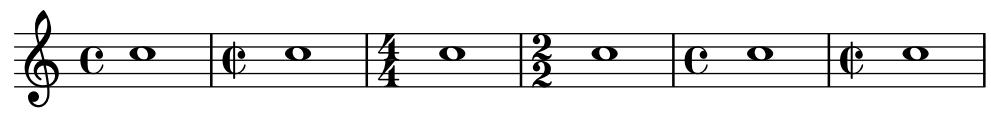
```
\relative c'' {
  \time 2/4
  c2 c
  \break
  c c
  \break
  \time 4/4
  c c c c
}
```






Il simbolo di indicazione di tempo usato nei tempi 2/2 e 4/4 può essere sostituito da un numero:

```
\relative c'' {
  % Stile predefinito
  \time 4/4 c1
  \time 2/2 c1
  % Passaggio allo stile numerico
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  % Ritorno allo stile predefinito
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}
```
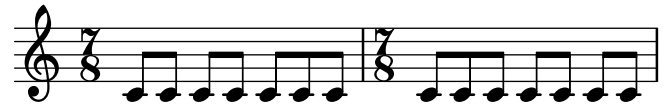


Le indicazioni di tempo mensurali sono trattate in [Indicazioni di tempo mensurali], pagina 446.

Oltre a impostare l'indicazione di tempo che appare nel pentagramma, il comando `\time` imposta anche i valori delle proprietà basate sull'indicazione di tempo, ovvero `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste proprietà si trovano in `scm/time-signature-settings.scm`.

Si può sovrascrivere il valore predefinito di `beatStructure` nel comando `\time` stesso specificandolo come primo argomento opzionale:
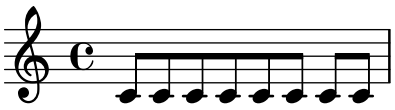
```
\score {
  \new Staff {
    \relative {
      \time 2,2,3 7/8
      \repeat unfold 7 { c'8 } |
      \time 3,2,2 7/8
      \repeat unfold 7 { c8 } |
    }
  }
}
```



Oppure si possono impostare tutti i valori predefiniti di queste variabili relative all'indicazione di tempo, incluse `baseMoment` e `beamExceptions`. I valori possono essere impostati in modo indipendente per diverse indicazioni di tempo. I nuovi valori hanno effetto appena viene eseguito un nuovo comando `\time` che abbia lo stesso valore dell'indicazione di tempo specificata nelle nuove impostazioni:

```
\score {
```

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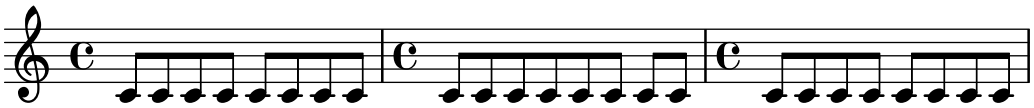


`\overrideTimeSignatureSettings` prende quattro argomenti:

1. *`timeSignatureFraction`*, una frazione che indica l'indicazione di tempo a cui questi valori si riferiscono.
2. *`baseMomentFraction`*, una frazione che contiene il numeratore e il denominatore dell'unità di tempo.
3. *`beatStructure`*, una lista Scheme che indica la struttura dei battiti nella misura, nell'unità di *`baseMomentFraction`*.
4. *`beamExceptions`*, una lista di associazione (*alist*) che contiene regole di disposizione delle travature che vanno oltre la fine ad ogni battito, come descritto in [Impostare il comportamento delle travature automatiche], pagina 88.

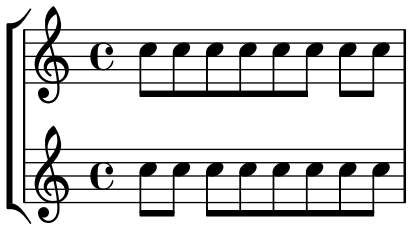
I valori modificati delle proprietà predefinite dell'indicazione di tempo possono essere ripristinati ai valori originali:

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



Si possono stabilire valori diversi delle proprietà predefinite dell'indicazione di tempo per righi diversi spostando `Timing_translator` e `Default_bar_line_engraver` dal contesto `Score` al contesto `Staff`.

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



Un ulteriore metodo per modificare queste variabili relative all'indicazione di tempo, che evita di mostrare di nuovo l'indicazione di tempo al momento del cambio, è descritto in [Impostare il comportamento delle travature automatiche], pagina 88.

## Comandi predefiniti
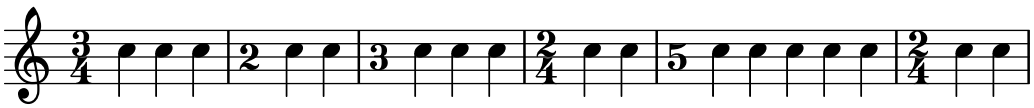
`\numericTimeSignature`, `\defaultTimeSignature`.

## Frammenti di codice selezionati

*Indicazione di tempo che mostra solo il numeratore (invece della frazione)*

Talvolta un'indicazione di tempo non deve mostrare la frazione intera (ad esempio 7/4), ma solo il numeratore (7 in questo caso). Si può ottenere facilmente con `\override Staff.TimeSignature.style = #'single-digit`, che cambia lo stile in modo permanente. Con `\revert Staff.TimeSignature.style`, questa impostazione può essere annullata. Per applicare lo stile a cifra singola (`single-digit`) a una sola indicazione di tempo, si usa il comando `\override` preceduto da `\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
}
```



## Vedi anche

Glossario musicale: Sezione "indicazione di tempo" in *Glossario Musicale*

Guida alla notazione: [Indicazioni di tempo mensurali], pagina 446, [Impostare il comportamento delle travature automatiche], pagina 88, [Gestione del tempo], pagina 122.

File installati: `scm/time-signature-settings.scm`.
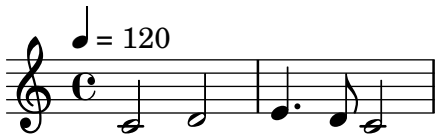
Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TimeSignature" in *Guida al Funzionamento Interno*, Sezione "Timing_translator" in *Guida al Funzionamento Interno*.

## Indicazioni metronomiche
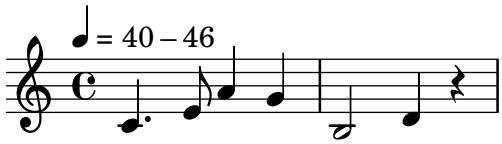
Un'indicazione metronomica è semplice da scrivere:

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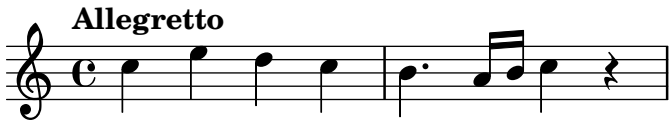



Le indicazioni metronomiche si possono rappresentare anche come una gamma di due numeri:

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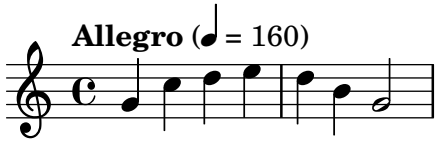



Al loro posto si possono usare delle indicazioni di tempo testuali:

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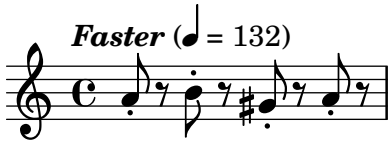



Un'indicazione metronomica, se combinata con del testo, viene posta automaticamente tra parentesi:

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




In generale, il testo può essere qualsiasi oggetto di tipo testuale:
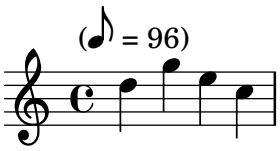
```
\relative {
  \tempo \markup { \italic Faster } 4 = 132
  a'8-. r8 b-. r gis-. r a-. r
}
```




È possibile scrivere un'indicazione metronomica tra parentesi e senza testo includendo una stringa vuota nell'input:

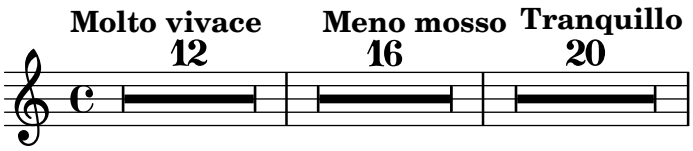
```
\relative {
```

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



In una parte per uno strumento che ha lunghi periodi pieni di pause, le indicazioni di tempo sono talvolta molto ravvicinate. Il comando `\markLengthOn` aggiunge dello spazio orizzontale per impedire che le indicazioni di tempo si sovrappongano; `\markLengthOff` ripristina il comportamento predefinito, per cui le indicazioni di tempo non sono tenute in considerazione ai fini della spaziatura orizzontale.

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




## Frammenti di codice selezionati

*Posizionare il metronomo e i numeri di chiamata sotto il rigo*

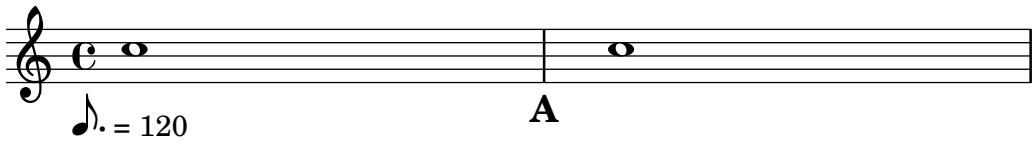
Di norma, il metronomo e i numeri di chiamata vengono posizionati sopra il rigo. Per metterli sotto il rigo basta impostare correttamente la proprietà `direction` di `MetronomeMark` o `RehearsalMark`.

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

{
  % Metronome marks below the staff
  \override Score.MetronomeMark.direction = #DOWN
  \tempo 8. = 120
  c''1

  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
```
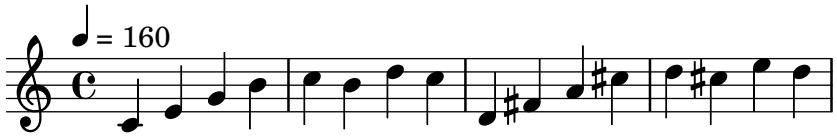
```
}
```




*Modificare il tempo senza mostrare l'indicazione metronomica*

Per cambiare il tempo del file MIDI senza che appaia l'indicazione metronomica, basta renderla invisibile.

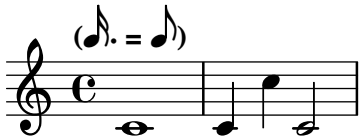
```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```




*Creare indicazioni metronomiche in modalità testuale*

Si possono creare nuove indicazioni metronomiche in modalità testuale, ma non modificheranno il tempo del file 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
}
```



I dettagli si trovano in Sezione 1.8.2 [Formattazione del testo], pagina 246.

## Vedi anche

Glossario musicale: Sezione “metronomo” in *Glossario Musicale*, Sezione “indicazione di tempo” in *Glossario Musicale*, Sezione “indicazione metronomica” in *Glossario Musicale*.

Guida alla notazione: Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione 3.5 [Creazione dell’output MIDI], pagina 525.

Frammenti di codice: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MetronomeMark” in *Guida al Funzionamento Interno*.
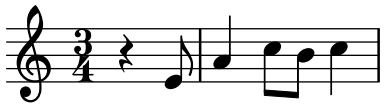
## Anacrusi

Le misure parziali, come l’*anacrusi* o la battuta in levare, si inseriscono col comando `\partial`:

```
\partial durata
```

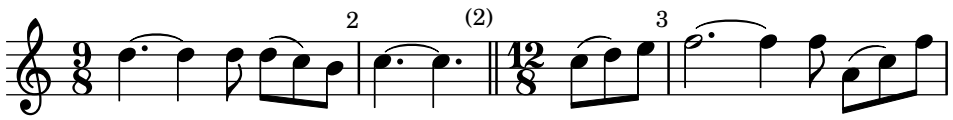
Quando si usa `\partial` all’inizio di una partitura, la *`durata`* è la lunghezza della musica che precede la prima battuta.

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



Quando si usa `\partial` dopo l’inizio di una partitura, la *durata* è la lunghezza *rimanente* della misura corrente. Non crea una battuta con un nuovo numero.
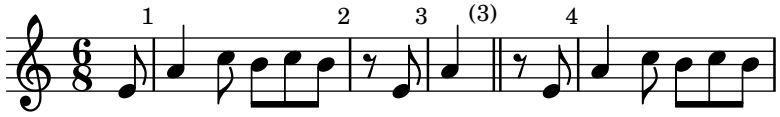
```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
        #end-of-line-invisible
  \time 9/8
  d''4.~ 4 d8 d( c) b | c4.~ 4. \bar "||"
  \time 12/8
  \partial 4.
  c8( d) e | f2.~ 4 f8 a,( c) f |
}
```




Il comando `\partial` è *obbligatorio* quando l’indicazione di tempo cambia in mezzo a una misura, ma si può usare anche da solo.

```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
        #end-of-line-invisible
  \time 6/8
  \partial 8
  e'8 | a4 c8 b[ c b] |
```

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




Il comando `\partial` imposta la proprietà `Timing.measurePosition`, che è un numero razionale che indica quanto tempo della misura è trascorsa.

## Vedi anche

Glossario musicale: Sezione “anacrusi” in *Glossario Musicale*.

Guida alla notazione: [Abbellimenti], pagina 116.
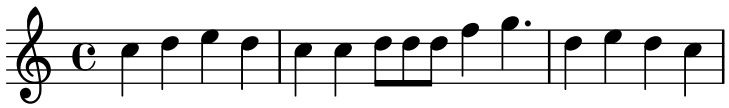
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Timing_translator” in *Guida al Funzionamento Interno*.

## Musica in tempo libero
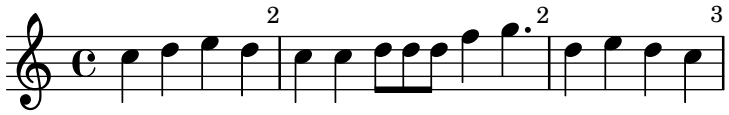
Nella musica in un tempo determinato l’inserimento delle stanghette e dei numeri di battuta è calcolato automaticamente. Nella musica in tempo libero (per esempio, la cadenza), un simile comportamento non è desiderabile, e può essere ‘disabilitato’ col comando `\cadenzaOn` e poi ‘riabilitato’ quando necessario con `\cadenzaOff`.

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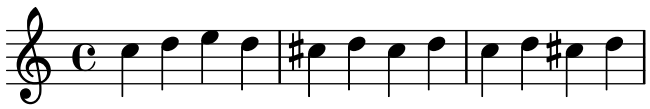


La numerazione delle battute riprende dopo la cadenza.

```
\relative c'' {
  % Mostra tutti i numeri di battuta
  \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
}
```
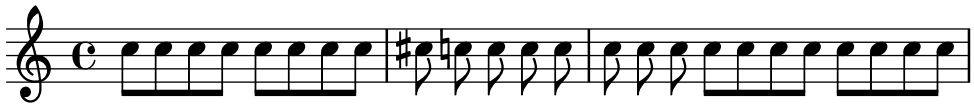


Se si inserisce un comando `\bar` dentro una cadenza non viene iniziata una nuova misura, anche se appare una stanghetta nell'output. Quindi qualsiasi alterazione, che di solito si considera sempre attiva fino alla fine della misura, sarà ancora valida dopo la stanghetta stampata da `\bar`. Se si desidera che le alterazioni successive appaiano, si dovranno inserire manualmente delle alterazioni forzate o di precauzione, come è spiegato in [Alterazioni], pagina 6.

```
\relative c'' {
  c4 d e d
  \cadenzaOn
  cis4 d cis d
  \bar "|"
  % Il primo cis viene stampato senza alterazione anche se si trova dopo \bar
  cis4 d cis! d
  \cadenzaOff
  \bar "|"
}
```



La disposizione automatica delle travature viene disabilitata da `\cadenzaOn`. Quindi tutte le travature nelle cadenze devono essere inserite manualmente. Si veda [Travature manuali], pagina 97.

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



Questi comandi predefiniti hanno effetto su tutti i righi di una partitura, anche quando inseriti in un solo contesto `Voice`. Per modificare questo comportamento, si sposta `Timing_translator` dal contesto `Score` al contesto `Staff`. Si veda [Notazione polimetrica], pagina 78.

## Comandi predefiniti

`\cadenzaOn`, `\cadenzaOff`.

## Vedi anche

Glossario musicale: Sezione "cadenza" in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.7 [Visibilità degli oggetti], pagina 638, [Notazione polimetrica], pagina 78, [Travature manuali], pagina 97, [Alterazioni], pagina 6.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

## Problemi noti e avvertimenti

Le interruzioni automatiche di linea e di pagina possono aver luogo solo dopo una stanghetta di battuta; quindi, per consentire delle interruzioni nei lunghi passaggi di musica in tempo libero è necessario inserire manualmente delle stanghette 'invisibili':

```
\bar ""
```

## Notazione polimetrica

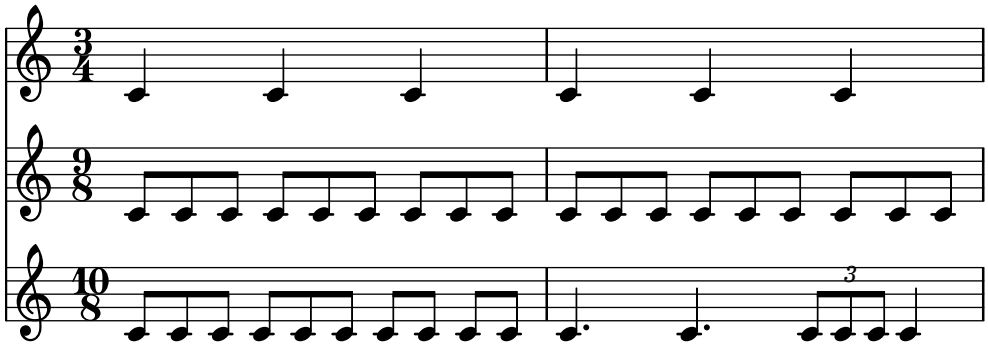
La notazione polimetrica è supportata esplicitamente o tramite la modifica manuale del simbolo d'indicazione di tempo (e la trasformazione della durata delle note).

### *Diverse indicazioni di tempo con misure di uguale lunghezza*

Si sceglie una normale indicazione di tempo per ogni rigo e si imposta `timeSignatureFraction` sulla frazione desiderata. Quindi si usa la funzione `\scaleDurations` per scalare la durata delle note di ogni rigo in modo che rientrino nella comune indicazione di tempo.

L'esempio seguente presenta simultaneamente musica con indicazioni di tempo di 3/4, 9/8 e 10/8. Nel secondo rigo le durate appaiono come moltiplicate per 2/3 (perché 2/3 * 9/8 = 3/4), mentre nel terzo rigo le durate appaiono come moltiplicate per 3/5 (perché 3/5 * 10/8 = 3/4). È possibile che si debbano inserire a mano le travature, perché la scalatura delle durate influenzerà le regole della disposizione automatica delle travature.

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



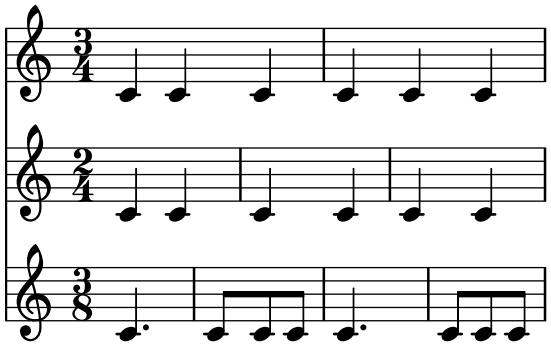

### *Diverse indicazioni di tempo con misure di lunghezza differenti*

Si può dare a ogni rigo la sua indicazione di tempo indipendente spostando `Timing_translator` e `Default_bar_line_engraver` nel contesto `Staff`.

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

% Ora ogni rigo ha la sua indicazione di tempo.

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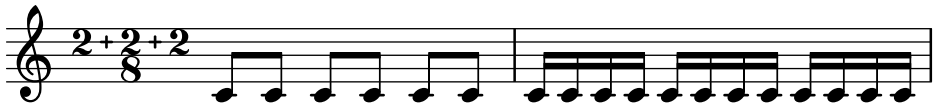


### *Indicazioni di tempo composto*

Si creano con la funzione `\compoundMeter`. La sintassi è:

```
\compoundMeter #'(lista di liste)
```

La struttura più semplice è una singola lista, dove l'*ultimo* numero indica il numero inferiore dell'indicazione di tempo e i numeri precedenti indicano i numeri superiori del segno di tempo.
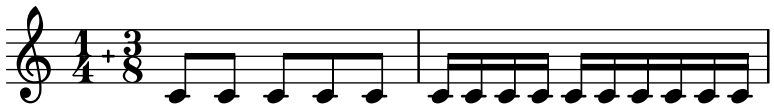
```
\relative {
  \compoundMeter #'((2 2 2 8))
  \repeat unfold 6 c'8 \repeat unfold 12 c16
}
```
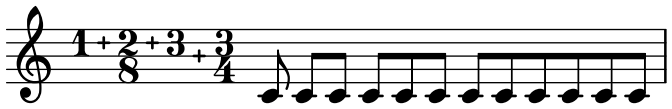


Si possono costruire tempi più complessi tramite ulteriori liste. Le modalità di disposizione automatica delle travature varieranno a seconda di questi valori.

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

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





## Vedi anche

Glossario musicale: Sezione "polimetrico" in *Glossario Musicale*, Sezione "indicazione di tempo polimetrico" in *Glossario Musicale*, Sezione "tempo" in *Glossario Musicale*.

Guida alla notazione: [Travature automatiche], pagina 85, [Travature manuali], pagina 97, [Indicazione di tempo], pagina 67, [Scalare le durate], pagina 53.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TimeSignature” in *Guida al Funzionamento Interno*, Sezione “Timing_translator” in *Guida al Funzionamento Interno*, Sezione “Default_bar_line_engraver” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Sebbene le note che si presentano nello stesso momento in ciascuno dei vari righi siano poste nello stesso punto orizzontale, le stanghette (in ciascun rigo) potrebbero causare una spaziatura difforme in ciascuna delle diverse indicazioni di tempo.
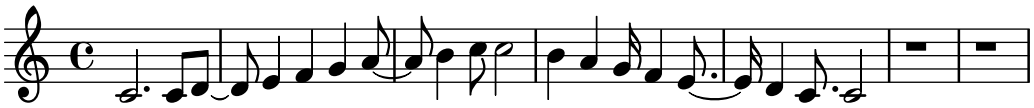
L'uso di un blocco `midi` con notazione polimetrica potrebbe causare avvertimenti indesiderati di controllo di battuta. In questo caso, spostare `Timing_translator` dal contesto `Score` al contesto `Staff` all'interno del blocco `midi`.

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

## Divisione automatica delle note

Le note le cui durate eccedono il valore della battuta possono essere convertite automaticamente in note con legature di valore a cavallo delle stanghette sostituendo l'incisore `Note_heads_engraver` con `Completion_heads_engraver`. Analogalmente, le pause le cui durate eccedono il valore della battuta possono essere divise automaticamente sostituendo `Rest_engraver` con `Completion_rest_engraver`. Nell'esempio seguente, le note e le pause che eccedono la durata di battuta vengono divise e le note sono anche connesse con legature di valore a cavallo della stanghetta.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
  \remove "Rest_engraver"
  \consists "Completion_rest_engraver"
}
\relative {
  c'2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2
}
```
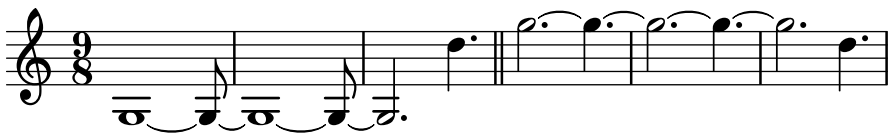


Questi incisori dividono tutte le note e le pause in corrispondenza della stanghetta e inseriscono le legature di valore. Uno dei suoi usi possibili è la verifica di partiture complesse: se le misure non sono riempite interamente, le legature di valore mostrano esattamente di quanto è ecceduta ogni misura.

La proprietà `completionUnit` imposta la durata preferita per le note divise.
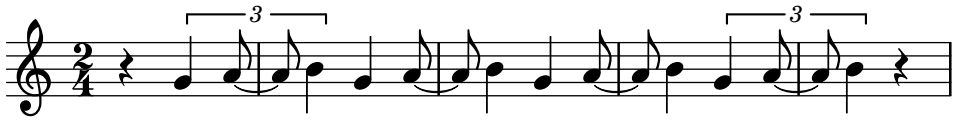
```
\new Voice \with {
```

```
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
} \relative {
  \time 9/8 g\breve. d''4. \bar "||"
  \set completionUnit = #(ly:make-moment 3 8)
  g\breve. d4.
}
```



Questi incisori dividono le note che hanno una durata ridimensionata, come quelle dei gruppi irregolari, in note con lo stesso fattore di ridimensionamento della nota di input.

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




## Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*

Manuale di apprendimento: Sezione “Gli incisori” in *Manuale di Apprendimento*, Sezione “Aggiungere e togliere gli incisori” in *Manuale di Apprendimento*.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Rest_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion_rest_engraver” in *Guida al Funzionamento Interno*, Sezione “Forbid_line_break_engraver” in *Guida al Funzionamento Interno*.

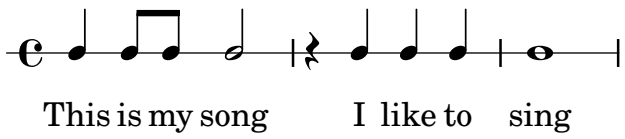
## Problemi noti e avvertimenti

In accordo col comportamento precedente, note e pause la cui durata è più lunga di una misura, come `c1*2`, sono divise in note prive di fattore di ridimensionamento, `{ c1 c1 }`. La proprietà `completionFactor` controlla questo comportamento e impostandola su `#f` fa sì che le note e le pause divise abbiano il fattore di ridimensionamento delle durate di input.

### Mostrare i ritmi della melodia

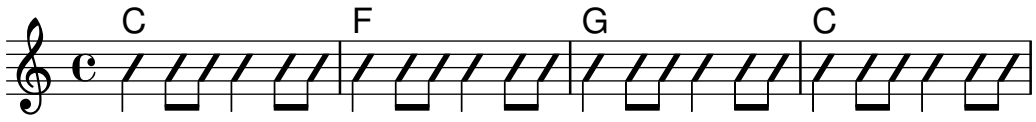
È possibile mostrare soltanto il ritmo di una melodia usando il rigo ritmico. Tutte le altezze delle note su tale rigo sono appiattite e il rigo stesso ha una sola linea

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




I diagrammi degli accordi per chitarra di solito mostrano i ritmi di accompagnamento. Si possono visualizzare usando l'incisore `Pitch_squash_engraver` e il comando `\improvisationOn`.

```
<<
  \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
  }
>>
```




La musica contenente accordi può essere usata anche come input per `RhythmicStaff` e per essere usata con l'incisore `Pitch_squash_engraver` se gli accordi vengono prima ridotti a note singole con la funzione musicale `\reduceChords`:

```
\new RhythmicStaff {
```

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



## Comandi predefiniti

`\improvisationOn`, `\improvisationOff`, `\reduceChords`.

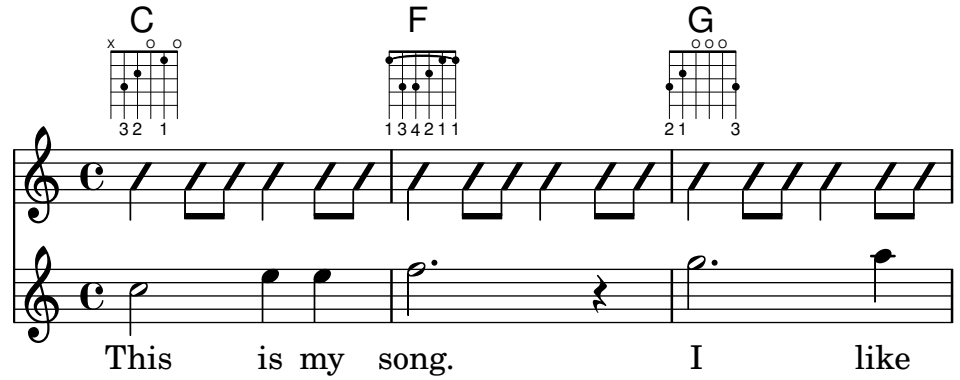
## Frammenti di codice selezionati

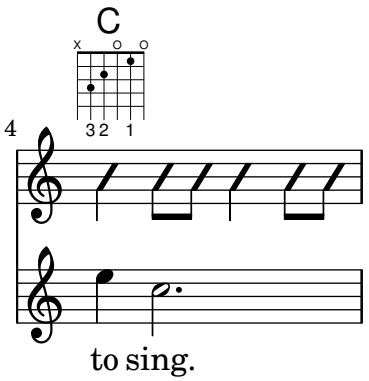
*Ritmi di accompagnamento per chitarra*

Per la musica per chitarra, è possibile mostrare i ritmi di accompagnamento, insieme alle note della melodia e ai nomi e ai diagrammi degli accordi.

```
\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.
    }
  }
>>
```




## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

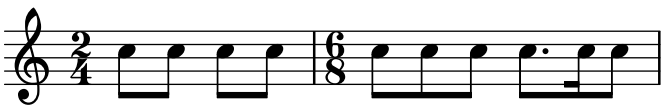
Guida al funzionamento interno: Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*.

### 1.2.4 Travature

## Travature automatiche

Le travature sono inserite automaticamente:
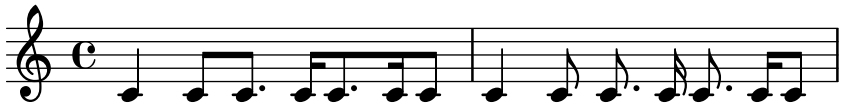
```
\relative c'' {
  \time 2/4 c8 c c c
  \time 6/8 c8 c c c8. c16 c8
}
```



Se queste impostazioni automatiche non sono soddisfacenti, si può definire esplicitamente la disposizione delle travature, come è spiegato in [Travature manuali], pagina 97. Le travature *devono* essere inserite manualmente se devono estendersi oltre le pause.

La disposizione automatica delle travature, se non necessaria, può essere disabilitata con `\autoBeamOff` e riabilitata con `\autoBeamOn`:

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



**Nota:** Se si usano le travature per indicare i melismi nelle parti vocali, occorre disabilitare la disposizione automatica delle travature con `\\autoBeamOff` e le travature devono essere indicate manualmente. L'uso di `\partCombine` insieme a `\autoBeamOff` può produrre risultati imprevisti. Si vedano i frammenti di codice per avere maggiori informazioni.

Si possono creare dei modelli di disposizione delle travature diversi da quelli automatici predefiniti, come è spiegato in [Impostare il comportamento delle travature automatiche], pagina 88.

## Comandi predefiniti
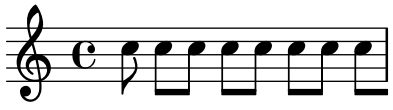
`\autoBeamOff`, `\autoBeamOn`.
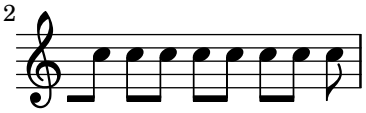
## Frammenti di codice selezionati

*Travature che attraversano le interruzioni di linea*

Le interruzioni di linea sono di norma proibite quando le travature attraversano la stanghetta di una battuta. Si può cambiare questo comportamento nel modo seguente:

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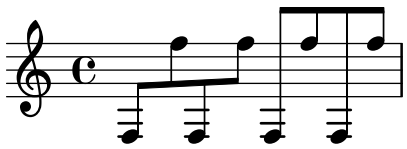





*Modificare la distanza delle travature angolari*

Le travature angolari vengono inserite automaticamente quando viene rilevata un'ampia distanza tra le teste di nota. Questo comportamento può essere regolato attraverso la proprietà `auto-knee-gap`. Viene disegnata una travatura angolare se la distanza è più grande del valore di `auto-knee-gap` più la larghezza della travatura (che dipende dalla durata delle note e dall'inclinazione della travatura). Il valore predefinito di `auto-knee-gap` è 5.5 spazi rigo.

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



*PartCombine e autoBeamOff*

La funzione `\autoBeamOff`, se usata insieme a `\partCombine`, può essere difficile da comprendere.

È preferibile usare invece

`\set Staff.autoBeaming = ##f`
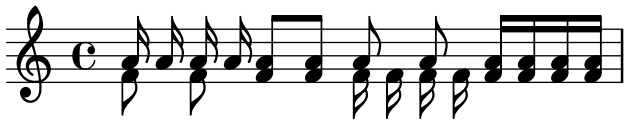
per assicurarsi che la disposizione delle travature sia disabilitata per tutto il rigo.

`\partCombine` funziona con 3 voci – gambo in su singolo, gambo in giù singolo, gambo in su unito.

L'uso di `\autoBeamOff` all'interno del primo argomento di partCombine ha effetto sulla voce che è attiva al momento in cui la funzione viene elaborata, ovvero sul gambo in su singolo o sul gambo in giù unito. L'uso di `\autoBeamOff` nel secondo argomento avrà effetto sulla voce che ha il gambo in giù singolo.

Per poter usare `\autoBeamOff` per impedire tutte le disposizioni automatiche delle travature, se usato con `\partCombine`, è necessario richiamare tre volte la funzione `\autoBeamOff`.

```
{
  %\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 |
  }
}
```



### Vedi anche

Guida alla notazione: [Travature manuali], pagina 97, [Impostare il comportamento delle travature automatiche], pagina 88.

File installati: `scm/auto-beam.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Auto_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamEvent” in *Guida al Funzionamento Interno*, Sezione “BeamForbidEvent” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*, Sezione “unbreakable-spanner-interface” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Le proprietà di una travatura sono determinate all'*inizio* della sua costruzione e qualsiasi ulteriore modifica alle sue proprietà che venga fatta prima che la travatura sia stata completata non avrà effetto finché non inizia la *successiva*, nuova travatura.

## Impostare il comportamento delle travature automatiche

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata da tre proprietà di contesto: `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste variabili possono essere sovrascritti, come vedremo tra breve, oppure si possono anche cambiare i valori predefiniti stessi, come è spiegato in [Indicazione di tempo], pagina 67.

Se è definita una regola `beamExceptions` per l'indicazione di tempo corrente, tale regola soltanto determina la disposizione delle travature; i valori di `baseMoment` e `beatStructure` vengono ignorati. Se non è definita alcuna regola `beamExceptions` per l'indicazione di tempo corrente, la disposizione delle travature è determinata dai valori di `baseMoment` e `beatStructure`.

### *Disposizione delle travature basata su* `baseMoment` *e* `beatStructure`

Dato che le indicazioni di tempo più comuni hanno delle regole `beamExceptions` già definite, occorre disabilitarle se la disposizione automatica deve basarsi su `baseMoment` e `beatStructure`. Le regole `beamExceptions` si disabilitano con questo comando
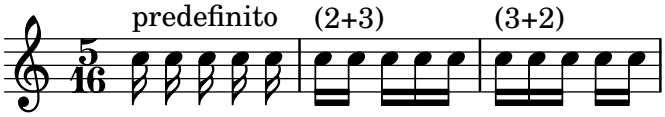
```
\set Timing.beamExceptions = #'()
```

Quando `beamExceptions` è impostato su `#'()`, o per impostazione esplicita o perché non sono state definite internamente le `beamExceptions` per l'indicazione di tempo corrente, le estremità delle travature si trovano sulle suddivisioni come specificato dalle proprietà di contesto `baseMoment` e `beatStructure`. `beatStructure` è una lista *Scheme* che definisce la lunghezza di ogni suddivisione in rapporto alla misura in unità di `baseMoment`. Per impostazione predefinita, `baseMoment` è uno fratto il denominatore dell'indicazione di tempo e ogni unità di `baseMoment` corrisponde a una singola suddivisione.

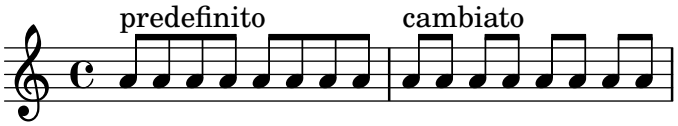
Per ogni indicazione di tempo esistono valori separati per `beatStructure` e `baseMoment`. Le modifiche di queste variabili hanno effetto solo sulle indicazioni di tempo attive, dunque tali modifiche devono essere poste dopo il comando `\time` che inizia una nuova indicazione di tempo, non prima. I nuovi valori assegnati a una certa indicazione di tempo sono mantenuti e reintrodotti ogni volta che quell'indicazione di tempo viene ristabilita.

```
\relative c''{
  \time 5/16
  c16^"predefinito" c c c c |
  % È improbabile che per un tempo di 5/16 sia stata definita beamExceptions,
```

```
  % ma disabilitiamola lo stesso per sicurezza
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 2,3
  c16^"(2+3)" c c c c |
  \set Timing.beatStructure = 3,2
  c16^"(3+2)" c c c c |
}
```



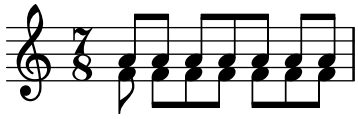

```
\relative {
  \time 4/4
  a'8^"predefinito" a a a a a a a
  % Disabilita beamExceptions perché è senz'altro definita
  % per il tempo 4/4
  \set Timing.beamExceptions = #'()
  \set Timing.baseMoment = #(ly:make-moment 1/4)
  \set Timing.beatStructure = 1,1,1,1
  a8^"cambiato" a a a a a a a
}
```




Le modifiche alle impostazioni delle travature possono essere limitate a contesti specifici. Se non si specifica alcuna impostazione in un contesto di livello più basso, verrà applicata l'impostazione del contesto che lo contiene.

```
\new Staff {
  \time 7/8
  % Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 7/
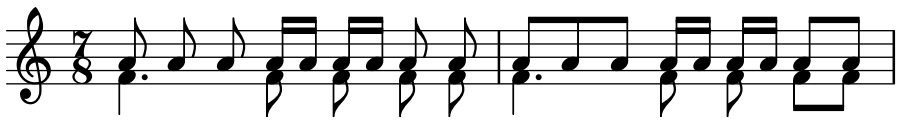
  \set Staff.beatStructure = 2,3,2
  <<
    \new Voice = one {
      \relative {
        a'8 a a a a a a
      }
    }
    \new Voice = two {
      \relative {
        \voiceTwo
        \set Voice.beatStructure = 1,3,3
        f'8 f f f f f f
      }
    }
  >>
}
```



Quando si usano più voci, occorre specificare il contesto `Staff` se si vuole applicare la disposizione delle travature a tutte le voci del rigo:
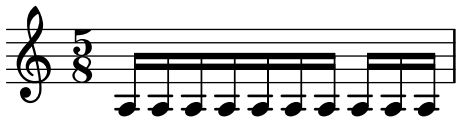
```
\time 7/8
% ritmo 3-1-1-2
% Se non si specifica il contesto, la modifica viene applicata a Voice e quindi non fu
% Dato che le voci sono autogenerate, tutto il ritmo avrà come baseMoment (1 . 8)
\set beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>

% Funziona correttamente se si specifica il contesto Staff
\set Staff.beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>
```



Il valore di `baseMoment` può essere regolato in modo da cambiare il comportamento delle travature, se si vuole. In questo caso occorre cambiare anche il valore di `beatStructure` così che sia compatibile col nuovo valore di `baseMoment`.

```
\time 5/8
% Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 5/8
\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = 7,3
\repeat unfold 10 { a16 }
```



`baseMoment` è un *momento*, ovvero un'unità della durata musicale. Una quantità di tipo *moment* viene creata dalla funzione Scheme `ly:make-moment`. Per maggiori informazioni su questa funzione, si veda [Gestione del tempo], pagina 122.

Per impostazione predefinita, `baseMoment` ha un valore di uno fratto il denominatore dell'indicazione di tempo. Le eccezioni a questa regola si trovano in `scm/time-signature-settings.scm`.
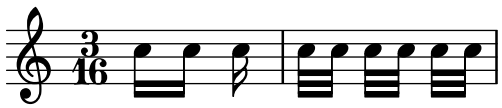
### *Disposizione delle travature con* `beamExceptions`

Le regole speciali di disposizione automatica delle travature (diverse da quelle che determinano la corrispondenza della travatura alla suddivisione) sono definite nella proprietà `beamExceptions`.

Il valore di `beamExceptions`, una struttura dati Scheme piuttosto complessa, è più facile da generare con la funzione `\beamExceptions`. A tale funzione viene passato uno o più schemi ritmici della misura, specificati con travature manuali. Le misure devono essere separate da un controllo di battuta `|` dato che la funzione non ha altro modo per determinare la lunghezza della misura. Ecco un semplice esempio:

```
\relative c'' {
  \time 3/16
  \set Timing.beatStructure = 2,1
```

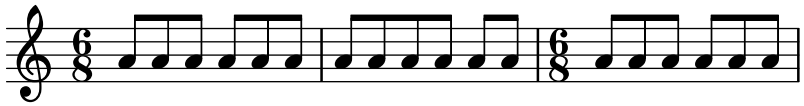
```
  \set Timing.beamExceptions =
    \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }
  c16 c c |
  \repeat unfold 6 { c32 } |
}
```



**Nota:** Il valore di `beamExceptions` deve essere una lista *completa* di eccezioni, ovvero bisogna includere tutte le eccezioni che si vogliono applicare. Non è possibile aggiungere, rimuovere o modificare soltanto una eccezione. Anche se questo può sembrare scomodo, significa anche che non c'è bisogno di conoscere le attuali impostazioni delle travature per poter specificare un nuovo modello di disposizione delle travature.

Quando cambia l'indicazione di tempo, vengono impostati i valori predefiniti di `Timing.baseMoment`, `Timing.beatStructure` e `Timing.beamExceptions`. L'impostazione dell'indicazione di tempo ripristina le impostazioni automatiche delle travature del contesto `Timing` ai valori predefiniti.

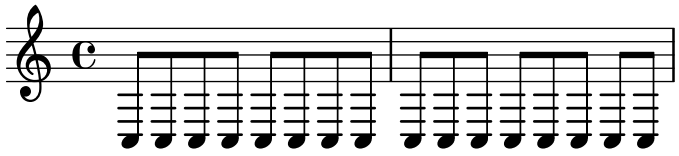
```
\relative a' {
  \time 6/8
  \repeat unfold 6 { a8 }
  % raggruppamento (4 + 2)
  \set Timing.beatStructure = 4,2
  \repeat unfold 6 { a8 }
  % ritorno al comportamento predefinito
  \time 6/8
  \repeat unfold 6 { a8 }
}
```



Le impostazioni predefinite della disposizione automatica delle travature per ogni tempo sono definite in `scm/time-signature-settings.scm`. La loro modifica è descritta in [Indicazione di tempo], pagina 67.
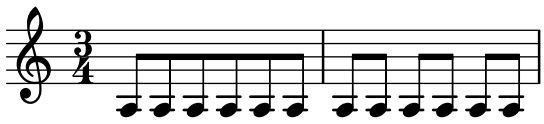
Molte impostazioni di travature automatiche per le indicazioni di tempo hanno un elemento `beamExceptions`. Ad esempio, il tempo 4/4 cerca di creare due travature nella misura se ci sono solo note di un ottavo. La regola `beamExceptions` può sovrascrivere l'impostazione di `beatStructure` se `beamExceptions` non viene annullato.

```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = 3,3,2
% Le travature non saranno raggruppate in (3 3 2) a causa di beamExceptions
\repeat unfold 8 {c8} |
% Il raggruppamento delle travature è (3 3 2) perché abbiamo tolto le impostazioni prec
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c8}
```



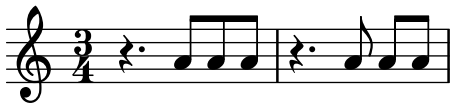
Analogalmente, le note di un ottavo in un tempo 3/4 sono raggruppate in un'unica travatura. Per raggrupparle secondo le suddivisioni, azzera `beamExceptions`.

```
\time 3/4
% il comportamento predefinito è un gruppo di (6) a causa di beamExceptions
\repeat unfold 6 {a8} |
% Le travature saranno raggruppate in (1 1 1) a causa dei valori predefiniti di baseMo
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a8}
```



Spesso, nelle partiture di età classica e romantica, le travature iniziano a metà della misura in un tempo 3/4; ma la pratica moderna preferisce evitare l'impressione ingannevole di un tempo 6/8 (vedi Gould, p. 153). Situazioni simili si incontrano anche per il tempo 3/8. Questo comportamento è controllato dalla proprietà di contesto `beamHalfMeasure`, che ha effetto soltanto sulle indicazioni di tempo che hanno 3 come numeratore:

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



### *Come funziona la disposizione automatica delle travature*

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata dalle proprietà di contesto `baseMoment`, `beatStructure` e `beamExceptions`.

Nel determinare l'aspetto delle travature vengono applicate le seguenti regole, in ordine di priorità:

- Se si specifica una travatura manuale con `[...]` imposta la travatura in quel modo, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per il tipo di travatura in questione, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per un tipo di travatura più lunga, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- usa i valori di `baseMoment` e `beatStructure` per determinare l'estensione delle suddivisioni della misura e terminare le travature alla fine delle suddivisioni.

Nelle regole precedenti, il *tipo di travatura* è la durata della nota più corta nel gruppo della travatura.

Le regole predefinite per le travature si trovano in `scm/time-signature-settings.scm`.

## Frammenti di codice selezionati

*Suddividere le travature*

Le travature di note consecutive di un sedicesimo (o più brevi) non vengono suddivise, ovvero i tre (o più) tratti della travatura si estendono, senza spezzarsi, sugli interi gruppi di note. Questo comportamento può essere modificato in modo da suddividere le travature in sottogruppi attraverso la proprietà `subdivideBeams`. Se impostata, le travature che comprendono più sottogruppi verranno suddivise a intervalli definiti dal valore attuale di `baseMoment`, riducendo le travature multiple al numero di travature che indica il valore metrico della suddivisione. Se il gruppo successivo alla suddivisione è più breve del valore metrico corrente (di solito perché la travatura è incompleta), il numero di travature riflette il gruppo di suddivisione più lungo possibile. Tuttavia, se rimane una sola nota dopo la divisione, questa restrizione non viene applicata. Si noti che `baseMoment`, se non impostata esplicitamente, equivale a uno fratto il denominatore dell'attuale indicazione di tempo. Deve quindi essere impostata su una frazione che stabilisca la durata del sottogruppo di travature; lo si può fare usando la funzione `ly:make-moment`, come è mostrato in questo frammento di codice. Inoltre quando `baseMoment` cambia, anche `beatStructure` deve essere modificato per accordarsi con `baseMoment`:

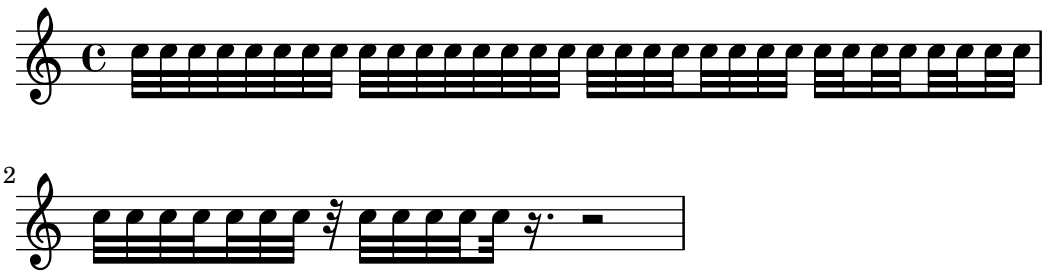
```
\relative c'' {
  c32[ c c c c c c c]
  \set subdivideBeams = ##t
  c32[ c c c c c c c]

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

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

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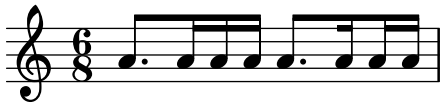
  % Shorten beam by 3/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = 2,2,2,2
  c32[ c c c c] r16.
  r2
}
```



*Travatura che segue strettamente il battito*

Si possono impostare i tratti di suddivisione della travatura in modo che siano rivolti verso la relativa pulsazione. La prima travatura fa sì che non spuntino i tratti di suddivisione (comportamento predefinito); la seconda travatura è orientata verso la pulsazione.

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



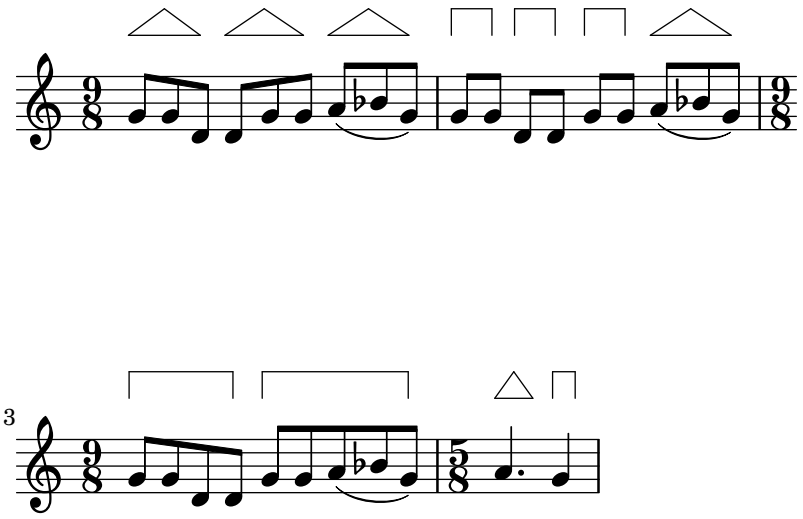
*Segni per la conduzione, segni di raggruppamento della misura*

Il raggruppamento delle pulsazioni all'interno della misura è regolato dalla proprietà di contesto `beatStructure`. I valori di `beatStructure` per varie indicazioni di tempo vengono stabiliti in `scm/time-signature-settings.scm`. Questi valori possono essere impostati o modificati con `\set`. Altrimenti, si può usare `\time` per impostare sia l'indicazione di tempo che la struttura delle pulsazioni. Per farlo si specifica il raggruppamento interno delle pulsazioni in una misura in una lista di numeri (nella sintassi di Scheme) prima dell'indicazione di tempo.

`\time` agisce nel contesto `Timing`, dunque non reimposterà i i valori di `beatStructure` e `baseMoment` che sono impostati in altri contesti di più basso livello, come `Voice`.

Se si include l'incisore `Measure_grouping_engraver` in uno dei contesti che regolano l'aspetto, appariranno i segni di raggruppamento della misura. Tali segni facilitano la lettura di musica moderna ritmicamente complessa. Nell'esempio la misura di 9/8 è raggruppata in due diversi schemi usando due metodi differenti, mentre la misura di 5/8 è raggruppata in base alle impostazioni predefinite in `scm/time-signature-settings.scm`:
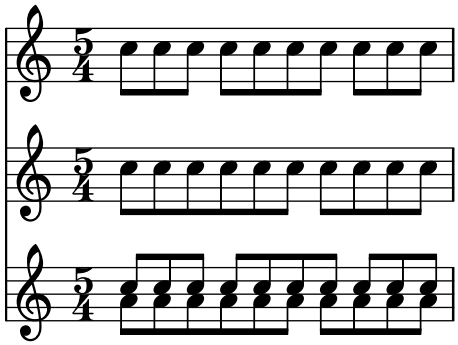
```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) |
    \time 4,5 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```



*Estremità delle travature nel contesto Score*

Le regole relative alle estremità delle travature definite nel contesto `Score` si applicano a tutti i righi, ma possono essere modificate anche ai livelli `Staff` e `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
        }
      >>
    }
  >>
}
```



## Vedi anche

Guida alla notazione: [Indicazione di tempo], pagina 67.

File installati: `scm/time-signature-settings.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

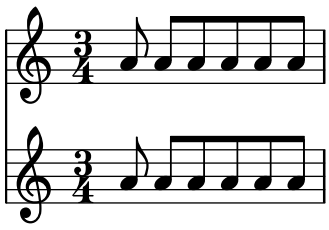
Guida al funzionamento interno: Sezione “Auto_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamForbidEvent” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se una partitura finisce prima del termine di una travatura automatica, cui mancano ancora delle note, quest'ultima travatura non apparirà. Lo stesso vale per le voci polifoniche, inserite con `<< ... \\ ... >>`. Una voce polifonica non apparirà se termina quando una travatura automatica è ancora in attesa di note. Per aggirare questi problemi occorre impostare manualmente l'ultima travatura della voce o della partitura.

`Timing` è un alias del contesto `Score`. Questo significa che la modifica della disposizione delle travature in un rigo avrà effetto anche sugli altri righi. Quindi un'impostazione di tempo in un rigo successivo reimposterà la disposizione personalizzata delle travature definita in un rigo precedente. Per evitare questo problema si può impostare l'indicazione di tempo su un solo rigo.

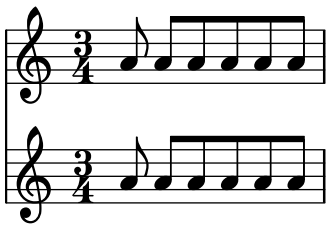
```
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1/8)
    \set Timing.beatStructure = 1,5
    \set Timing.beamExceptions = #'()
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \repeat unfold 6 { a'8 }
  }
>>
```



Si possono cambiare anche le impostazioni predefinite delle travature, in modo che sia usata sempre la disposizione delle travature desiderata. Le modifiche nelle impostazioni della travatura automatica per le indicazioni di tempo sono descritte in [Indicazione di tempo], pagina 67.
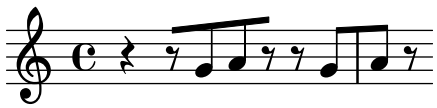
```
<<
```

```
  \new Staff {
    \overrideTimeSignatureSettings
      3/4               % timeSignatureFraction
      1/8               % baseMomentFraction
      1,5               % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
>>
```



## Travature manuali

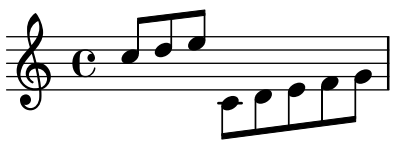
In alcuni casi potrebbe essere necessario scavalcare l'algoritmo di disposizione automatica delle travature. Ad esempio, questo algoritmo non inserirà delle travature tra le pause o tra le stanghette; e nelle partiture corali la disposizione delle travature è spesso determinato dall'articolazione del testo piuttosto che da quella musicale. Tali travature possono essere specificate manualmente indicandone l'inizio e la fine con `[` e `]`.

```
\relative { r4 r8[ g' a r] r g[ | a] r }
```
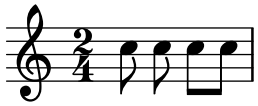


La direzione delle travature può essere impostata manualmente attraverso gli indicatori di direzione:

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



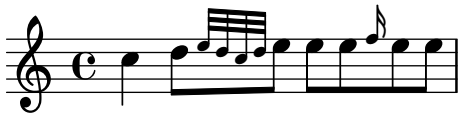
Le note individuali possono essere contrassegnate con `\noBeam` per impedire che vengano inserite in una travatura:

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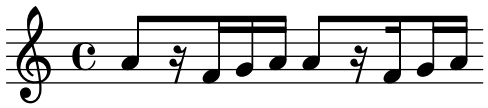


Le travature degli abbellimenti e quelle delle note normali possono coesistere simultaneamente. Gli abbellimenti privi di travatura non vengono inseriti nella travatura delle note normali.

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



Si può ottenere un controllo manuale delle travature ancora più preciso agendo sulle proprietà `stemLeftBeamCount` e `stemRightBeamCount`, che specificano il numero di travature da creare a sinistra e a destra della nota successiva. Se una di queste proprietà viene impostata, il suo valore verrà usato una volta sola, e la proprietà sarà poi cancellata. In questo esempio, l'ultima nota `f` ha una sola travatura a sinistra: la travatura corrispondente alla sottodivisione di un ottavo all'interno dell'intero raggruppamento.

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



## Comandi predefiniti

`\noBeam`.

## Frammenti di codice selezionati

*Code e punte delle travature*

È possibile ottenere delle codette su note isolate e dei tratti di suddivisione all'estremità della travatura con una combinazione di `stemLeftBeamCount`, `stemRightBeamCount` e una coppia di indicatori della travatura `[]`.

Per ottenere delle codette rivolte a destra, si usa la coppia di indicatori `[]` e si imposta `stemLeftBeamCount` a zero (vedi Example 1).

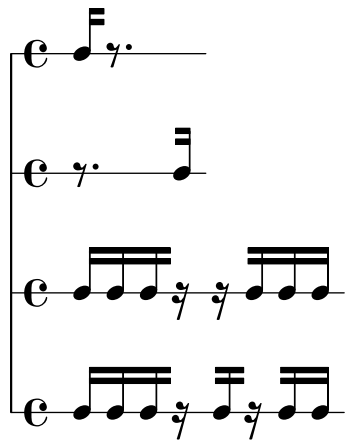
Per ottenere delle codette rivolte a sinistra, si imposta invece `stemRightBeamCount` (Example 2).

Perché i tratti di suddivisione alla fine di un gruppo di note unite da travatura siano rivolti a destra, si imposta `stemRightBeamCount` su un valore positivo. Perché i tratti di suddivisione all'inizio di un gruppo di note unite da travatura siano rivolti a sinistra, si imposta invece `stemLeftBeamCount` (Example 3).

Talvolta, ad esempio per una nota isolata circondata da pause, ha senso avere una coda che punti sia a destra che a sinistra. Lo si può fare con una coppia di indicatori di travatura `[]` da soli (Example 4).

(Nota che `\set stemLeftBeamCount` è sempre equivalente a `\once \set`. In altre parole, le impostazioni che definiscono il conteggio delle travature non "permangono", quindi la coppia di code attaccate al `16[]` solitario nell'ultimo esempio non hanno nulla a che fare con l'impostazione `\set` di due note prima.)

```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
    \new RhythmicStaff {
      r8.
      \set stemRightBeamCount = #0
      16[]
    }
    % Example 3
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = #2
      16 r r
      \set stemLeftBeamCount = #2
      16 16 16
    }
    % Example 4
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = #2
      16 r16
      16[]
      r16
      \set stemLeftBeamCount = #2
      16 16
    }
  >>
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Abbellimenti], pagina 116.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamEvent” in *Guida al Funzionamento Interno*, Sezione “Beam_engraver” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*, Sezione “Stem_engraver” in *Guida al Funzionamento Interno*.

## Travature a raggiera
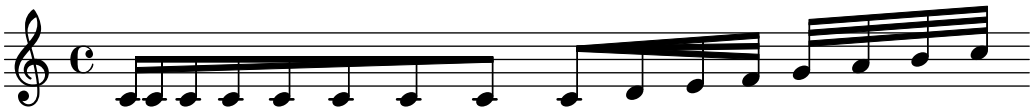
Le travature a raggiera servono a indicare che un gruppo di note determinato deve essere eseguito a un tempo progressivamente accelerato (o rallentato), senza cambiare l’andamento complessivo del brano. L’estensione della travatura a raggiera deve essere indicato a mano con `[` e `]`, e la convergenza o divergenza delle travature si determina specificando la la direzione della proprietà `Beam` di `grow-direction`.

Perché il *ritardando* o l’*accelerando* indicati dalla travatura a raggiera trovino riscontro nella disposizione delle note e nell’esecuzione del file MIDI, le note devono essere raggruppate in un’espressione musicale delimitata da parentesi graffe e preceduta dal comando `featherDurations`, che specifica il rapporto tra le durate delle prime e delle ultime note del gruppo.

Le parentesi quadre indicano l’estensione della travatura, mentre quelle graffe indicano quali note devono avere una durata modificata. Di norma queste parentesi delimitano lo stesso gruppo di note, ma questo non è tassativo: i due comandi sono indipendenti.

Nell’esempio seguente le otto note da un sedicesimo occupano esattamente lo stesso tempo di una nota di due quarti, ma la prima nota dura la metà dell’ultima e le note intermedie si allungano gradualmente. Le prime quattro note da un trentaduesimo sono progressivamente più veloci, mentre le ultime quattro presentano lo stesso tempo.

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



La spaziatura rappresenta la durata effettiva delle note solo in modo approssimato, mentre il tempo nel file MIDI è esatto.

### Comandi predefiniti

`\featherDurations`.

### Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

### Problemi noti e avvertimenti

Il comando `\featherDurations` funziona solamente con frammenti di musica molto brevi e quando i numeri della frazione sono piccoli.
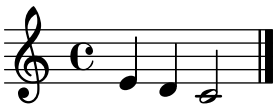
## 1.2.5 Battute

### Stanghette

Le stanghette delimitano le misure e sono usate anche per indicare i ritornelli. Di norma, le stanghette semplici sono inserite automaticamente in base all’indicazione di tempo.

Si possono inserire altri tipi di stanghette col comando `\bar`. Ad esempio, di solito si usa una stanghetta finale al termine di un brano:

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



Se l’ultima nota di una misura non termina entro la stanghetta inserita automaticamente, non viene segnalato un errore: si presuppone che la nota continui nella misura successiva. Ma se ci sono tante misure simili in sequenza, la musica potrebbe apparire compressa oppure scorrere fuori dalla pagina. Questo accade perché le interruzioni di linea automatiche si verificano solo al termine di misure complete, ovvero quando tutte le note terminano prima dell’inizio di una misura.

> **Nota:** Una durata errata può impedire un’interruzione di linea, causando una linea di musica altamente compressa oppure a musica che prosegue fuori dalla pagina.

Le interruzioni di linea sono permesse anche in caso si stanghette inserite a mano anche all’interno di misure incomplete. Per permettere un’interruzione di linea senza che appaia una stanghetta si usa:

```
\bar ""
```

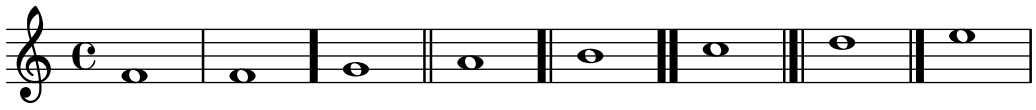
Questo comando inserirà una stanghetta invisibile e consentirà (senza però forzarla) un’interruzione di linea in questo punto. Il conteggio dei numeri di battuta non incrementa. Per forzare un’interruzione di linea si veda Sezione 4.3.1 [Interruzioni di linea], pagina 554.

Si possono inserire questa e altre stanghette speciali in qualsiasi punto. Quando coincidono con la fine di una misura, sostituiscono la stanghetta semplice che sarebbe stata posta automaticamente. Quando non coincidono con la fine di una misura, la stanghetta specificata viene inserita in quel punto.

Si noti che le stanghette manuali hanno una funzione puramente visiva. Non hanno alcun effetto sulle proprietà di una normale stanghetta, come i numeri della misura, le alterazioni, le interruzioni di linea, etc. Non influiscono nemmeno sul conteggio e sulla posizione delle stanghette automatiche successive. Quando una stanghetta manuale è posta nel punto in cui si trova già una normale stanghetta, le caratteristiche della stanghetta originale non sono alterate.

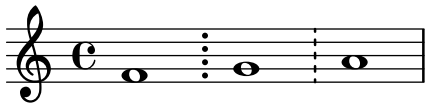
Sono disponibili per l'inserimento manuale due tipi di stanghette semplici e cinque tipi di stanghette doppie:

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



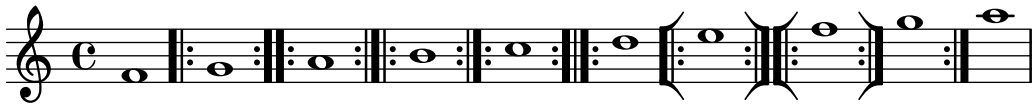
oltre alle stanghette puntate e tratteggiate:

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



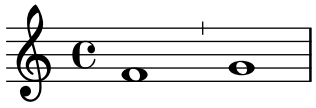
e a nove tipi di stanghette per le ripetizioni:

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



Inoltre, una stanghetta può apparire come un semplice segno di spunta:

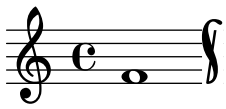
```
f'1 \bar "'" g'1
```



Tuttavia,dato che questi segni di spunta sono tipicamente usati nella notazione gregoriana, è preferibile usare `\divisioMinima`, come è descritto nella sezione [Divisiones], pagina 454, della parte dedicata al canto gregoriano.

LilyPond supporta la notazione gregoriana russa e fornisce una stanghetta speciale per questo tipo di notazione:

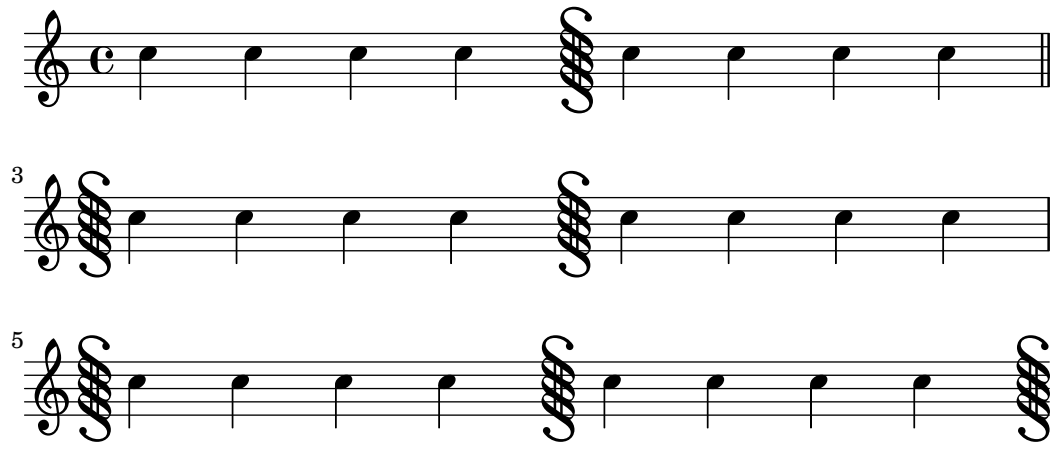
```
f'1 \bar "k"
```

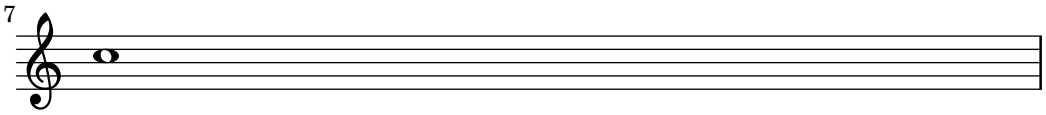


I dettagli di questo tipo di notazione sono spiegati in Sezione 2.9.5 [Scrivere in notazione quadrata di Kiev], pagina 462.

Per i segni di tipo segno interni al rigo, ci sono tre tipi di stanghette che differiscono nel comportamento quando incontrano un'interruzione di linea:

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

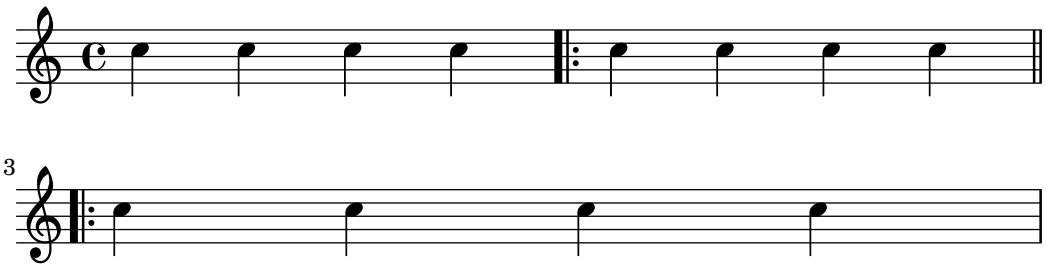




Sebbene LilyPond preveda l'inserimento manuale delle stanghette che indicano i ritornelli, ciò non consente il riconoscimento della musica come una sezione da ripetere. Tali sezioni devono essere inserite con i vari comandi di ripetizione (vedi Sezione 1.4 [Ripetizioni], pagina 152), che creano automaticamente le stanghette appropriate.

Inoltre si può specificare `".|:-||"`, che è equivalente a `".|:"` tranne in presenza di un'interruzione di linea, dove crea una doppia stanghetta alla fine della linea e una stanghetta di inizio ripetizione all'inizio della linea successiva.
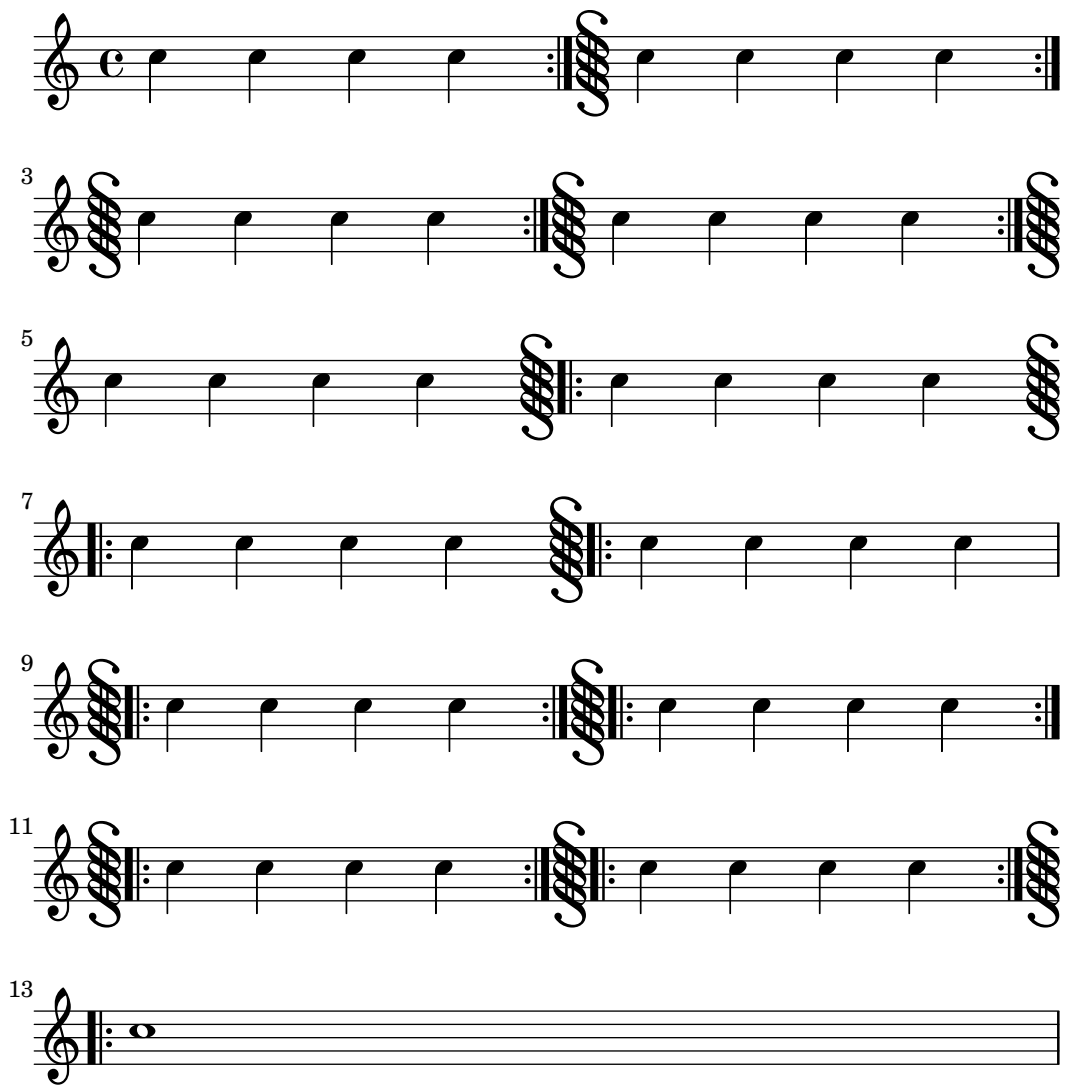
```
\relative c'' {
  c4 c c c
  \bar ".|:-||"
  c4 c c c \break
  \bar ".|:-||"
  c4 c c c
}
```



Esistono sei diverse combinazioni di ripetizioni e indicazioni di segno:

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

```
  c1
}
```




Esiste inoltre un comando `\inStaffSegno` che crea una stanghetta con segno in congiunzione con un'appropriata stanghetta di ripetizione se usata con un comando `\repeat volta`, vedi [Ripetizioni normali], pagina 153..

Si possono definire nuovi tipi di stanghette con `\defineBarLine`:

`\defineBarLine` *`tipo-stanghetta #'(fine inizio span)`*

Le variabili di `\defineBarLine` possono includere la stringa 'vuota' `""`,che è equivalente a una stanghetta invisibile. Oppure possono essere impostate su `#f`, che fa sì che non appaia alcuna stanghetta.

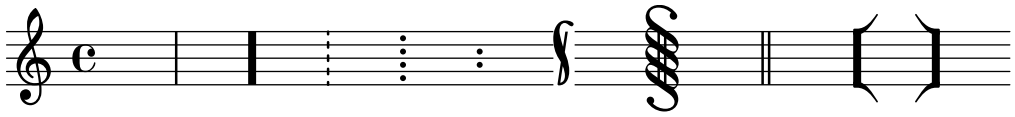
Dopo averla definita, si può richiamare la nuova stanghetta col comando `\bar` *tipo-stanghetta*.

Attualmente sono disponibile dieci tipi di stanghetta:

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

\new Staff {
```

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

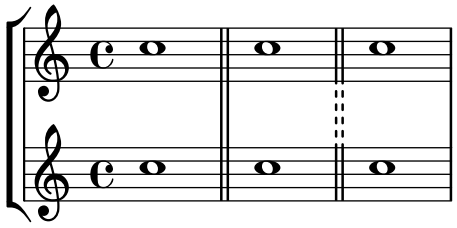


La stanghetta `"="` crea una stanghetta doppia da combinare con il il segno. Non va usata per creare una stanghetta doppia indipendente; in questo caso è preferibile usare `\bar "||"`.

Il segno `"-"` introduce le annotazioni alle stanghette che servono a distinguere quelle che hanno aspetto identico ma un diverso comportamento in corrispondenza delle interruzioni di linea e/o un diverso modo di connettere le stanghette tra i righi. La parte che segue il segno `"-"` non viene usato per costruire la stanghetta.

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

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

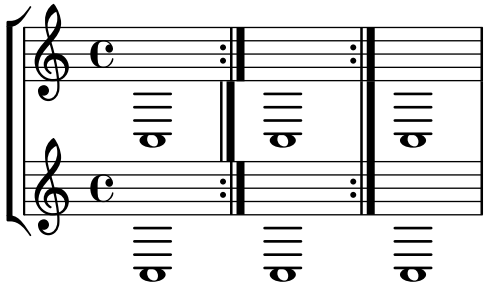


Inoltre, lo spazio `" "` fa da spaziatore e fa sì che le stanghette tra i righi siano allineate correttamente alle stanghette principali:

```
\defineBarLine ":|.-sbagliata" #'(":|." "" "|.")
\defineBarLine ":|.-giusta" #'(":|." "" " |.")

\new StaffGroup <<
```
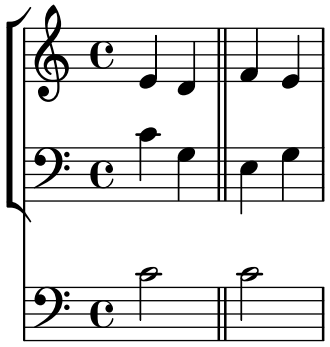
```
  \new Staff {
    c1 \bar ":|.-sbagliata"
    c1 \bar ":|.-giusta"
    c1
  }
  \new Staff {
    c1
    c1
    c1
  }
>>
```



Se servono ulteriori elementi, LilyPond fornisce un modo semplice per definirli. Maggiori informazioni sulla modifica e l'aggiunta delle stanghette sono presenti nel file `scm/bar-line.scm`.

Nelle partiture con molti righi, un comando `\bar` inserito in un rigo viene applicato automaticamente a tutti i righi. Le stanghette risultanti sono connesse tra i diversi righi di un `StaffGroup`, `PianoStaff` o `GrandStaff`.

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



Il comando '`\bar` *`tipo-stanghetta`*' è una scorciatoia di '`\set Timing.whichBar =` *`tipo-stanghetta`*'. Una stanghetta viene creata ogni volta che si imposta la proprietà `whichBar`.

Il tipo di stanghetta predefinita per le stanghette inserite automaticamente è `"|"`. Si può modificare in qualsiasi momento con '`\set Timing.defaultBarType = tipo-stanghetta`'.

## Vedi anche

Guida alla notazione: Sezione 4.3.1 [Interruzioni di linea], pagina 554, Sezione 1.4 [Ripetizioni], pagina 152, [Raggruppare i righi], pagina 195.

File installati: `scm/bar-line.scm`.

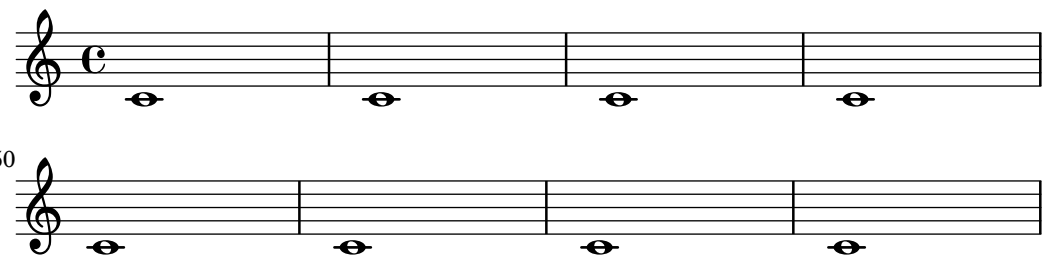
Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "BarLine" in *Guida al Funzionamento Interno* (creata al livello `Staff`), Sezione "SpanBar" in *Guida al Funzionamento Interno* (tra i righi), Sezione "Timing_translator" in *Guida al Funzionamento Interno* (per le proprietà di Timing).

## Numeri di battuta

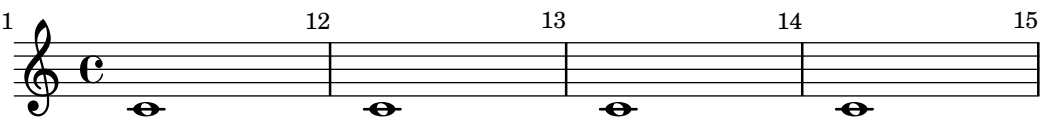
I numeri di battuta appaiono all'inizio di ogni linea tranne la prima. Il numero viene salvato nella proprietà `currentBarNumber`, che viene aggiornata automaticamente per ogni misura. Può anche essere impostata a mano:

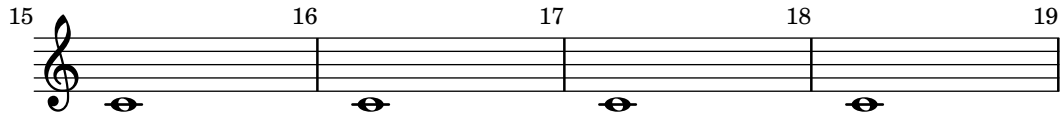
```
\relative c' {
  c1 c c c
  \break
  \set Score.currentBarNumber = #50
  c1 c c c
}
```




I numeri di battuta possono essere mostrati a intervalli regolari anziché solo all'inizio di ogni linea. Per farlo occorre sovrascrivere il comportamento predefinito e permettere ai numeri di battuta di apparire anche in punti diversi dall'inizio della linea. Questo comportamento è regolato dalla proprietà `break-visibility` di `BarNumber`, che considera tre valori impostabili su `#t` o `#f`, i quali indicano se il numero di battuta corrispondente debba essere visibile o no. L'ordine dei tre valori è `end of line visible`, `middle of line visible`, `beginning of line visible`. Nell'esempio seguente i numeri di battuta compaiono in tutti i punti possibili:

```
\relative c' {
  \override Score.BarNumber.break-visibility = ##(#t #t #t)
  \set Score.currentBarNumber = #11
  % Permette la visualizzazione del primo numero di battuta
  \bar ""
  c1 | c | c | c
  \break
  c1 | c | c | c
}
```





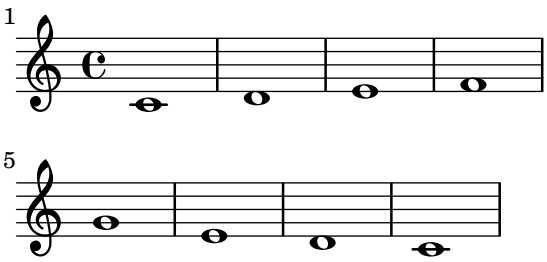

## Frammenti di codice selezionati

*Mostrare il numero di battuta nella prima misura*

Il primo numero di battuta di una partitura viene soppresso se è inferiore o uguale a '1'. Se si imposta `barNumberVisibility` su `all-bar-numbers-visible`, verrà mostrato il numero di battuta della prima misura e di tutte quelle successive. Si noti che perché funzioni è necessario inserire una stanghetta invisibile prima della prima nota.

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

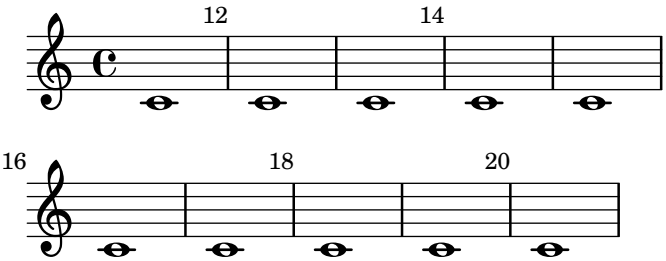
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```




*Mostrare i numeri di battuta a intervalli regolari*

I numeri di battuta possono essere resi visbili a intervalli regolari attraverso la proprietà `barNumberVisibility`. In questo esempio vengono mostrati ogni due misure eccetto alla fine della linea.

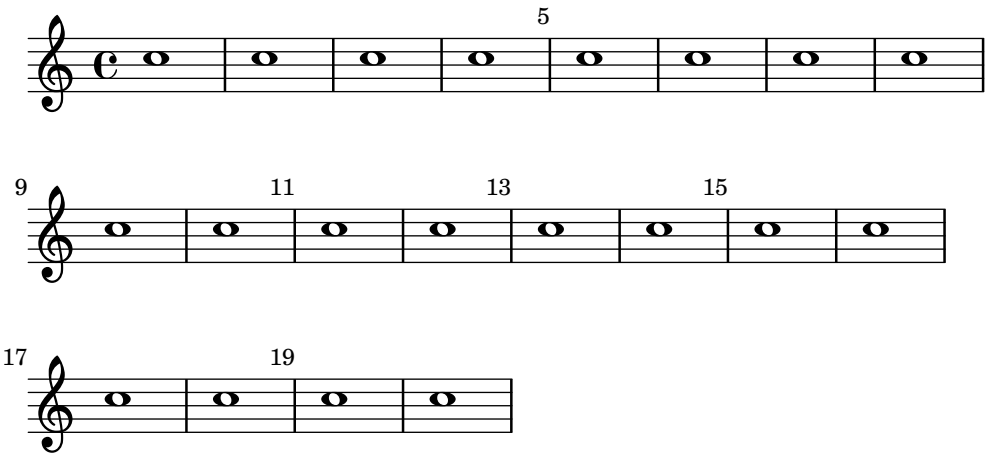
```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c |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



*Stampare i numeri di battuta a intervalli regolari variabili*

L'intervallo dei numeri di battuta può essere modificato cambiando la funzione di contesto `{set-bar-number-visibility}`.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \context Score \applyContext #(set-bar-number-visibility 4)
  \repeat unfold 10 c'1
  \context Score \applyContext #(set-bar-number-visibility 2)
  \repeat unfold 10 c
}
```




*Numeri di battuta racchiusi in rettangoli o cerchi*

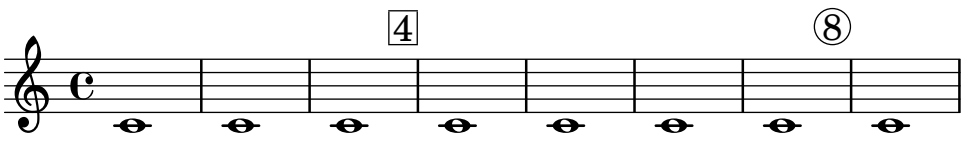
I numeri di battuta possono apparire anche all'interno di rettangoli o cerchi.

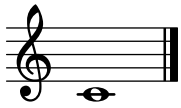
```
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % 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 "|."
}
```
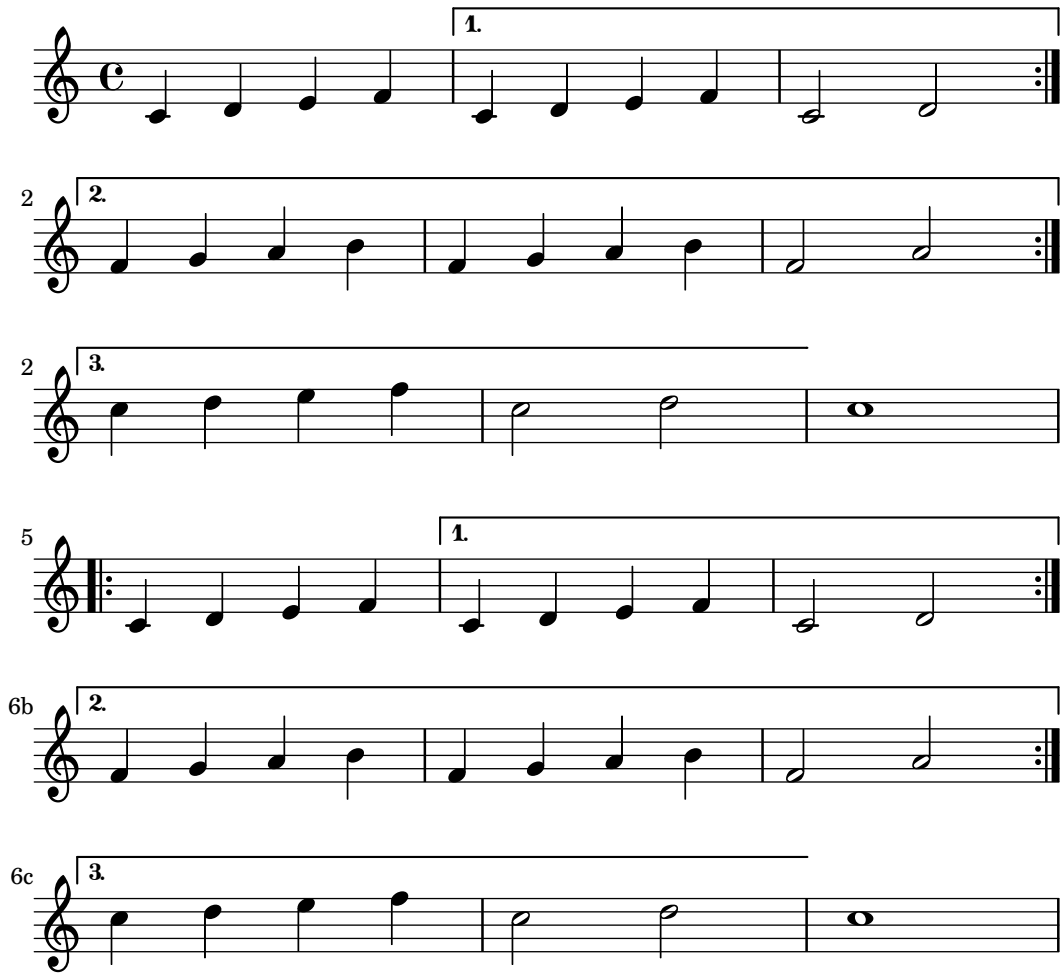




*Numeri di battuta alternativi*

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

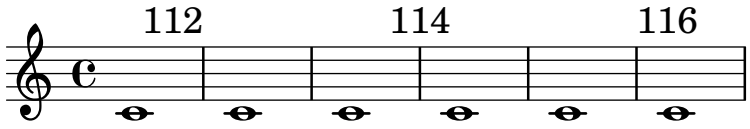
```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```



*Allineare i numeri di battuta*

Per impostazione predefinita i numeri di battuta sono allineati a destra rispetto al loro oggetto genitore. Di solito si tratta del margine sinistro della linea oppure, se i numeri appaiono all'interno della linea, del lato sinistro della stanghetta. I numeri possono essere posizionati anche direttamente sopra la stanghetta oppure allineati a sinistra della stanghetta.

```
\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
}
```
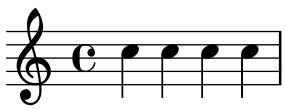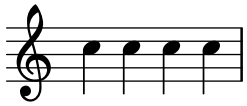



*Togliere i numeri di battuta da uno spartito*

I numeri di battuta possono essere tolti rimuovendo l'incisore `Bar_number_engraver` dal contesto `Score`.

```
\layout {
  \context {
    \Score
    \omit BarNumber
    % or:
    %\remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```





## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BarNumber” in *Guida al Funzionamento Interno*, Sezione “Bar_number_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

I numeri di battuta possono collidere con la parte superiore della parentesi quadra di `StaffGroup`, se presente. Per evitare la collisione, si può usare la proprietà `padding` di `BarNumber` per posizionare correttamente il numero. Si veda Sezione “StaffGroup” in *Guida al Funzionamento Interno* e Sezione “BarNumber” in *Guida al Funzionamento Interno* per maggiori informazioni.

## Controlli di battuta e del numero di battuta

I controlli di battuta aiutano a rilevare gli errori di durata. Il controllo di battuta si inserisce col simbolo della barra verticale, `|`, in un qualsiasi punto in cui è previsto l’inserimento di una stanghetta. Se vengono trovati controlli di battuta in punti diversi, viene creata una lista di avvisi nel file di log che mostra i numeri di linea e le linee in cui il controllo è fallito. Nell’esempio seguente il secondo controllo di battuta segnalerà un errore.

```
\time 3/4 c2 e4 | g2 |
```

Una durata non corretta può generare uno spartito completamente alterato, specialmente nel caso di brani polifonici. Quindi il primo passo da compiere per correggere l’input è la verifica dei controlli di battuta e delle durate errate.

Se i controlli di battuta successivi sono spostati dello stesso intervallo musicale, viene mostrato solo il primo messaggio di avviso. Così l’avvertimento si concentra sulla causa dell’errore di tempo.
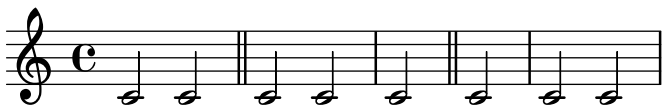
I controlli di battuta possono essere usati anche all’interno del testo vocale:

```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle |
}
```

Attenzione: i segni di controllo di ottava nel testo vocale sono elaborati nel momento musicale in cui la sillaba *che segue* il segno di controllo viene elaborata. Se il testo è associato alle note di una voce che ha una pausa all’inizio di una battuta, non è possibile individuare alcuna sillaba all’inizio di quella battuta e apparirà un avvertimento se viene posto un controllo di battuta in quel punto del testo vocale.

È anche possibile ridefinire l’azione da prendere quando si incontra un controllo di battuta o simbolo di barra verticale, `|`, nell’input, in modo che avvenga qualcosa di diverso dal controllo di battuta. Si può fare assegnando un’espressione musicale a `"|"`. Nell’esempio seguente `|`, invece di controllare la fine di una battuta, viene usato per inserire una stanghetta doppia ovunque appaia nell’input.

```
"|" = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Quando si copiano brani di una certa ampiezza, può essere d'aiuto verificare che i numeri di battuta di LilyPond corrispondano all'originale a partire dal quale si sta scrivendo il brano. Si può abilitare con `\barNumberCheck`, ad esempio,

```
\barNumberCheck #123
```

genererà un avvertimento se `currentBarNumber` non è 123 nel momento in cui viene elaborato.

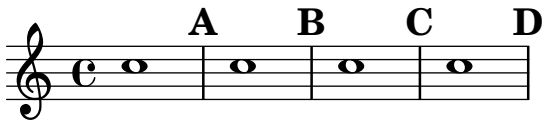
### Vedi anche

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

### Segni di chiamata

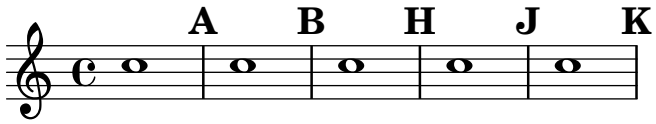
Per creare un segno di chiamata si usa il comando `\mark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}
```




Il segno viene incrementato automaticamente se si usa `\mark \default`, ma è possibile usare anche un numero intero come argomento in modo da impostare il segno manualmente. Il valore da usare viene salvato nella proprietà `rehearsalMark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```

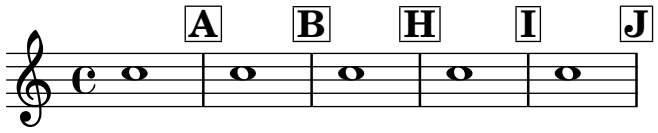



La lettera 'I' viene saltata, come vuole la tradizione tipografica. Se si desidera includere la lettera 'I', si può usare uno dei seguenti comandi, a seconda dello stile che si vuole (solo lettere, lettere in un quadrato o lettere in un cerchio).

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

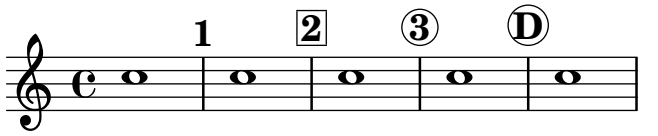
```
\relative c'' {
  \set Score.markFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
```

```
}
```




Lo stile viene definito dalla proprietà `markFormatter`. È una funzione che accoglie come argomenti il segno corrente (un numero intero) e il contesto corrente. Dovrebbe restituire un oggetto testuale. Nell'esempio seguente, `markFormatter` viene prima impostato su una procedura predefinita e dopo alcune misure su una procedura che produce un numero racchiuso in un quadrato.

```
\relative c'' {
  \set Score.markFormatter = #format-mark-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = #format-mark-box-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-letters
  c1
}
```




Il file `scm/translation-functions.scm` contiene le definizioni di `format-mark-letters` (il formato predefinito), `format-mark-box-letters`, `format-mark-numbers` e `format-mark-box-numbers`. Possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

Si possono usare `format-mark-barnumbers`, `format-mark-box-barnumbers` e `format-mark-circle-barnumbers` per ottenere i numeri di battuta invece di numeri o lettere crescenti.

Si possono specificare manualmente altri stili di segni di chiamata:

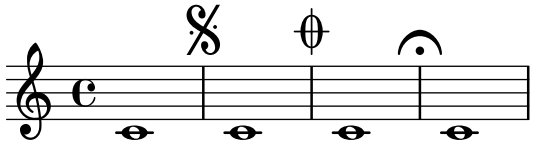
```
\mark "A1"
```

Si noti che `Score.markFormatter` non ha effetto sui segni specificati in questo modo. Tuttavia, è possibile applicare un `\markup` alla stringa.

```
\mark \markup { \box A1 }
```

I glifi musicali (come il Segno) possono essere posti dentro il comando `\mark`

```
\relative c' {
  c1 \mark \markup { \musicglyph "scripts.segno" }
  c1 \mark \markup { \musicglyph "scripts.coda" }
  c1 \mark \markup { \musicglyph "scripts.ufermata" }
  c1
}
```



L'elenco dei simboli che possono essere prodotti con `\musicglyph` si trova in Sezione A.8 [Il font Emmentaler], pagina 679.

Per le più comuni modifiche relative al posizionamento dei segni di chiamata, si veda Sezione 1.8.2 [Formattazione del testo], pagina 246. Per ottenere un controllo più preciso si consiglia di studiare il funzionamento della proprietà `break-alignable-interface` descritta in Sezione 5.5.1 [Allineamento degli oggetti], pagina 646.

Il file `scm/translation-functions.scm` contiene le definizioni di `format-mark-numbers` e `format-mark-letters`, che possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

### Vedi anche

Guida alla notazione: Sezione A.8 [Il font Emmentaler], pagina 679, Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione 5.5.1 [Allineamento degli oggetti], pagina 646.

File installati: `scm/translation-functions.scm`.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.
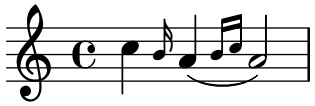
Guida al funzionamento interno: Sezione "MarkEvent" in *Guida al Funzionamento Interno*, Sezione "Mark_engraver" in *Guida al Funzionamento Interno*, Sezione "RehearsalMark" in *Guida al Funzionamento Interno*.

## 1.2.6 Questioni ritmiche particolari

### Abbellimenti
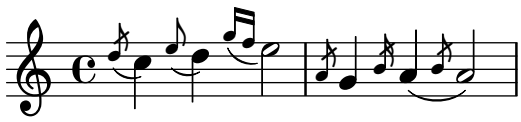
Gli abbellimenti sono degli ornamenti musicali che hanno un carattere in corpo più piccolo e non alterano la durata della misura.

```
\relative {
  c''4 \grace b16 a4(
  \grace { b16 c16 } a2)
}
```
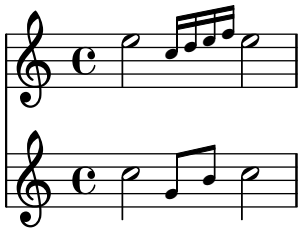


Esistono altri tre tipi di abbellimenti possibili; l'*acciaccatura* – un abbellimento in tempo libero indicato da una nota con legatura di portamento e un gambo barrato – e l'*appoggiatura*, che sottrae un valore determinato della nota principale a cui corrisponde e ha un gambo non barrato. È anche possibile creare un abbellimento con gambo barrato, come l'acciaccatura, ma privo di legatura di portamento, in modo da collocarla tra note già poste sotto una legatura: si usa il comando `\slashedGrace`.

```
\relative {
  \acciaccatura d''8 c4
  \appoggiatura e8 d4
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
  \slashedGrace b8 a2)
}
```



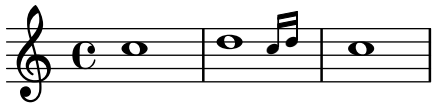
Il posizionamento degli abbellimenti è sincronizzato sui diversi righi. Nell'esempio seguente, ci sono due abbellimenti da un sedicesimo ogni abbellimento da un ottavo

```
<<
  \new Staff \relative { e''2 \grace { c16 d e f } e2 }
  \new Staff \relative { c''2 \grace { g8 b } c2 }
>>
```



Se si desidera risolvere una nota su un abbellimento, si usa il comando `\afterGrace`. Considera due argomenti: la nota principale e gli abbellimenti che la seguono.

```
\relative { c''1 \afterGrace d1 { c16[ d] } c1 }
```
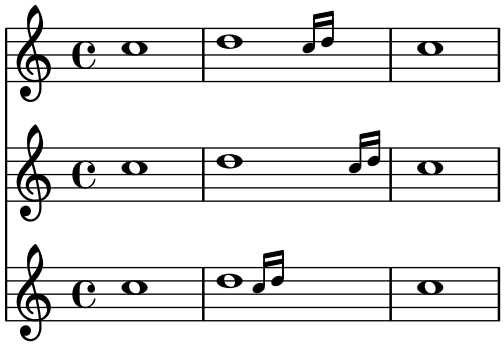


In questo modo, gli abbellimenti sono collocati *dopo* l'inizio della nota principale. Il momento temporale in cui sono posti gli abbellimenti è una certa frazione della durata della nota principale. L'impostazione predefinita di

```
afterGraceFraction = 3/4
```

può essere ridefinita nel livello superiore. Ma è anche possibile specificare la frazione per ogni singolo comando `\afterGrace`.
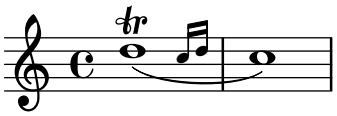
L'esempio seguente mostra le diverse spaziature che si ottengono con la frazione predefinita, impostandola a `15/16` e infine a `1/2` della nota principale.

```
<<
  \new Staff \relative {
    c''1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff \relative {
    c''1 \afterGrace 15/16 d1 { c16[ d] } c1
  }
  \new Staff \relative {
    c''1 \afterGrace 1/2 d1 { c16[ d] } c1
  }
>>
```
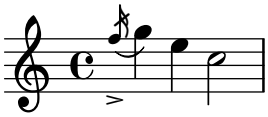


Lo spazio tra la nota principale e l'abbellimento può essere definito anche attraverso delle pause spaziatrici. L'esempio seguente sposta l'abbellimento di uno spazio equivalente ai 7/8 della nota principale.

```
\new Voice \relative {
  <<
    { d''1^\trill_( }
    { s2 s4. \grace { c16 d } }
  >>
  c1)
}
```



L'espressione musicale introdotta dal comando `\grace` avrà delle impostazioni tipografiche speciali; per esempio, per rimpicciolire il tipo di carattere e impostare le direzioni. Dunque le modifiche che sovrascrivono tali impostazioni speciali devono essere poste all'interno del blocco `\grace`. Lo stesso vale per le modifiche che ripristinano i valori predefiniti. Nell'esempio seguente la direzione predefinita del gambo viene prima sovrascritta e poi ripristinata.

```
\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
}
```
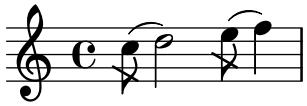


## Frammenti di codice selezionati

*Usare il gambo barrato degli abbellimenti con le teste normali*

Il gambo barrato presente nelle acciaccature può essere applicato in altre situazioni.
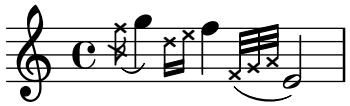
```
\relative c'' {
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}
```



*Modificare l'aspetto degli abbellimenti di un intero brano*

L'aspetto di tutte le espressioni contenute nei blocchi `\grace` di un brano può essere modificato con le funzioni `add-grace-property` e `remove-grace-property`. L'esempio seguente toglie la definizione della direzione di `Stem` nell'abbellimento, in modo che gli abbellimenti non siano sempre rivolti in su, e barra le teste di nota.

```
\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
    }
  }
}
```
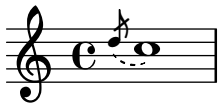


*Ridefinire le impostazioni predefinite globali degli abbellimenti*

Le impostazioni globali predefinite degli abbellimenti sono salvate negli identificatori `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` e `stopAppoggiaturaMusic`, che sono definiti nel file `ly/grace-init.ly`. Ridefinendoli si possono ottenere effetti diversi.

```
startAcciaccaturaMusic = {
  <>(
  \override Flag.stroke-style = #"grace"
  \slurDashed
}

stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>)
}

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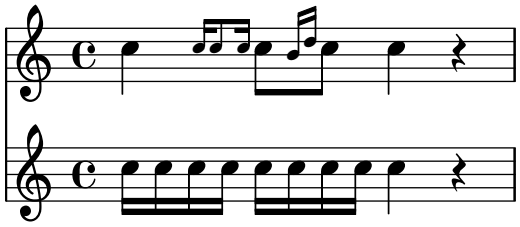


*Posizionare gli abbellimenti con dello spazio fluttuante*

Se si imposta la proprietà `'strict-grace-spacing`, le colonne musicali degli abbellimenti 'fluttuano', ovvero si scollegano dalle note normali: prima vengono spaziate le note normali, poi le colonne musicali degli abbellimenti vengono messe a sinistra delle colonne delle note principali.

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

```
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```



## Vedi anche

Glossario musicale: Sezione “acciaccatura” in *Glossario Musicale*, Sezione “acciaccatura” in *Glossario Musicale*, Sezione “appoggiatura” in *Glossario Musicale*.

Guida alla notazione: [Scalare le durate], pagina 53, [Travature manuali], pagina 97.

File installati: `ly/grace-init.ly`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.
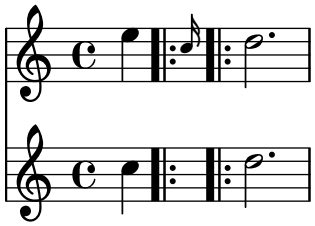
Guida al funzionamento interno: Sezione “GraceMusic” in *Guida al Funzionamento Interno*, Sezione “Grace_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_auto_beam_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace_spacing_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Una *acciaccatura* con molte note raggruppate sotto una travatura è priva della barra trasversale e ha il medesimo aspetto di una *appoggiatura* composta da varie note raggruppate sotto una travatura.
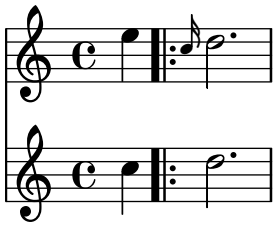
La sincronizzazione degli abbellimenti può nascondere delle sorprese, perché vengono sincronizzati anche altri elementi della notazione del rigo, come le armature di chiave, le stanghette, etc. Fai attenzione quando metti insieme righi che hanno degli abbellimenti con righi che non ne hanno. Ad esempio

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" d2. }
>>
```



Si può ovviare a questo problema inserendo degli abbellimenti della durata corrispondente negli altri righi. Riprendendo l'esempio precedente

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" \grace s16 d2. }
>>
```



L'uso del comando `\grace` nella parte con le pause spaziatrici è obbligatorio, anche se la parte visibile usa `\acciaccatura` o `\appoggiatura`, perché altrimenti apparirà un orribile frammento di legatura di portamento che connette la nota di abbellimento invisibile alla nota seguente.

Le sezioni con abbellimenti devono essere usate solamente all'interno di espressioni musicali sequenziali. Non è permesso annidare o affiancare gruppi di abbellimenti; potrebbero verificarsi blocchi del programma o altri errori se non si rispetta questa limitazione.

Ogni abbellimento generato nell'output MIDI ha una durata di 1/4 della sua vera durata. Se la durata complessiva degli abbellimenti è maggiore della durata della nota che li precede, verrà generato l'errore "`Going back in MIDI time`". A meno che non si diminuisca la durata degli abbellimenti, ad esempio:

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

diventa:

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

Oppure si cambia esplicitamente la durata musicale:

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

Vedi [Scalare le durate], pagina 53.

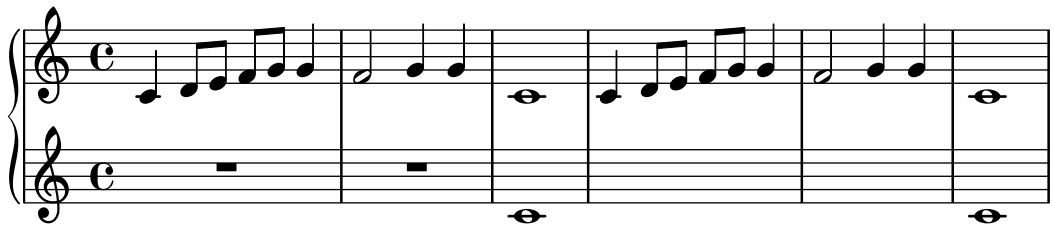
## Allineamento sulle cadenze

Nell'ambito di una partitura per orchestra, le cadenze presentano un problema peculiare: quando si scrive una partitura che include una cadenza o un altro passaggio solistico, tutti gli altri strumenti devono saltare esattamente la durata complessiva delle note del passaggio, altrimenti inizieranno troppo presto o troppo tardi.

Una possibile soluzione a questo problema consiste nell'uso delle funzioni `mmrest-of-length` e `skip-of-length`. Queste funzioni Scheme prendono come argomento una sezione di musica salvata in una variabile e generano una pausa multipla o `\skip` della lunghezza esatta del brano.

```
MyCadenza = \relative {
  c'4 d8 e f g g4
  f2 g4 g
}
```

```
\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
    c'1
  }
>>
```



## Vedi anche

Glossario musicale: Sezione “cadenza” in *Glossario Musicale*.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Gestione del tempo

Il tempo è gestito da `Timing_translator`, che si trova nel contesto `Score`. Un suo alias, `Timing`, viene aggiunto nel contesto nel quale si trova `Timing_translator`. Per assicurarsi che l’alias `Timing` sia disponibile, occorre istanziare esplicitamente il contesto che lo contiene (come `Voice` o `Staff`).

Si usano le seguenti proprietà di `Timing` per tenere traccia del tempo in una partitura.

`currentBarNumber`
: Il numero di battuta corrente. Un esempio che mostra l’uso di questa proprietà si trova in [Numeri di battuta], pagina 108.

`measureLength`
: La durata delle misure nel tempo corrente. Per un tempo di 4/4 è 1, per un tempo di 6/8 è 3/4. Il suo valore determina quando debbano essere inserite le stanghette e come debbano essere generate le travature automatiche.
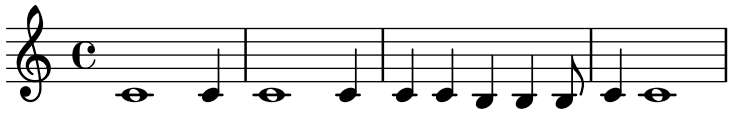
`measurePosition`
: Il punto della misura in cui ci si trova. Questa quantità viene reimpostata sottraendo `measureLength` ogni volta che `measureLength` viene raggiunto o superato. Quando questo accade, `currentBarNumber` viene incrementato.

`timing`
: Se impostato su `#t`, le variabili precedenti sono aggiornate ad ogni momento temporale. Se impostato su `#f`, l’incisore rimane nella misura corrente per un tempo indefinito.

Si può cambiare il tempo impostando esplicitamente una qualsiasi di queste variabili. Nel prossimo esempio, viene visualizzata l’indicazione di tempo predefinita di 4/4, ma `measureLength` è impostato su 5/4. A 4/8 della terza misura, `measurePosition` si sposta in

avanti di 1/8 fino a 5/8, diminuendo quella misura di 1/8. Quindi la stanghetta successiva si troverà a 9/8 invece che a 5/4.

```
\new Voice \relative {
  \set Timing.measureLength = #(ly:make-moment 5/4)
  c'1 c4 |
  c1 c4 |
  c4 c
  \set Timing.measurePosition = #(ly:make-moment 5/8)
  b4 b b8 |
  c4 c1 |
}
```



Come mostra l'esempio, `ly:make-moment n/m` definisce una durata di n/m della nota intera. Ad esempio, `ly:make-moment 1/8` corrisponde alla durata di un ottavo mentre `ly:make-moment 7/16` a quella di sette sedicesimi.
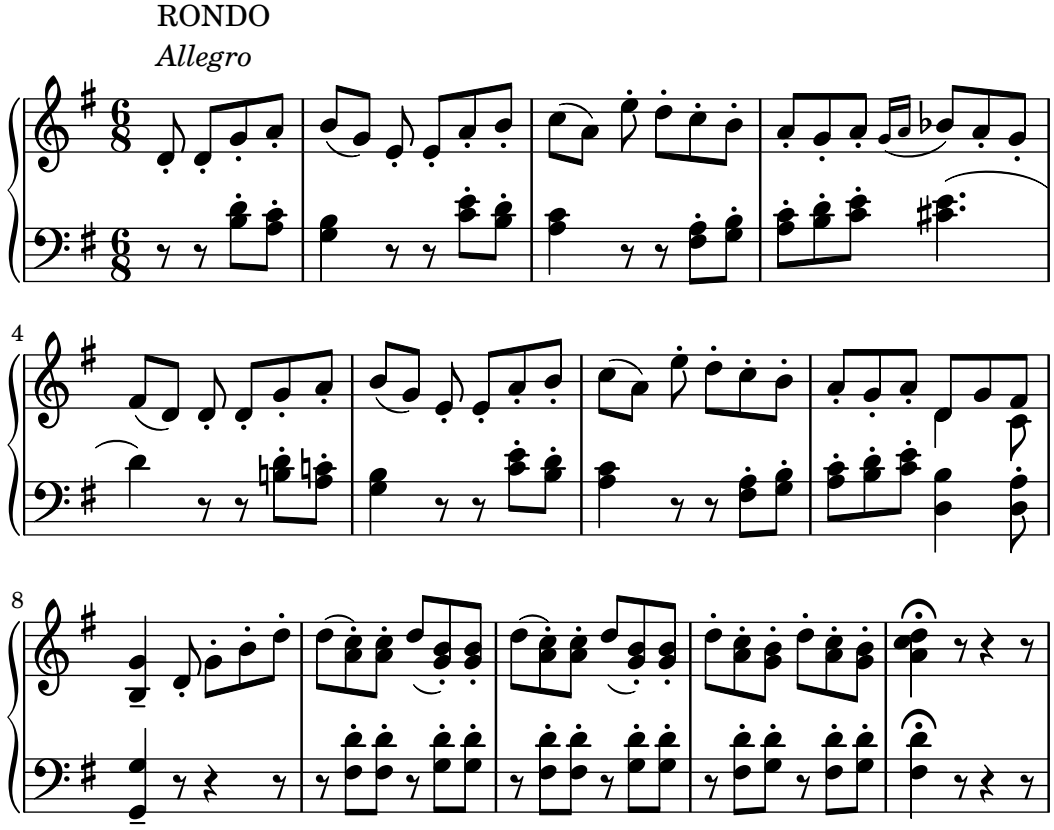
### Vedi anche

Guida alla notazione: [Numeri di battuta], pagina 108, [Musica in tempo libero], pagina 76.

Frammenti di codice: Sezione "Rhythms" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Timing_translator" in *Guida al Funzionamento Interno*, Sezione "Score" in *Guida al Funzionamento Interno*.

## 1.3 Segni di espressione



Questa sezione elenca vari segni di espressione che si possono usare in una partitura.

### 1.3.1 Segni di espressione collegati alle note

Questa sezione spiega come creare segni di espressione collegati alle note: articolazioni, abbellimenti e dinamiche. Sono trattati anche i metodi per creare nuove indicazioni dinamiche.
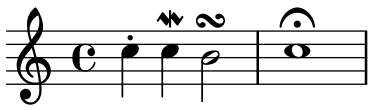
## Articolazioni e abbellimenti

I diversi simboli che rappresentano articolazioni, ornamenti e altre indicazioni esecutive possono essere collegati a una nota con questa sintassi:

```
nota\nome
```
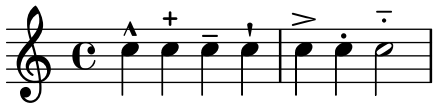
I valori possibili per *nome* sono elencati in Sezione A.14 [Elenco delle articolazioni], pagina 764. Ad esempio:

```
\relative {
  c''4\staccato c\mordent b2\turn
  c1\fermata
}
```



Alcune di queste articolazioni hanno delle abbreviazioni che ne semplificano l'inserimento. Le abbreviazioni sono attaccate al nome della nota e la loro sintassi è composta da un trattino `-` seguito da un simbolo che indica l'articolazione. Esistono abbreviazioni predefinite per *marcato*, *chiuso*, *tenuto*, *staccatissimo*, *accento*, *staccato* e *portato*. L'output corrispondente è:

```
\relative {
  c''4-^ c-+ c-- c-!
  c4-> c-. c2-_
}
```



Le regole per il posizionamento predefinito delle articolazioni sono definite in `scm/script.scm`. Articolazioni e ornamenti possono essere posizionati manualmente sopra o sotto il rigo; si veda Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Le articolazioni sono oggetti `Script`. Le loro proprietà sono descritte in dettaglio in Sezione "Script" in *Guida al Funzionamento Interno*.

Oltre alle articolazioni, si può attaccare alle note anche un testo, posto tra virgolette o in un blocco `\markup{}`. Si veda [Scritte], pagina 239.

Ulteriori informazioni sull'ordine degli oggetti Script e TextScript collegati alle note si trovano in Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

## Frammenti di codice selezionati

*Modificare i valori predefiniti per le abbreviazioni delle articolazioni*

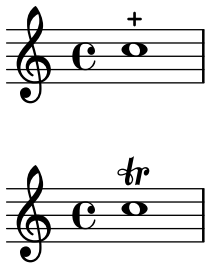
Le abbreviazioni sono definite in '`ly/script-init.ly`', dove sono assegnati valori predefiniti alle variabili `dashHat`, `dashPlus`, `dashDash`, `dashBang`, `dashLarger`, `dashDot` e `dashUnderscore`. Questi valori predefiniti possono essere modificati. Ad esempio, per associare

l'abbreviazione `-+` (`dashPlus`) al simbolo del *trillo* invece che al simbolo `+` predefinito, si assegna il valore `trill` alla variabile `dashPlus`:

```
\relative c'' { c1-+ }

dashPlus = "trill"

\relative c'' { c1-+ }
```
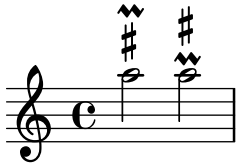




*Controllo dell'ordine verticale degli script*

L'ordine verticale degli script è determinato dalla proprietà `'script-priority`. Più il numero è piccolo, più sarà posto vicino alla nota. In questo esempio, il simbolo di *diesis* (oggetto `TextScript`) ha prima la priorità più bassa, dunque è posto più in basso nel primo esempio. Nel secondo, il *trillo* (oggetto `Script`) ha la priorità più bassa, quindi si trova all'interno. Quando due oggetti hanno la stessa priorità, l'ordine in cui sono inseriti determina quale viene prima.

```
\relative c''' {
  \once \override TextScript.script-priority = #-100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```
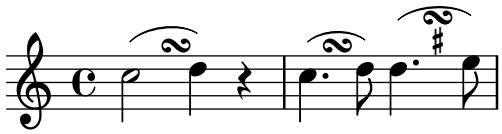


*Creare un gruppetto ritardato*

Creare un gruppetto ritardato, dove la nota più bassa del gruppetto usa l'alterazione, richiede vari \override. La proprietà `outside-staff-priority` deve essere impostata su `#f`, perché altrimenti questa avrebbe la precedenza sulla proprietà `avoid-slur`. Cambiando le frazioni `2/3` e `1/3` si aggiusta la posizione orizzontale.

```
\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
    { c4.( d8) }
    { s4 s\turn }
  >>
  \transpose c d \relative c'' <<
    { c4.( d8) }
    {
      s4
```

```
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion.outside-staff-priority = ##f
      \once \override AccidentalSuggestion.avoid-slur = #'inside
      \once \override AccidentalSuggestion.font-size = -3
      \once \override AccidentalSuggestion.script-priority = -1
      \single \hideNotes
      b8-\turn \noBeam
      s8
    }
  >>
}
```



## Vedi anche

Glossario Musicale: Sezione “tenuto” in *Glossario Musicale*, Sezione “accento” in *Glossario Musicale*, Sezione “staccato” in *Glossario Musicale*, Sezione “portato” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: [Scritte], pagina 239, Sezione 5.4.2 [Direzione e posizionamento], pagina 630, Sezione A.14 [Elenco delle articolazioni], pagina 764, [Trilli], pagina 150.

File installati: `scm/script.scm`.

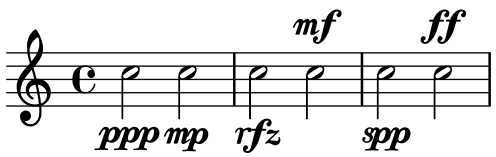
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Dinamiche

Le indicazioni dinamiche assolute si indicano con un comando che segue una nota, come ad esempio `c4\ff`. Le indicazioni dinamiche disponibili sono `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz` e `\rfz`. Le indicazioni dinamiche possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in dettaglio in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

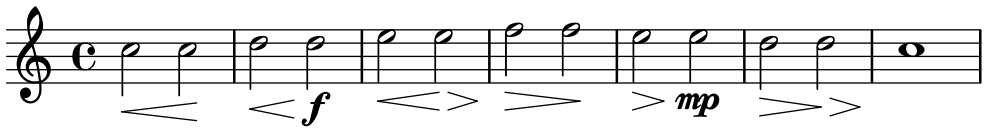
```
\relative c'' {
  c2\ppp c\mp
  c2\rfz c^\mf
  c2_\spp c^\ff
}
```




Un’indicazione di *crescendo* inizia con `\<` e termina con `\!`, un’indicazione dinamica assoluta o un’ulteriore indicazione di crescendo o decrescendo. Un’indicazione di *decrescendo* inizia con `\>` e termina nello stesso modo, ovvero con `\!`, un’indicazione dinamica assoluta oppure un altro segno di crescendo o decrescendo. Si possono usare `\cr` e `\decr` al posto di `\<` e `\>`; si possono

usare `\endcr` e `\enddecr` al posto di `\!` per terminare unindicazione di crescendo o decrescendo, rispettivamente. Le *forcelle* vengono create con questa notazione.

```
\relative c'' {
  c2\< c\!
  d2\< d\f
  e2\< e\>
  f2\> f\!
  e2\> e\mp
  d2\> d\>
  c1\!
}
```
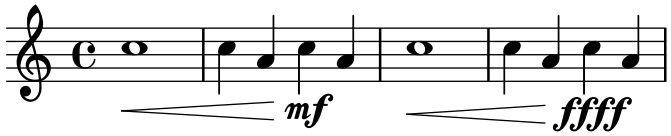



Una forcella che termina con `\!` si estenderà fino al margine destro della nota a cui è assegnato `\!`. Nel caso in cui sia terminata con l'inizio di un altro segno di *crescendo* o *decrescendo*, si estenderà fino al centro della nota a cui è assegnato il successivo `\<` o `\>`. La forcella successiva partirà dal margine destro della stessa nota invece che dal margine sinistro, come accade quando si termina con `\!`. Una forcella che termina sul battere si fermerà alla stanghetta precedente.

```
\relative {
  c''1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
}
```



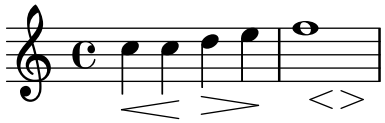
Le forcelle terminate con indicazioni dinamiche assolute invece che da `\!` avranno un aspetto simile. Tuttavia, la lunghezza dell'indicazione dinamica assoluta stessa può cambiare il punto in cui finisce la forcella precedente.

```
\relative {
  c''1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
}
```
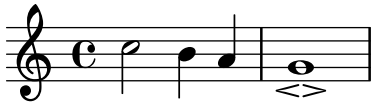



Occorre usare le pause spaziatrici per attaccare più di un'indicazione a una nota. Questo è utile soprattutto quando si aggiunge un *crescendo* e un *decrescendo* alla stessa nota:

```
\relative {
  c''4\< c\! d\> e\!
  << f1 { s4 s4\< s4\> s4\! } >>
}
```
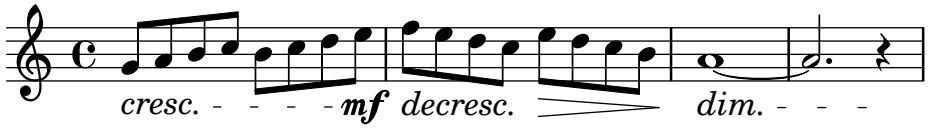


Il comando `\espressivo` permette di indicare un crescendo e un decrescendo sulla stessa nota. Tuttavia, si tenga presente che viene implementato come articolazione, non come dinamica.

```
\relative {
  c''2 b4 a
  g1\espressivo
}
```
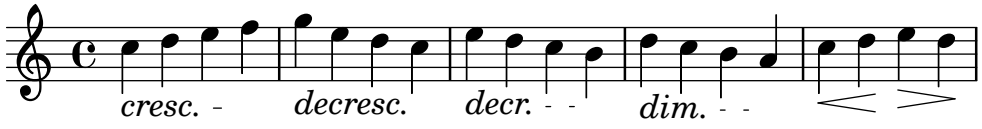


Le indicazioni di crescendo testuali iniziano con `\cresc`, quelle di decrescendo con `\decresc` o `\dim`. Le linee di estensione sono aggiunte automaticamente.

```
\relative {
  g'8\cresc a b c b c d e\mf |
  f8\decresc e d c e\> d c b |
  a1\dim ~ |
  a2. r4\! |
}
```




Le indicazioni testuali per i cambi di dinamica possono essere impiegate anche per sostituire le forcelle:

```
\relative c'' {
  \crescTextCresc
  c4\< d e f\! |
  \dimTextDecresc
  g4\> e d c\! |
  \dimTextDecr
  e4\> d c b\! |
  \dimTextDim
  d4\> c b a\! |
  \crescHairpin
  \dimHairpin
  c4\< d\! e\> d\! |
}
```



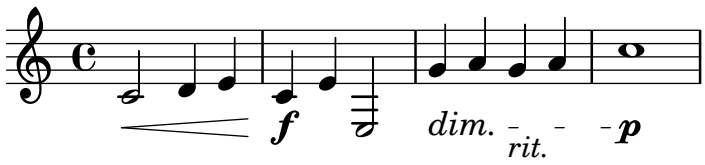

Per creare nuove indicazioni dinamiche assolute o testi da allineare alle dinamiche, si veda [Nuove indicazioni dinamiche], pagina 133.

Il posizionamento verticale della dinamica è gestito da Sezione “DynamicLineSpanner” in *Guida al Funzionamento Interno*.

Esiste un contesto `Dynamics` che permette di posizionare le indicazioni dinamiche su un’apposita linea orizzontale. Si usano le pause spaziatrici per indicarne la collocazione temporale (le note in un contesto `Dynamics` occupano infatti il rispettivo valore musicale, ma senza

comparire sul rigo). Il contesto `Dynamics` può contenere altri elementi utili come indicazioni testuali, estensori del testo e indicazioni di pedalizzazione del pianoforte.

```
<<
  \new Staff \relative {
    c'2 d4 e |
    c4 e e,2 |
    g'4 a g a |
    c1 |
  }
  \new Dynamics {
    s1\< |
    s1\f |
    s2\dim s2-"rit." |
    s1\p |
  }
>>
```




## Comandi predefiniti

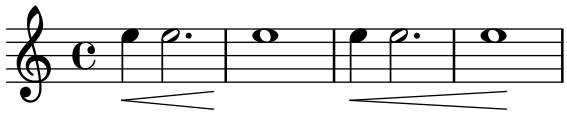
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

## Frammenti di codice selezionati

*Impostare il comportamento delle forcelle sulle stanghette*

Se la nota che termina una forcella si trova sul primo battito di una battuta, la forcella si ferma prima della stanghetta che precede la nota. Si può controllare questo comportamento modificando la proprietà `'to-barline`.

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

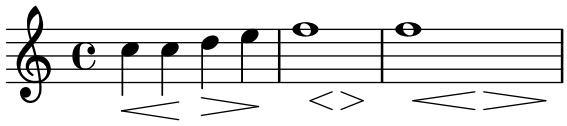


*Impostare la lunghezza minima delle forcelle*

Se le forcelle sono troppo corte, possono essere allungate modificando la proprietà `minimum-length` dell'oggetto `Hairpin`.

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

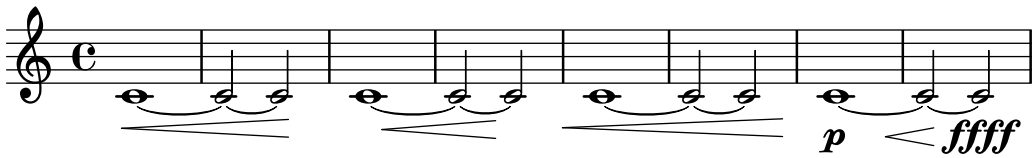
```
  \override Hairpin.minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```



*Spostare le estremità delle forcelle*

Le estremità delle forcelle possono essere spostate in modo relativo alla loro posizione predefinita (offset) impostando la proprietà `shorten-pair` dell'oggetto `Hairpin`. Valori positivi spostano le estremità a destra, valori negativi le spostano a sinistra. Diversamente dalla proprietà `minimum-length`, questa proprietà modifica solo l'aspetto della forcella; non cambia la spaziatura orizzontale (inclusa la posizione delle dinamiche confinanti). Questo metodo è quindi utile per ritoccare una forcella entro lo spazio ad essa allocato.

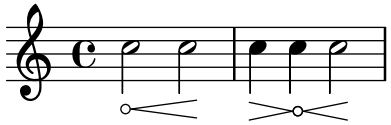
```
{
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(2 . 2)
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(-2 . -2)
  c'1~\<
  c'2~ c'\!
  c'1~\p-\tweak shorten-pair #'(2 . 0)\<
  c'2~ c'\ffff
}
```




*Forcelle con notazione al niente*

Le forcelle di dinamica possono essere rappresentate con ua punta tonda (notazione "al niente") impostando la proprietà `circled-tip` dell'oggetto `Hairpin` su `#t`.

```
\relative c'' {
  \override Hairpin.circled-tip = ##t
  c2\< c\!
  c4\> c\< c2\!
}
```

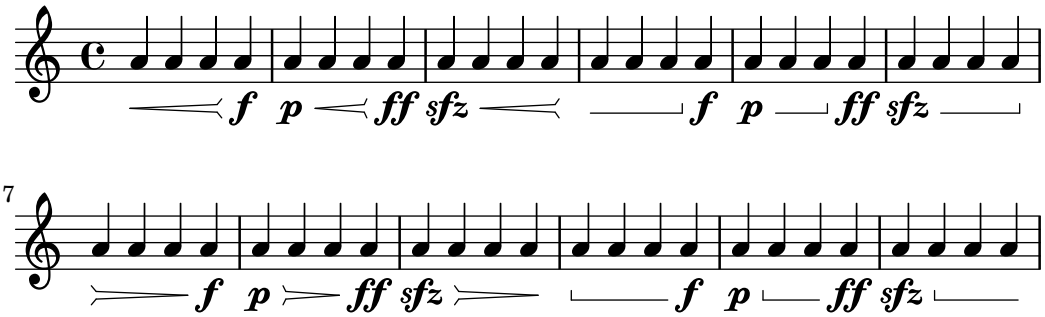


*Stampare le forcelle in vari stili*

Il segno di dinamica della forcella può avere diversi stili

```
\relative c'' {
  \override Hairpin.stencil = #flared-hairpin
```

```
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```

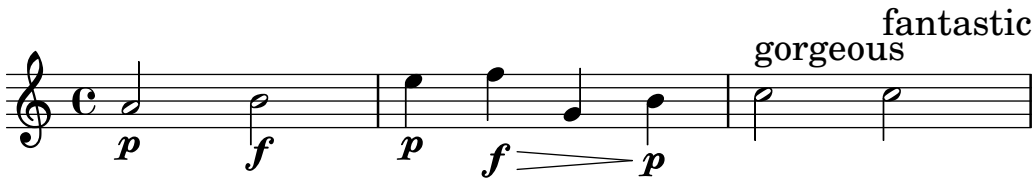


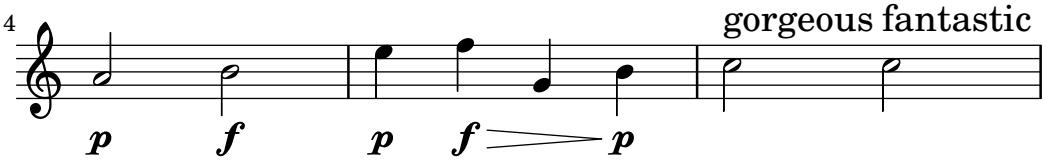

*Dinamiche e segni testuali allineati verticalmente*

Tutti gli oggetti `DynamicLineSpanner` (forcelle e testi di dinamica) sono posti a una distanza minima dal rigo determinata da `'staff-padding`. Se si imposta `'staff-padding` su un valore abbastanza grande, le dinamiche saranno allineate.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = #3
  \textLengthOn
  \override TextScript.staff-padding = #1
  \music
}
```

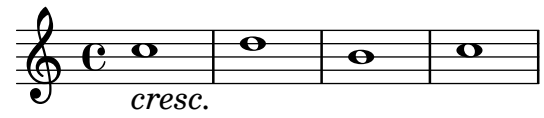





*Nascondere la linea di estensione per le dinamiche testuali*

I cambi di dinamica in stile testuale (come *cresc.* e *dim.*) appaiono con una linea tratteggiata che mostra la loro estensione. Questa linea può essere soppressa nel modo seguente:

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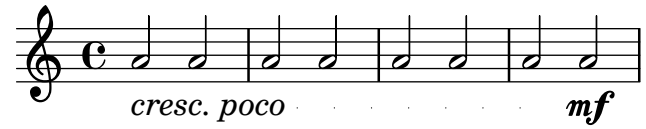



*Nascondere la linea di estensione per le dinamiche testuali*

Il testo usato per i *crescendo* e i *decrescendo* può essere cambiato modificando le proprietà di contesto `crescendoText` e `decrescendoText`.

Lo stile della linea dell'estensore può essere cambiato modificando la proprietà `'style` di `DynamicTextSpanner`. Il valore predefinito è `'dashed-line`; gli altri valori possibili sono `'line`, `'dotted-line` e `'none`.

```
\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
}
```




## Vedi anche

Glossario Musicale: Sezione "al niente" in *Glossario Musicale*, Sezione "crescendo" in *Glossario Musicale*, Sezione "decrescendo" in *Glossario Musicale*, Sezione "forcella" in *Glossario Musicale*. Manuale di apprendimento: Sezione "Articolazioni e dinamiche" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Nuove indicazioni dinamiche], pagina 133, Sezione 3.5.9 [Miglioramento dell'output MIDI], pagina 536, Sezione 3.5.4 [Gestione delle dinamiche nel MIDI], pagina 527.

Frammenti: Sezione "Expressive marks" in *Frammenti di codice*.
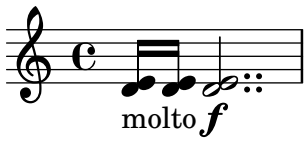
Guida al funzionamento interno: Sezione "DynamicText" in *Guida al Funzionamento Interno*, Sezione "Hairpin" in *Guida al Funzionamento Interno*, Sezione "DynamicLineSpanner" in *Guida al Funzionamento Interno*, Sezione "Dynamics" in *Guida al Funzionamento Interno*.

## Nuove indicazioni dinamiche

Il modo più semplice per creare indicazioni dinamiche è usare gli oggetti `\markup`.

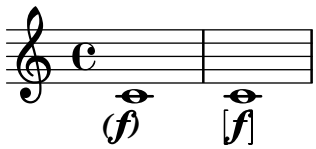
```
moltoF = \markup { molto \dynamic f }

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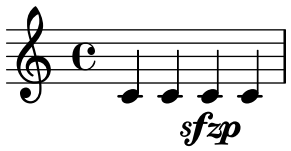



In modalità markup, si possono creare dinamiche editoriali (racchiuse tra parentesi normali o quadrate). La sintassi della modalità markup è descritta in Sezione 1.8.2 [Formattazione del testo], pagina 246.

```
roundF = \markup {
    \center-align \concat { \bold { \italic ( }
           \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative {
  c'1_\roundF
  c1_\boxF
}
```




È possibile creare semplicemente indicazioni dinamiche centrate verticalmente con la funzione `make-dynamic-script`.

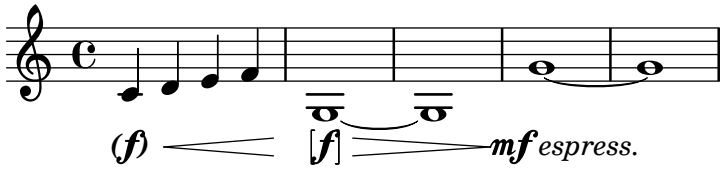
```
sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfzp c
}
```




In generale, `make-dynamic-script` assume come argomento qualsiasi oggetto markup. Il tipo di carattere per la dinamica contiene solo i caratteri `f,m,p,r,s` e `z`; dunque, se si desidera creare un'indicazione dinamica che contenga testo semplice e simboli di punteggiatura, occorre usare dei comandi markup che ripristinino la famiglia e la codifica del tipo di carattere per il testo normale, ad esempio `\normal-text`. Il vantaggio nell'uso di `make-dynamic-script` al posto di un normale markup è l'allineamento verticale degli oggetti markup e delle forcelle collegate alla stessa testa di nota.

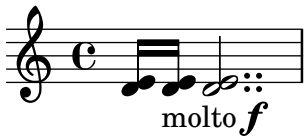
```
roundF = \markup { \center-align \concat {
```

```
            \normal-text { \bold { \italic ( } }
            \dynamic f
            \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
               \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative {
  c'4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g1
  g'1~\mfEspressDynamic
  g1
}
```
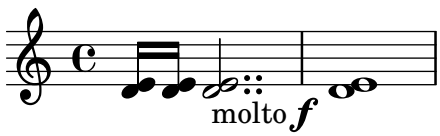



Si può usare anche la forma Scheme della modalità markup. La sintassi è spiegata in Sezione "Markup construction in Scheme" in *Estendere*.

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




Per allineare a sinistra il testo di dinamica invece di centrarlo su una nota, si usa un `\tweak`:

```
moltoF = \tweak DynamicText.self-alignment-X #LEFT
         #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF <d e>1
}
```



Le impostazioni dei tipi di carattere in modalità markup sono descritti in [Scelta del tipo di carattere e della dimensione], pagina 248.

### Vedi anche

Guida alla notazione: Sezione 1.8.2 [Formattazione del testo], pagina 246, [Scelta del tipo di carattere e della dimensione], pagina 248, Sezione 3.5.9 [Miglioramento dell'output MIDI], pagina 536, Sezione 3.5.4 [Gestione delle dinamiche nel MIDI], pagina 527.

Extending LilyPond: Sezione "Markup construction in Scheme" in *Estendere*.

Frammenti: Sezione "Expressive marks" in *Frammenti di codice*.

## 1.3.2 Indicazioni espressive curvilinee
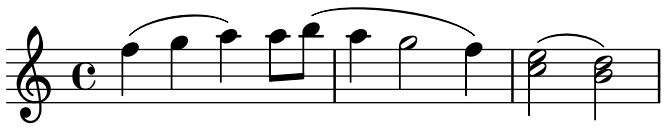
Questa sezione spiega come creare varie indicazioni espressive con forma curvilinea: legature di portamento, legature di frase, respiri, portamenti indeterminati discendenti (cadute) o ascendenti.

### Legature di portamento

Le *legature di portamento* si inseriscono con delle parentesi:

> **Nota:** Nella musica polifonica, una legatura di portamento deve terminare nella stessa voce in cui è iniziata.

```
\relative {
  f''4( g a) a8 b(
  a4 g2 f4)
  <c e>2( <b d>2)
}
```



Le legature di portamento possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Tracciare due legature di portamento simultanee o sovrapposte richiede una particolare attenzione. Di solito le legature di portamento più esterne indicano in realtà una legatura di frase e le legature di frase possono essere sovrapposte a una normale legatura, vedi [Legature di frase], pagina 138. Quando invece si vogliono usare molteplici legature di portamento normali in una sola voce, l'inizio e la fine di ogni legatura devono essere preceduti da un `\=` seguito da una chiave identificativa (un simbolo o un numero intero non negativo).
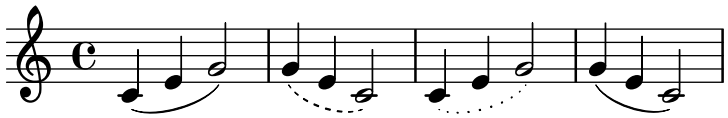
```
\fixed c' {
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >
}
```



Le legature di portamento possono essere continue, punteggiate o tratteggiate. Lo stile predefinito è quello continuo:
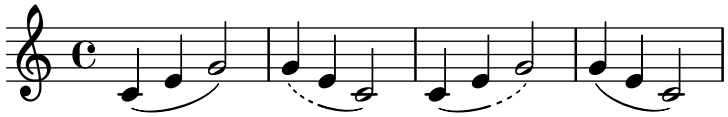
```
\relative {
```

```
  c'4( e g2)
  \slurDashed
  g4( e c2)
  \slurDotted
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```
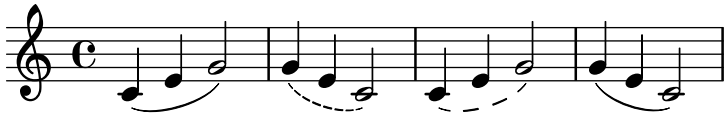


Le legature di portamento possono essere anche semitratteggiate (half-dashed), ovvero con la prima metà tratteggiata e la seconda continua; oppure semicontinue (half-solid), ovvero con la prima metà continua e la seconda tratteggiata:

```
\relative {
  c'4( e g2)
  \slurHalfDashed
  g4( e c2)
  \slurHalfSolid
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Si possono definire modelli di tratteggio personalizzati per le legature di portamento:

```
\relative {
  c'4( e g2)
  \slurDashPattern #0.7 #0.75
  g4( e c2)
  \slurDashPattern #0.5 #2.0
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



## Comandi predefiniti

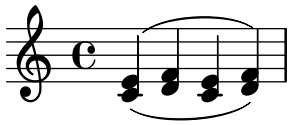
`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

## Frammenti di codice selezionati

*Uso delle doppie legature di portamento per gli accordi legati*

Alcuni compositori scrivono due *legature di portamento* per indicare gli accordi legati. Si può ottenere questo risultato impostando `doubleSlurs`.
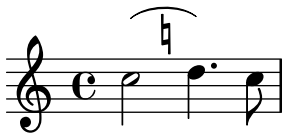
```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```



*Posizionare il testo a margine dentro le legature di portamento*

I testi a margine devono avere la proprietà `outside-staff-priority` impostata su false per poter apparire dentro le legature di portamento.

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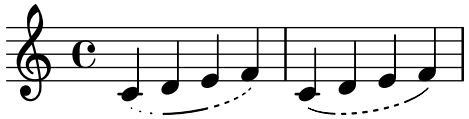


*Legature di portamento con complesse strutture di tratteggio*

Le legature di portamento possono avere schemi di tratteggio complessi definendo la proprietà `dash-definition`. `dash-definition` è una lista di `dash-elements`. Un `dash-element` è una lista di parametri che definiscono il comportamento del tratteggio per un segmento della legatura.

La legatura di portamento è definita come il parametro t della curva di bezier che va da 0 sul margine sinistro della legatura fino a 1 su quello destro. `dash-element` è una lista di `(inizio-t fine-t frazione-trattino punto-trattino)`. La regione della legatura di portamento che va da `inizio-t` a `fine-t` avrà una frazione `frazione-trattino` di ogni `punto-trattino` nero. `punto-trattino` viene definito in spazi rigo. `frazione-trattino` è impostato su 1 per una legatura di portamento continua.

```
\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)
}
```



## Vedi anche

Glossario Musicale: Sezione “legatura di portamento” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Legature di frase], pagina 138.
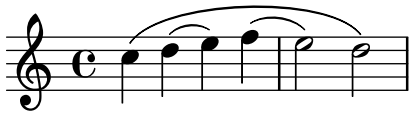
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Slur” in *Guida al Funzionamento Interno*.

## Legature di frase

Le *legature di frase*, che indicano una frase musicale, si scrivono con i comandi `\(` e `\)`:

```
\relative {
  c''4\( d( e) f(
  e2) d\)
}
```
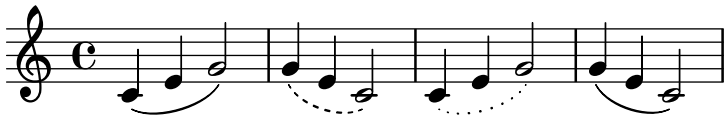


A livello tipografico, una legatura di frase si comporta in modo pressoché identico a una normale legatura di portamento. Sono però trattate come oggetti diversi; ad esempio, `\slurUp` non ha effetto su una legatura di frase. Le legature di frase possono essere posizionate sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Per inserire più legature di frase simultanee o sovrapposte si usa `\=`, come per le normali legature di portamento (vedi [Legature di portamento], pagina 135).

Le legature di frase possono essere continue, puntate o tratteggiate. Lo stile predefinito è quello continuo:
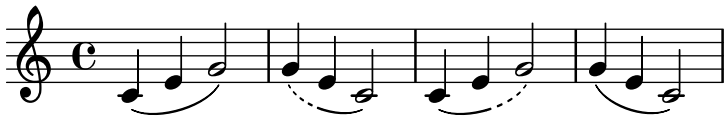
```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashed
  g4\( e c2\)
  \phrasingSlurDotted
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Le legature di frase possono essere anche semitratteggiate (la prima metà tratteggiata, la seconda continua) o semicontinue (la prima metà continua, la seconda tratteggiata):
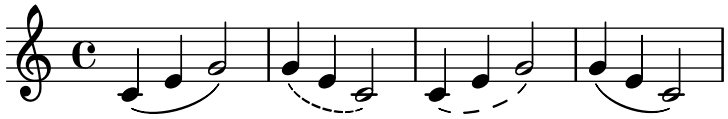
```
\relative {
  c'4\( e g2\)
  \phrasingSlurHalfDashed
```

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



Si possono definire modelli di tratteggio personalizzati anche per le legature di frase:

```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashPattern #0.7 #0.75
  g4\( e c2\)
  \phrasingSlurDashPattern #0.5 #2.0
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Le definizioni dei modelli di tratteggio per le legature di frase hanno la stessa struttura di quelle per le legature di portamento. Per maggiori informazioni sui modelli complessi di tratteggio si consultino i frammenti in [Legature di portamento], pagina 135.

### Comandi predefiniti

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`, `\phrasingSlurDashed`, `\phrasingSlurDotted`, `\phrasingSlurHalfDashed`, `\phrasingSlurHalfSolid`, `\phrasingSlurDashPattern`, `\phrasingSlurSolid`.

### Vedi anche

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Legature di portamento], pagina 135.
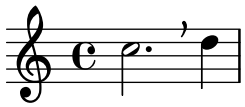
Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “PhrasingSlur” in *Guida al Funzionamento Interno*.

## Respiri

I respiri si inseriscono col comando `\breathe`:
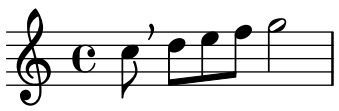
```
{ c''2. \breathe d''4 }
```



Diversamente da altri segni di espressione, il respiro non è associato alla nota precedente ma è un evento musicale separato. Dunque tutti i segni espressivi attaccati alla nota precedente, tutte le parentesi quadre che indicano le travature manuali e le parentesi che indicano le legature di portamento e di frase devono essere poste prima di `\breathe`.

Un respiro termina una travatura automatica; per evitare questo comportamento, si veda [Travature manuali], pagina 97.

```
\relative { c''8 \breathe d e f g2 }
```



È supportata la divisio, indicatore del respiro nella musica antica. Maggiori dettagli in [Divisiones], pagina 454.

## Frammenti di codice selezionati

*Cambiare il simbolo del segno di respiro*

Il glifo del respiro può essere modificato sovrascrivendo la proprietà `text` dell'oggetto di formattazione `BreathingSign` con qualsiasi testo incluso in un blocco markup.
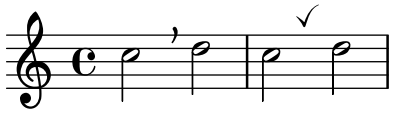
```
\relative c'' {
  c2
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.rvarcomma" }
  \breathe
  d2
}
```



*Usare un segno di spunta come simbolo di respiro*

La musica vocale e per fiati usa frequentemente il segno di spunta come segno di respiro. Questo indica un respiro che sottrae un po' di tempo alla nota precedente invece di prendere una piccola pausa, indicata dal segno di respiro rappresentato dalla virgola. Il segno può essere spostato un po' su per allontanarlo dal rigo.

```
\relative c'' {
  c2
  \breathe
  d2
  \override BreathingSign.Y-offset = #2.6
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.tickmark" }
  c2
  \breathe
  d2
}
```
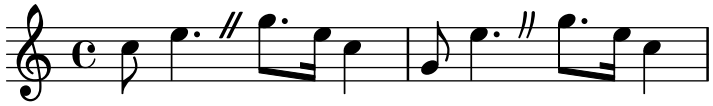


*Inserire una cesura*

I segni di cesura possono essere creati sovrascrivendo la proprietà `'text` dell'oggetto `BreathingSign`. È disponibile anche un segno di cesura curvo.

```
\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
}
```



## Vedi anche

Glossario Musicale: Sezione "cesura" in *Glossario Musicale*.

Guida alla notazione: [Divisiones], pagina 454.

Frammenti: Sezione "Expressive marks" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "BreathingEvent" in *Guida al Funzionamento Interno*, Sezione "BreathingSign" in *Guida al Funzionamento Interno*, Sezione "Breathing_sign_engraver" in *Guida al Funzionamento Interno*.

## Portamenti indeterminati discendenti (cadute) e ascendenti

I *portamenti indeterminati verso il basso (cadute) e verso l'alto* possono essere aggiunti alle note col comando `\bendAfter`. La direzione del portamento è indicata con un più o un meno (su o giù). Il numero indica l'intervallo per cui il portamento si estenderà *oltre* la nota principale.

```
\relative c'' {
  c2\bendAfter #+4
  c2\bendAfter #-4
  c2\bendAfter #+6.5
  c2\bendAfter #-6.5
  c2\bendAfter #+8
  c2\bendAfter #-8
}
```



## Frammenti di codice selezionati

*Cambiare la forma dei portamenti indeterminati verso il basso o verso l'alto*

La proprietà `shortest-duration-space` può essere modificata per cambiare la forma dei portamenti indeterminati verso il basso o verso l'alto.

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

```
  \override Score.SpacingSpanner.shortest-duration-space = #4.0
  c2-\bendAfter #5
  c2-\bendAfter #-4.75
  c2-\bendAfter #8.5
  c2-\bendAfter #-6
}
```



## Vedi anche

Glossario Musicale: Sezione “portamento indeterminato verso il basso” in *Glossario Musicale*, Sezione “portamento indeterminato verso l’alto” in *Glossario Musicale*.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

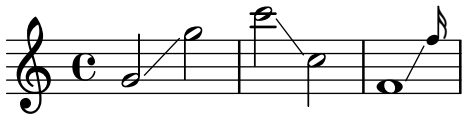
### 1.3.3 Indicazioni espressive lineari

Questa sezione spiega come creare varie indicazioni espressive che seguono una traiettoria lineare: glissandi, arpeggi e trilli.

## Glissando

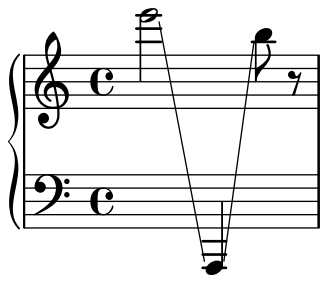
Un *glissando* si crea attaccando `\glissando` a una nota:

```
\relative {
  g'2\glissando g'
  c2\glissando c,
  \afterGrace f,1\glissando f'16
}
```



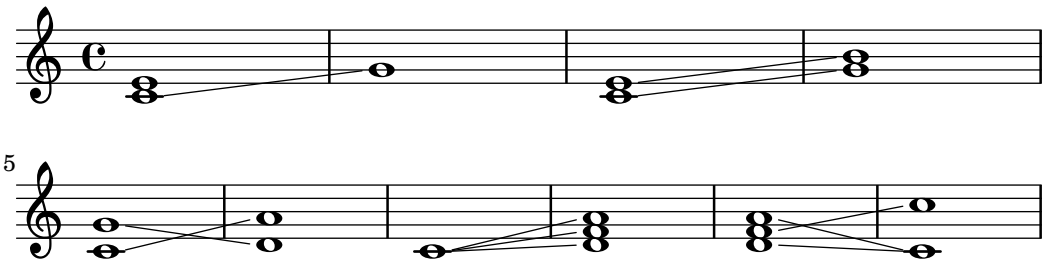
Un glissando può collegare note appartenenti a righi diversi:

```
\new PianoStaff <<
  \new Staff = "right" {
    e'''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>
```



Un glissando può collegare le note negli accordi. Se è necessario qualcosa di diverso dal normale abbinamento uno a uno delle note, si possono definire le connessioni tra le note attraverso `\glissandoMap`, dove le note di un accordo sono numerate a partire da zero nell'ordine in cui appaiono nel file di input `.ly`.

```
\relative {
  <c' e>1\glissando g' |
  <c, e>1\glissando |
  <g' b> |
  \break
  \set glissandoMap = #'((0 . 1) (1 . 0))
  <c, g'>1\glissando |
  <d a'> |
  \set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
  c1\glissando |
  <d f a> |
  \set glissandoMap = #'((2 . 0) (1 . 0) (0 . 1))
  <f d a'>1\glissando |
  <c c'> |
}
```




Si possono adottare diversi stili di glissando. Maggiori dettagli in Sezione 5.4.8 [Stili della linea], pagina 644.
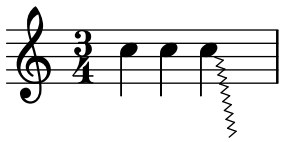
## Frammenti di codice selezionati

*Glissando contemporaneo*

Un glissando contemporaneo senza una nota finale può essere creato usando una nota nascosta e un tempo di cadenza.

```
\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
```

```
  \bar "|"
}
```



*Aggiungere i segni di tempo per i glissandi lunghi*

I battiti saltati nei glissandi molto lunghi vengono talvolta segnalati con delle indicazioni di tempo, che consistono solitamente in dei gambi privi di teste di nota. Questi gambi possono essere usati anche per contenere segni di espressione intermedi.

Se i gambi non si allineano bene al glissando, può essere necessario riposizionarli leggermente.
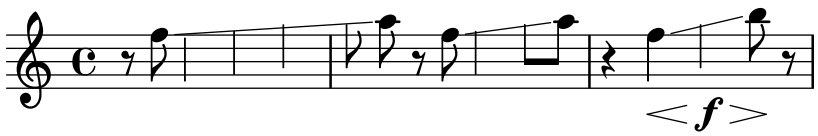
```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

glissandoSkipOff = {
  \revert NoteColumn.glissando-skip
  \undo \hide NoteHead
  \revert NoteHead.no-ledgers
}

\relative c'' {
  r8 f8\glissando
  \glissandoSkipOn
  f4 g a a8\noBeam
  \glissandoSkipOff
  a8

  r8 f8\glissando
  \glissandoSkipOn
  g4 a8
  \glissandoSkipOff
  a8 |

  r4 f\glissando \<
  \glissandoSkipOn
  a4\f \>
  \glissandoSkipOff
  b8\! r |
}
```

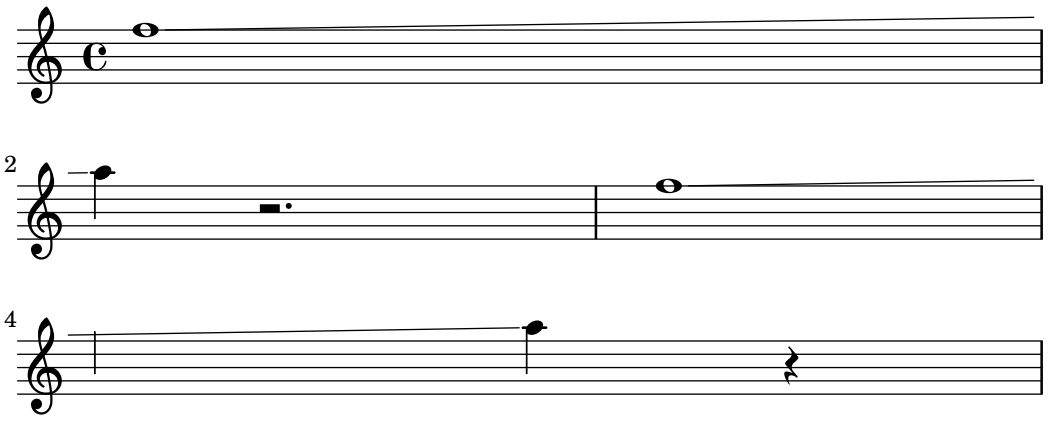


*Lasciare che i glissandi vadano a capo*

Per permettere a un glissando di andare a capo se capita su un'interruzione di riga, si impostano le proprietà `breakable` e `after-line-breaking` su `#t`:

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

\relative c'' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
  a4 r2. |
  f1\glissando
  \once \glissandoSkipOn
  \break
  a2 a4 r4 |
}
```



*Estendere i glissandi sulle volte delle ripetizioni*

Un glissando che si estende in vari blocchi `\alternative` può essere simulato aggiungendo all'inizio di ogni blocco `\alternative` una nota di abbellimento nascosta da cui inizia un glissando. La nota di abbellimento deve avere la stessa altezza della nota da cui parte il glissando iniziale. In questo frammento si usa una funzione musicale che prende come argomento l'altezza della nota di abbellimento.

Attenzione: nella musica polifonica la nota di abbellimento deve avere una nota di abbellimento corrispondente in tutte le altre voci.

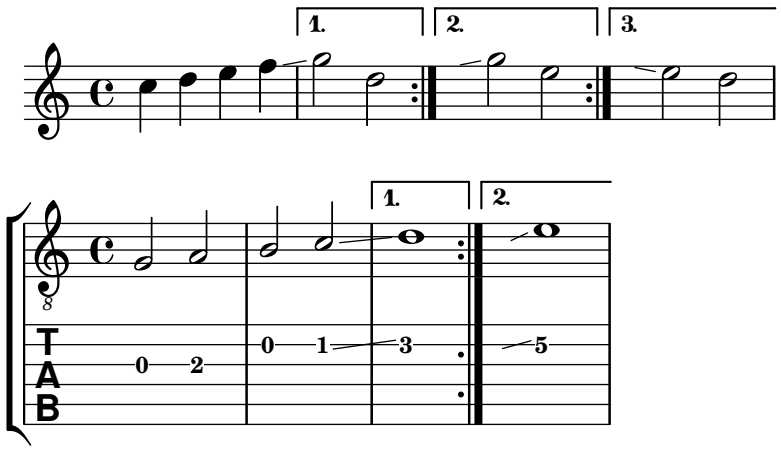
```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
```

```
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music =  \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \new Voice { \clef "G_8" \music }
    >>
    \new TabStaff  <<
      \new TabVoice { \clef "moderntab" \music }
    >>
  >>
}
```




## Vedi anche

Glossario Musicale: Sezione “glissando” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.8 [Stili della linea], pagina 644.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Glissando” in *Guida al Funzionamento Interno*.
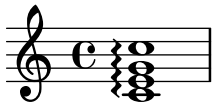
### Problemi noti e avvertimenti

Non è supportato il testo lungo la linea del glissando (ad esempio *gliss.*).

## Arpeggio

Un *arpeggio* su un accordo (detto anche accordo spezzato) si ottiene aggiungendo `\arpeggio` all’accordo:

```
\relative { <c' e g c>1\arpeggio }
```



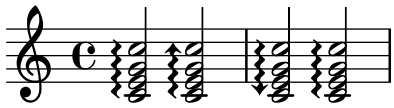
Si possono scrivere vari tipi di arpeggio. `\arpeggioNormal` ripristina l’arpeggio normale:

```
\relative {
  <c' e g c>2\arpeggio

  \arpeggioArrowUp
  <c e g c>2\arpeggio

  \arpeggioArrowDown
  <c e g c>2\arpeggio

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



Si possono creare simboli di arpeggio speciali *in forma di parentesi*:
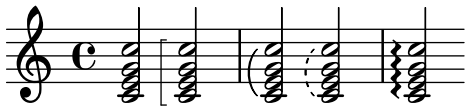
```
\relative {
  <c' e g c>2

  \arpeggioBracket
  <c e g c>2\arpeggio

  \arpeggioParenthesis
  <c e g c>2\arpeggio

  \arpeggioParenthesisDashed
  <c e g c>2\arpeggio

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



Le proprietà del tratteggio della parentesi dell'arpeggio sono regolate dalla proprietà `'dash-definition`, descritta in [Legature di portamento], pagina 135.

Gli arpeggi possono essere scritti in modo esplicito con le legature di valore. Per maggiori dettagli si veda [Legature di valore], pagina 55.

## Comandi predefiniti

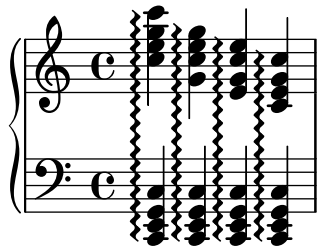
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis`, `\arpeggioParenthesisDashed`.

## Frammenti di codice selezionati

*Creare degli arpeggi che attraversano il rigo del pianoforte*

In un rigo per pianoforte (`PianoStaff`), è possibile far sì che un *arpeggio* attraversi i righi impostando la proprietà `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
    }
  }
>>
```

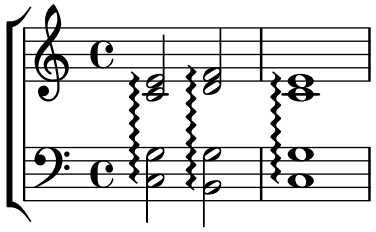


*Creare degli arpeggi che attraversano i righi in altri contesti*

Si possono creare *arpeggi* che attraversano i righi in contesti diversi da `GrandStaff`, `PianoStaff` e `StaffGroup` se l'incisore `Span_arpeggio_engraver` è incluso nel contesto `Score`.

```
\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
```
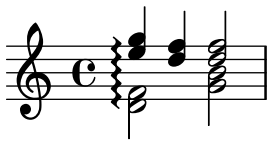
```
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
      \consists "Span_arpeggio_engraver"
    }
  }
}
```



*Creare degli arpeggi che attraversano note appartenenti a voci diverse*

Si può disegnare un *arpeggio* che attraversa delle note in voci diverse dello stesso rigo se si aggiunge l'incisore `Span_arpeggio_engraver` nel contesto `Staff`:

```
\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 }
  >>
}
```



## Vedi anche

Glossario Musicale: Sezione "arpeggio" in *Glossario Musicale*.

Guida alla notazione: [Legature di portamento], pagina 135, [Legature di valore], pagina 55.

Frammenti: Sezione "Expressive marks" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Arpeggio" in *Guida al Funzionamento Interno*, Sezione "Slur" in *Guida al Funzionamento Interno*, Sezione "PianoStaff" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non è possibile mostrare simultaneamente arpeggi connessi e non connessi in un `PianoStaff`.

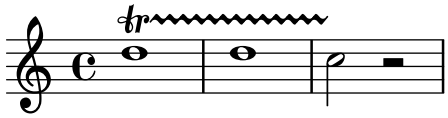
L'arpeggio in forma di parentesi non può essere impostato con facilità negli arpeggi che attraversano i righi; occorre ricorrere a metodi più complessi descritti in [Gambi trasversali ai righi], pagina 338.

## Trilli

I trilli senza linea di estensione si ottengono col comando `\trill`; si veda [Articolazioni e abbellimenti], pagina 124.
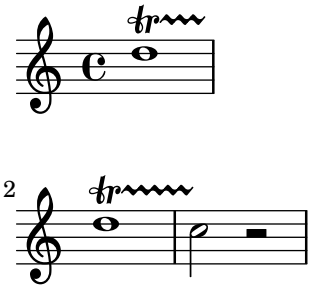
I trilli con linea di estensione si ottengono con `\startTrillSpan` e `\stopTrillSpan`:

```
\relative {
  d''1\startTrillSpan
  d1
  c2\stopTrillSpan
  r2
}
```



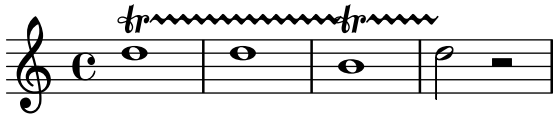
Un estensore del trillo che va a capo ricomincerà esattamente sopra la prima nota della nuova riga.

```
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan
  r2
}
```
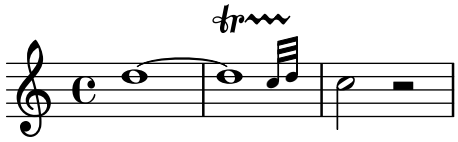



È possibile tracciare trilli consecutivi senza dover esplicitare i comandi `\stopTrillSpan`, perché il trillo successivo diventerà automaticamente il limite destro di quello precedente.

```
\relative {
  d''1\startTrillSpan
  d1
  b1\startTrillSpan
  d2\stopTrillSpan
  r2
}
```
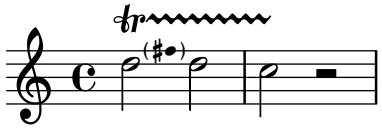


I trilli possono essere anche combinati con le note di abbellimento. La sintassi di questo costrutto e il metodo per posizionare in modo preciso gli abbellimenti sono descritti in [Abbellimenti], pagina 116.

```
\relative {
  d''1~\afterGrace
  d1\startTrillSpan { c32[ d]\stopTrillSpan }
  c2 r2
}
```



I trilli che richiedono una nota ausiliaria dall'altezza esplicita si ottengono col comando `\pitchedTrill`. Il primo argomento è la nota principale e il secondo è la nota *trillata*, che appare come una testa di nota senza gambo e racchiusa tra parentesi.

```
\relative {
  \pitchedTrill
  d''2\startTrillSpan fis
  d2
  c2\stopTrillSpan
  r2
}
```



L'alterazione del primo trillo con notina in una misura viene sempre visualizzata, anche per i bequadri.
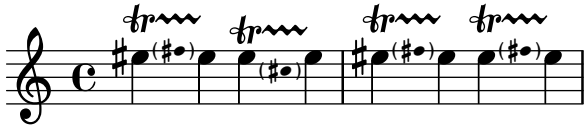
```
{
  \key d \major
  \pitchedTrill
  d'2\startTrillSpan cis d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan c d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan e d\stopTrillSpan
}
```



Alterazioni successive (della stessa nota nella stessa misura) devono essere aggiunte manualmente.

```
\relative {
  \pitchedTrill
```

```
  eis''4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan cis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis!
  eis4\stopTrillSpan
}
```



### Comandi predefiniti

`\startTrillSpan`, `\stopTrillSpan`.

### Vedi anche
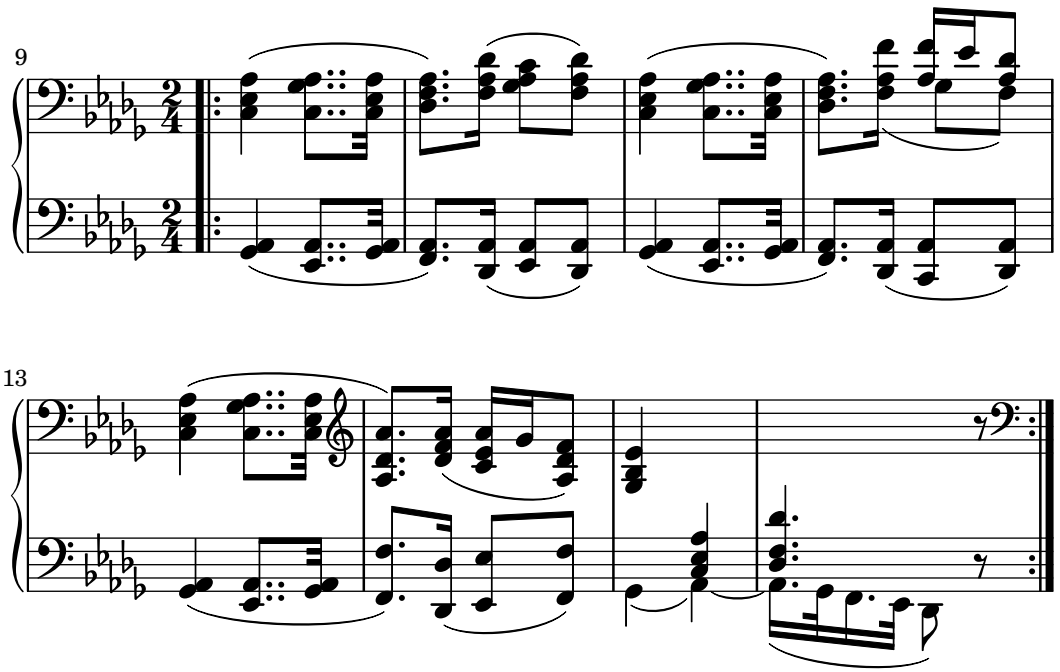
Glossario Musicale: Sezione “trillo” in *Glossario Musicale*.

Guida alla notazione: [Articolazioni e abbellimenti], pagina 124, [Abbellimenti], pagina 116.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TrillSpanner” in *Guida al Funzionamento Interno*.

## 1.4 Ripetizioni



La ripetizione è un concetto chiave in musica e può essere resa con varie forme di notazione. LilyPond supporta i seguenti tipi di ripetizioni:

`volta` La musica ripetuta non viene scritta per intero ma racchiusa tra barre di ripetizione. Se la ripetizione si trova all'inizio di un brano, la stanghetta di ritornello è posta soltanto alla fine della ripetizione. I finali alternativi (volte) appaiono da sinistra a destra e sono evidenziati da delle parentesi. Questa è la notazione standard per le ripetizioni con finali alternativi.

`unfold` La musica ripetuta viene scritta per intero, tante volte quante sono specificate dal *`numero-ripetizioni`*. È utile quando si scrive musica ripetitiva.

`percent` Si tratta di ripetizioni del singolo tempo (battito) o della battuta. Hanno l'aspetto di una barra obliqua o di segni di percentuale.

`tremolo` Si usa per scrivere travature a tremolo.

### 1.4.1 Ripetizioni lunghe

Questa sezione spiega come inserire ripetizioni lunghe (solitamente di più battute). Tali ripetizioni possono essere in due forme: racchiuse tra segni di ritornello oppure ricopiate interamente (adatte a scrivere musica ripetitiva). Si possono anche controllare manualmente i segni di ripetizione.

#### Ripetizioni normali
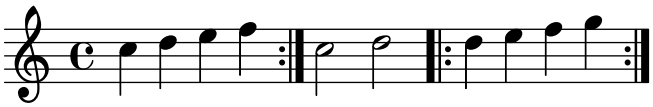
La sintassi per una normale ripetizione è

```
\repeat volta numero-di-ripetizioni espressione-musicale
```

dove *`espressione-musicale`* è la musica da ripetere.
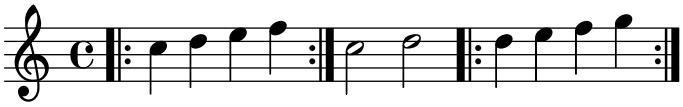
Un'unica ripetizione senza finale alternativo:

```
\relative {
  \repeat volta 2 { c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



Il segno di inizio della ripetizione, per impostazione predefinita, non appare nella prima misura. È tuttavia possibile aggiungerlo inserendo a mano la battuta `\bar ".|:"` prima della prima nota.

```
\relative {
  \repeat volta 2 { \bar ".|:" c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



I finali alternativi si ottengono con `\alternative`. Ogni gruppo di alternative deve essere a sua volta racchiuso tra parentesi.

```
\repeat volta numero-di-ripetizioni espressione-musicale
```
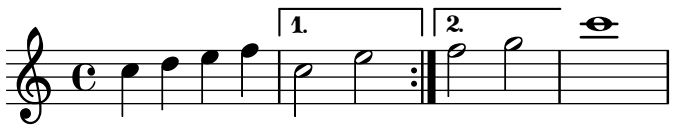
```
\alternative {
  { espressione-musicale }
}
```

dove *espressione-musicale* è la musica.

Se il numero di ripetizioni è superiore a quello dei finali alternativi, alle prime ripetizioni viene assegnata la prima alternativa.
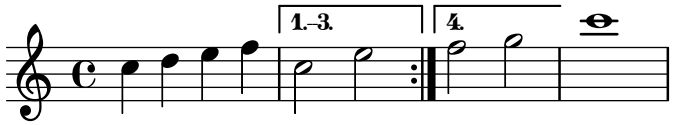
Una singola ripetizione con un finale alternativo:

```
\relative {
  \repeat volta 2 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```



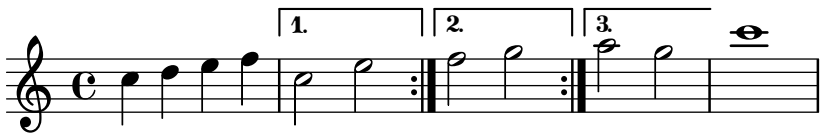

Molteplici ripetizioni con un finale alternativo:

```
\relative {
  \repeat volta 4 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```




Molteplici ripetizioni con più di un finale alternativo:

```
\relative {
  \repeat volta 3 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
    { a2 g | }
  }
  c1
}
```
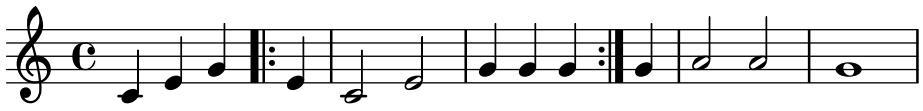


**Nota:** Se ci sono due o più finali alternativi, non ci deve essere niente tra la parentesi di chiusura di uno e quella di apertura di quello successivo all'interno di un blocco `\alternative`, altrimenti non si otterrà il numero atteso di finali.

**Nota:** Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione "Appare un rigo in più" in *Uso del Programma*.
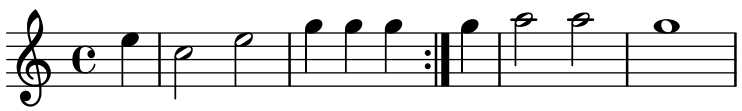
Se una ripetizione che non ha finali alternativi inizia in mezzo a una misura, solitamente termina in un punto corrispondente nel mezzo di una misura successiva (così che tra le due estremità ci sia una misura completa). In questo caso i segni di ripetizione non sono delle "vere" e proprie stanghette, dunque né i controlli di battuta né i comandi `\partial` devono essere messi lì:

```
\relative {
  c'4 e g
  \repeat volta 4 {
    e4 |
    c2 e |
    g4 g g
  }
  g4 |
  a2 a |
  g1 |
}
```



Se una ripetizione senza finali alternativi inizia con una misura parziale, si applicano gli stessi principi dell'esempio precedente, a parte il fatto che è richiesto un comando `\partial` all'inizio della misura:

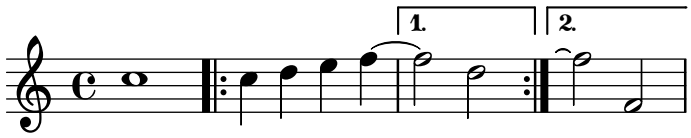
```
\partial 4
\repeat volta 4 {
  e'4 |
  c2 e |
  g4 g g
}
g4 |
a2 a |
g1 |
```



Si possono aggiungere delle legature di valore a un secondo finale:

```
\relative {
  c''1
  \repeat volta 2 { c4 d e f~ }
  \alternative {
```
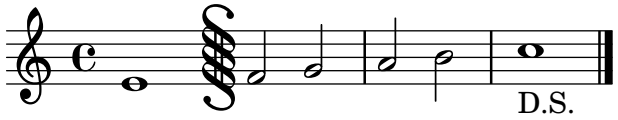
```
    { f2 d }
    { f2\repeatTie f, }
  }
}
```




Il comando `\inStaffSegno` può essere usato per generare una stanghetta composita che incorpora il simbolo di segno nella stanghetta di ripetizione appropriata se usato col comando `\repeat volta`. Il tipo corretto di stanghetta di ripetizione, ovvero inizio della ripetizione, fine della ripetizione e doppia ripetizione, viene selezionato automaticamente. Il corrispondente segno "D.S." deve essere agiunto manualmente.
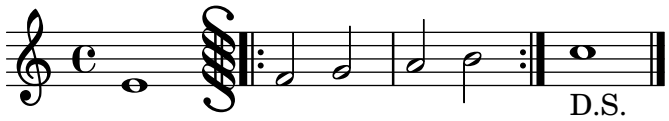
Lontano da una ripetizione:

```
\relative {
  e'1
  \inStaffSegno
  f2 g a b
  c1_"D.S." \bar "|."
}
```



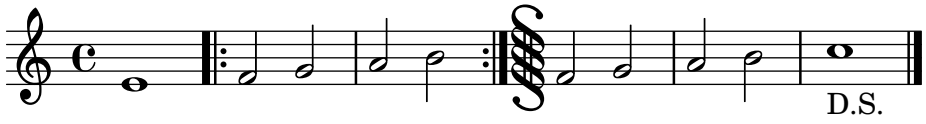

All'inizio di una ripetizione:

```
\relative {
  e'1
  \repeat volta 2 {
    \inStaffSegno  % inizio ripetizione
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```




Alla fine di una ripetizione:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \inStaffSegno  % fine ripetizione
  }
  f2 g a b
  c1_"D.S." \bar "|."
}
```



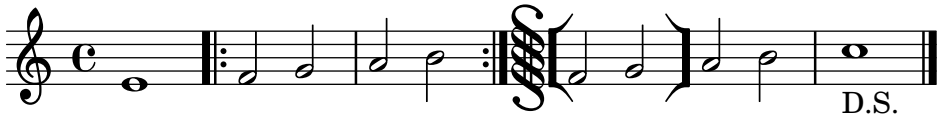

Tra due ripetizioni:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }
  \inStaffSegno  % doppia ripetizione
  \repeat volta 2 {
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```




Si possono impostare simboli alternativi delle stanghette modificando nel contesto Score le proprietà `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` o `doubleRepeatSegnoType` per il tipo di stanghetta richiesto. I tipi di stanghetta alternativi devono essere selezionati dai tipi predefiniti o dai tipi precedentemente definiti col comando `\defineBarLine` (vedi [Stanghette], pagina 101).

```
\defineBarLine ":|.S[" #'(":|." "S[" "")
\defineBarLine "]" #'("]" "" "")
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \once \set Score.endRepeatSegnoType = ":|.S["
    \inStaffSegno
  }
  f2 g \bar "]" a b
  c1_"D.S." \bar "|."
}
```
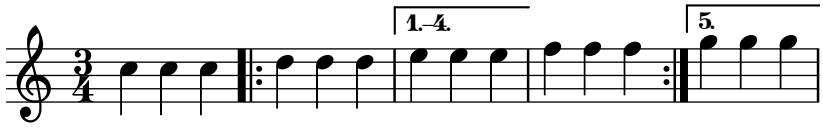



## Frammenti di codice selezionati

*Accorciare le parentesi delle volte*

Per impostazione predefinita, le parentesi delle volte si estendono per tutta l'alternativa, ma si possono accorciare impostando `voltaSpannerDuration`. Nell'esempio seguente, la parentesi dura una misura, che ha una durata di 3/4.

```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3/4)
```

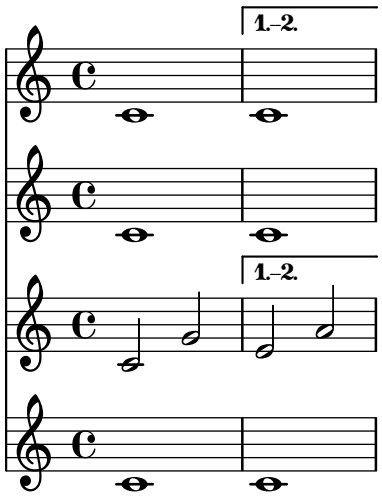
```
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```




*Aggiungere le parentesi delle volte a altri righi*

L'incisore `Volta_engraver` risiede nel contesto `Score`, quindi le parentesi delle ripetizioni appaiono di norma soltanto sul rigo superiore. Questo comportamento può essere modificato aggiungendo l'incisore `Volta_engraver` al contesto `Staff` in cui si desidera far apparire le parentesi; si veda anche il frammento "Volta multirigo".

```
<<
  \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' } }
>>
```

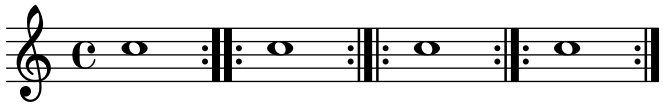



*Impostare la doppia ripetizione predefinita per le volte*

Esistono tre diversi stili di doppie ripetizioni per le volte, che si possono impostare con `doubleRepeatType`.

```
\relative c'' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":..:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.:"
```
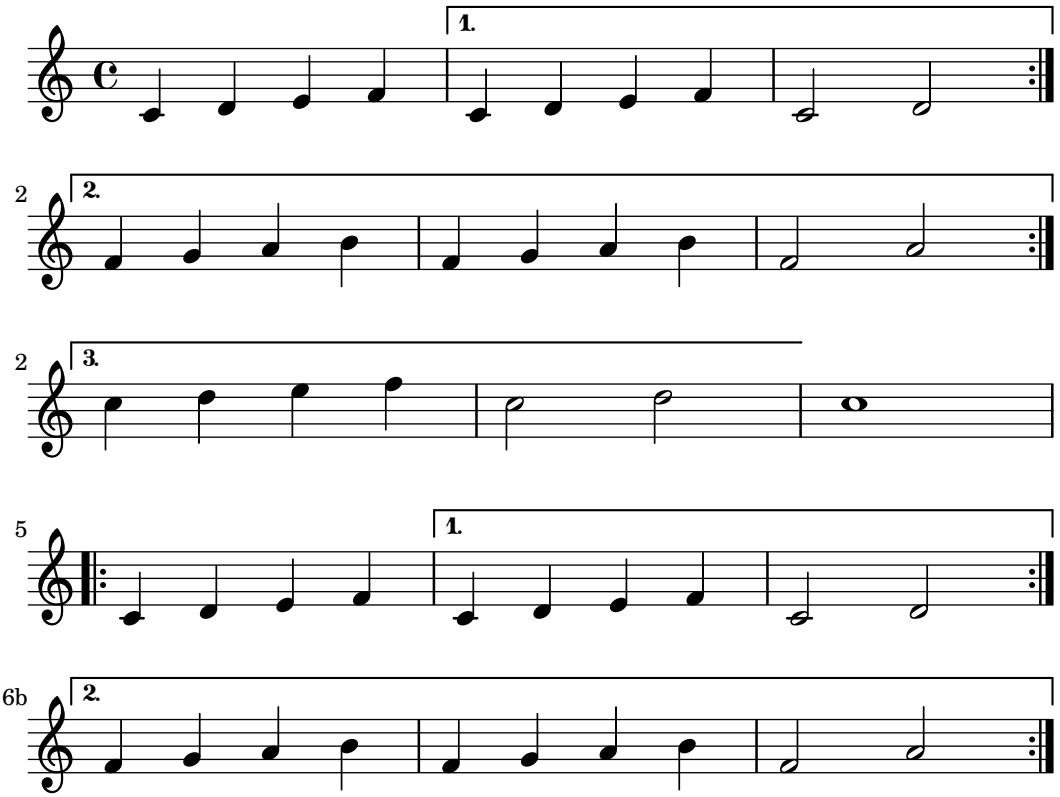
```
  \repeat volta 1 { c1 }
}
```
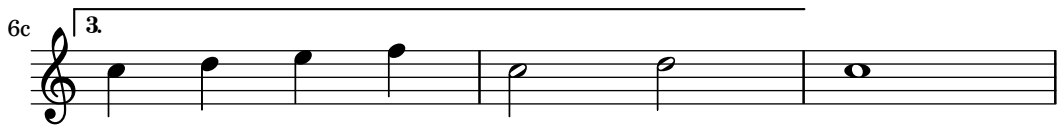


*Numeri di battuta alternativi*

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```






## Vedi anche

Glossario Musicale: Sezione “ripetizione” in *Glossario Musicale*, Sezione “volta” in *Glossario Musicale*.

Guida alla notazione: [Stanghette], pagina 101, Sezione 5.1.4 [Modifica dei componenti aggiuntivi di un contesto], pagina 600, [Modifica di legature di valore e di portamento], pagina 652, [Gestione del tempo], pagina 122.

File installati: `ly/engraver-init.ly`.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VoltaBracket” in *Guida al Funzionamento Interno*, Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “VoltaRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “UnfoldedRepeatedMusic” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le legature di portamento che si estendono da un blocco `\repeat` verso un blocco `\alternative` funzioneranno solo nel primo finale alternativo. L’aspetto grafico di una legatura di portamento che continua negli altri finali alternativi può essere simulato con `\repeatTie` se la legatura si estende solo su una nota del blocco dell’alternativa, sebbene questo metodo non funzioni in `TabStaff`. Altri metodi che si possono adattare per indicare legature di portamento che continuano su varie note dei blocchi di alternativa, e che funzionano anche nei contesti `TabStaff`, sono presentati in [Modifica di legature di valore e di portamento], pagina 652.

Inoltre le legature di portamento non possono ricollegarsi dalla fine di un’alternativa all’inizio della ripetizione.

I glissandi che si estendono da un blocco `\repeat` in un blocco `\alternative` funzioneranno soltanto per il primo finale alternativo. L’aspetto grafico di un glissando che continua negli altri finali alternativi può essere indicato creando un glissando che inizia su una nota di abbellimento nascosta. Vedere ad esempio il frammento “Estendere i glissandi attraverso le ripetizioni” nei Frammenti Selezionati in [Glissando], pagina 142.

Se una ripetizione che inizia con una misura incompleta ha un blocco `\alternative` che contiene modifiche alla proprietà `measureLength`, l’uso di `\unfoldRepeats` causerà l’erroneo posizionamento delle stanghette e degli avvisi di controllo di battuta.

Una ripetizione annidata come la seguente

```
\repeat ...
\repeat ...
\alternative
```

è ambigua, perché non è chiaro a quale `\repeat` appartenga il blocco `\alternative`. Questa ambiguità si risolve facendo in modo che `\alternative` appartenga sempre al blocco `\repeat` interno. Per chiarezza, si consiglia di usare le parentesi in queste situazioni.
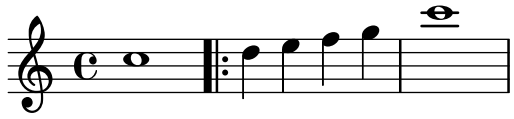
## Indicazioni di ripetizione manuali

> **Nota:** Questi metodi vengono usati solo per mostrare tipi di ripetizioni inusuali, e potrebbero causare un comportamento inaspettato. Nella maggior parte dei casi, le ripetizioni devono essere create col comando standard `\repeat` oppure stampando le stanghette opportune. Maggiori informazioni in [Stanghette], pagina 101.

La proprietà `repeatCommands` permette di controllare la formattazione delle ripetizioni. Il suo valore è una lista Scheme dei comandi di ripetizione.

`start-repeat`
: Stampa una stanghetta `.|:`.
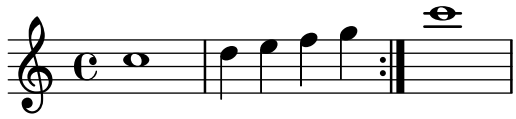
```
\relative {
  c''1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
  c1
}
```



Come vuole la pratica comune di incisione, i segni di ripetizione non vengono stampati all'inizio di un brano.
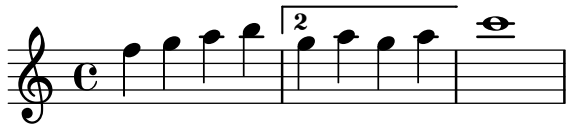
`end-repeat`
: Stampa una stanghetta `:|.`:

```
\relative {
  c''1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
  c1
}
```



`(volta` *numero*`) ... (volta #f)`
: Crea una nuova volta col numero specificato. La parentesi della volta deve essere terminata esplicitamente, altrimenti non sarà stampata.

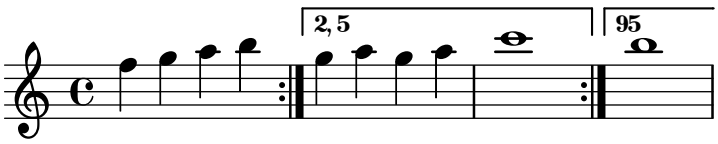
```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2"))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}
```




Comandi di ripetizione multipli possono trovarsi nello stesso punto:
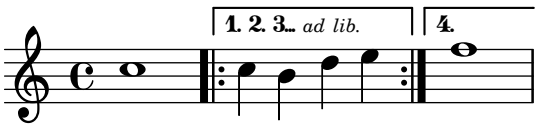
```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2, 5") end-repeat)
  g4 a g a
```

```
  c1
  \set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
  b1
  \set Score.repeatCommands = #'((volta #f))
}
```




Si può includere del testo nella parentesi della volta. Il testo può consistere di un numero, di più numeri o di un'indicazione testuale, si veda Sezione 1.8.2 [Formattazione del testo], pagina 246. Il modo più semplice per usare del testo è definirlo prima e poi includerlo nella lista Scheme,

```
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative {
  c''1
  \set Score.repeatCommands =
    #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
```




## Vedi anche

Guida alla notazione: [Stanghette], pagina 101, Sezione 1.8.2 [Formattazione del testo], pagina 246.

Frammenti: Sezione "Repeats" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "VoltaBracket" in *Guida al Funzionamento Interno*, Sezione "RepeatedMusic" in *Guida al Funzionamento Interno*, Sezione "VoltaRepeatedMusic" in *Guida al Funzionamento Interno*.
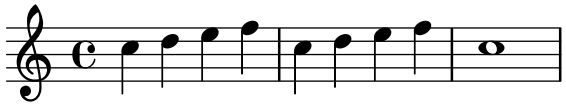
## Ripetizioni ricopiate

Col comando `unfold`, le ripetizioni possono servire a semplificare la scrittura di musica ripetitiva. La sintassi è

```
\repeat unfold numero-di-ripetizioni espressione-musicale
```

dove *`espressione-musicale`* è la musica e *`numero-di-ripetizioni`* è il numero di volte per cui è ripetuta *`espressione-musicale`*.

```
\relative {
  \repeat unfold 2 { c''4 d e f }
  c1
}
```



In alcuni casi, specialmente in un contesto `\relative`, la funzione `\repeat unfold` non equivale a riscrivere l'espressione musicale più volte. Ad esempio
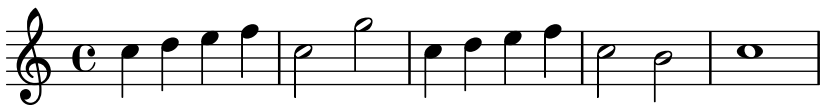
```
\repeat unfold 2 { a'4 b c }
```

non equivale a

```
a'4 b c | a'4 b c
```
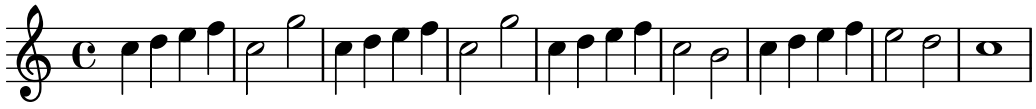
Le ripetizioni dispiegate (unfold) possono avere dei finali alternativi.

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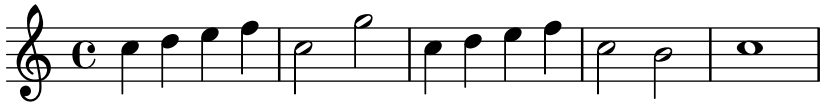


Se il numero di ripetizioni è maggiore del numero di finali alternativi, la prima alternativa viene applicata più volte, finché le alternative rimaste non esauriscono il numero totale delle ripetizioni.

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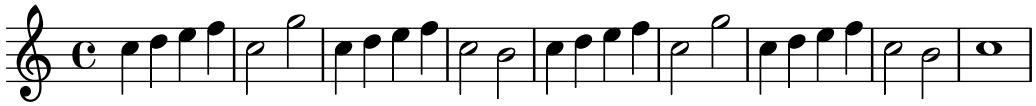


Se il numero di finali alternativi è maggiore del numero di ripetizioni, solo le prime alternative vengono applicate. Le alternative rimanenti saranno ignorate e non verranno stampate.

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



È anche possibile annidare molteplici funzioni `unfold` (con o senza finali alternativi).

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



Gli accordi si ripetono col simbolo di ripetizione dell'accordo `q`. Vedi [Ripetizione di un accordo], pagina 171.

> **Nota:** Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione "Appare un rigo in più" in *Uso del Programma*.

### Vedi anche

Guida alla notazione: [Ripetizione di un accordo], pagina 171.

Frammenti: Sezione "Repeats" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "RepeatedMusic" in *Guida al Funzionamento Interno*, Sezione "UnfoldedRepeatedMusic" in *Guida al Funzionamento Interno*.

## 1.4.2 Ripetizioni brevi

Questa sezione tratta il modo in cui inserire brevi ripetizioni. Le ripetizioni brevi possono avere due forme: segni di tratto obliquo o percentuale per rappresentare le ripetizioni di una singola nota, di una singola misura o di due misure; tremolo negli altri casi.

### Ripetizioni con percentuale

Brevi sezioni ripetute vengono stampate la prima volta volta e le ripetizioni vengono sostituite da un apposito segno.

La sintassi è

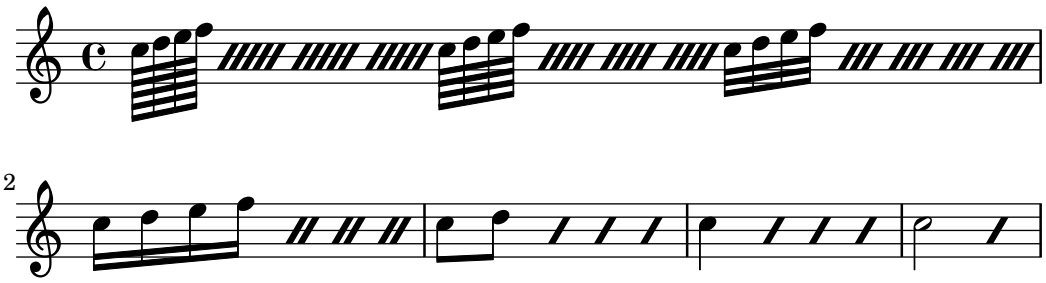
```
\repeat percent numero espressione-musicale
```

dove *`espressione-musicale`* è l'espressione musicale da ripetere.

Fraseggi più brevi di una misura vengono sostituiti dal tratto obliquo.
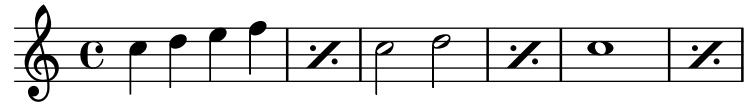
```
\relative c'' {
  \repeat percent 4 { c128 d e f }
  \repeat percent 4 { c64 d e f }
  \repeat percent 5 { c32 d e f }
  \repeat percent 4 { c16 d e f }
  \repeat percent 4 { c8 d }
```

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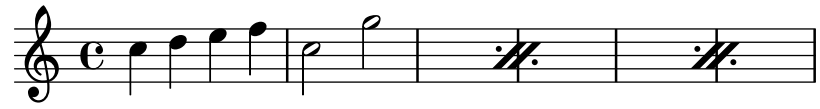


Fraseggi di una o due misure vengono sostituiti da simboli simili alla percentuale.

```
\relative c'' {
  \repeat percent 2 { c4 d e f }
  \repeat percent 2 { c2 d }
  \repeat percent 2 { c1 }
}
```

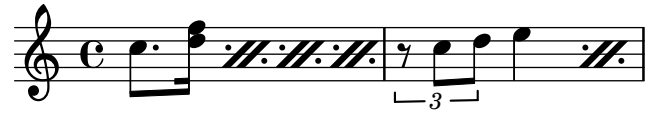


```
\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}
```



Fraseggi più brevi di una misura ma con durate miste adottano un simbolo di doppia percentuale.

```
\relative {
  \repeat percent 4 { c''8. <d f>16 }
  \repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
}
```

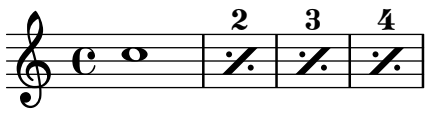



## Frammenti di codice selezionati

*Contatore della ripetizione con segno percentuale*

Le ripetizioni di misura che hanno più di due ripetizioni possono avere un contatore se si cambia la proprietà opportuna, come mostra questo esempio:

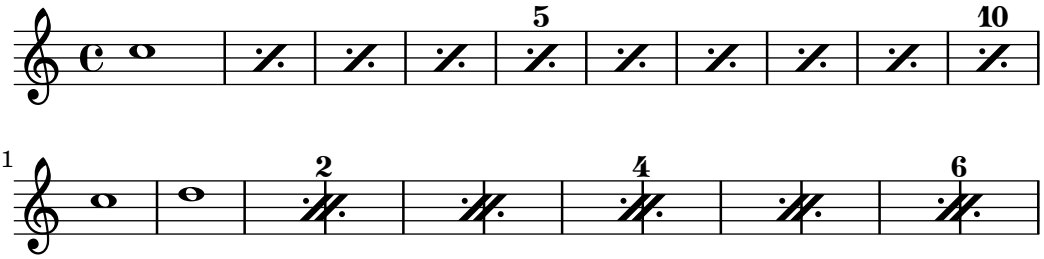
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



*Visibilità del conto della ripetizione con segno percentuale*

I contatori della ripetizione con segno percentuale possono essere mostrati a intervalli regolari impostando la proprietà di contesto `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 }
}
```
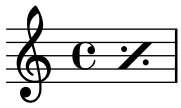


*Ripetizioni con segni di percentuale isolati*

Si possono stampare anche segni di percentuale isolati.

```
makePercent =
#(define-music-function (note) (ly:music?)
   "Make a percent repeat the same length as NOTE."
   (make-music 'PercentEvent
               'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



## Vedi anche

Glossario Musicale: Sezione “percent repeat” in *Glossario Musicale*, Sezione “simile” in *Glossario Musicale*.
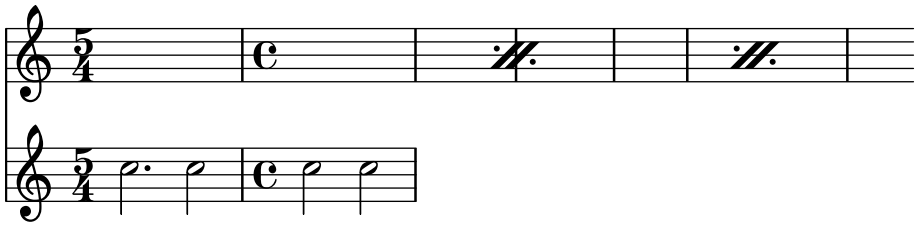
Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RepeatSlash” in *Guida al Funzionamento Interno*, Sezione “RepeatSlashEvent” in *Guida al Funzionamento Interno*, Sezione “DoubleRepeatSlash” in *Guida al Funzionamento Interno*, Sezione “PercentRepeat” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “Percent_repeat_engraver” in *Guida al Funzionamento Interno*, Sezione “DoublePercentEvent” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeat” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “Double_percent_repeat_engraver” in *Guida al Funzionamento Interno*, Sezione “Slash_repeat_engraver” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

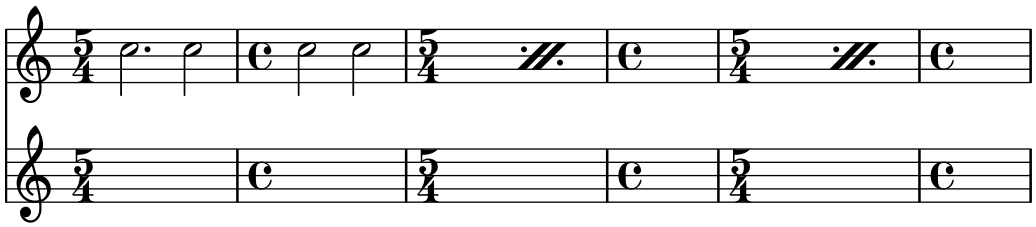
Le ripetizioni con percentuale non contengono nient'altro che il segno di percentuale; in particolare, i cambi di tempo non saranno ripetuti.

```
\repeat percent 3 { \time 5/4 c2. 2 \time 4/4 2 2 }
```



Qualsiasi cambio di tempo o comando `\partial` devono trovarsi in passaggi paralleli *esterni* a qualsiasi ripetizione con percentuale, per esempio su una traccia di tempo separata.

```
<<
  \repeat percent 3 { c2. 2 2 2 }
  \repeat unfold 3 { \time 5/4 s4*5 \time 4/4 s1 }
>>
```
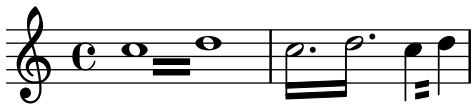


### Ripetizioni con tremolo

I tremoli possono avere due forme: alternanza tra due note, o due accordi, e rapida ripetizione di una singola nota o accordo. I tremoli costituiti da un'alternanza si indicano con delle travature che collegano le note o gli accordi che si alternano, mentre i tremoli che consistono in una rapida ripetizione di una nota singola si indicano aggiungendo delle travature o dei tratti di suddivisione obliqui alla singola nota.

Per inserire i segni del tremolo tra le note, si usa `\repeat` con lo stile tremolo:

```
\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}
```



La sintassi di `\repeat tremolo` prevede specificamente che all'interno delle parentesi siano indicate due note, e che il numero di ripetizioni corrisponda a un valore espresso in durate di note normali o puntate. Dunque `\repeat tremolo 7` è valido e produce una nota doppiamente puntata, mentre `\repeat tremolo 9` non è valido.

La durata del tremolo equivale alla durata dell'espressione musicale tra parentesi moltiplicata per il numero di ripetizioni: `\repeat tremolo 8 { c16 d16 }` corrisponde a un tremolo di una semibreve, rappresentata come due semibrevi unite dalle travature del tremolo.
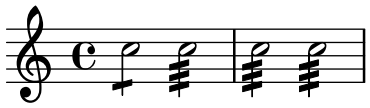
Ci sono due modi di inserire dei segni di tremolo su una singola nota. Anche in questo caso si usa la sintassi `\repeat tremolo`, ma la nota non deve essere racchiusa tra parentesi:

```
\repeat tremolo 4 c'16
```



Si può ottenere lo stesso output aggiungendo `:N` dopo la nota, dove *N* indica la durata della suddivisione (deve essere almeno 8). Se *N* è 8, viene aggiunta una travatura al gambo della nota. Se *N* è omesso, viene usato l'ultimo valore:

```
\relative {
  c''2:8 c:32
  c: c:
}
```

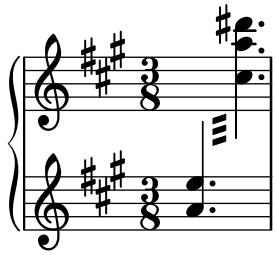


## Frammenti di codice selezionati

*Tremoli attraverso i righi*

Dato che `\repeat tremolo` si aspetta esattamente due argomenti musicali per i tremoli di accordi, la nota o l'accordo che cambiano rigo in un tremolo che attraversa i righi devono essere posti tra parentesi graffe insieme al comando `\change Staff`.
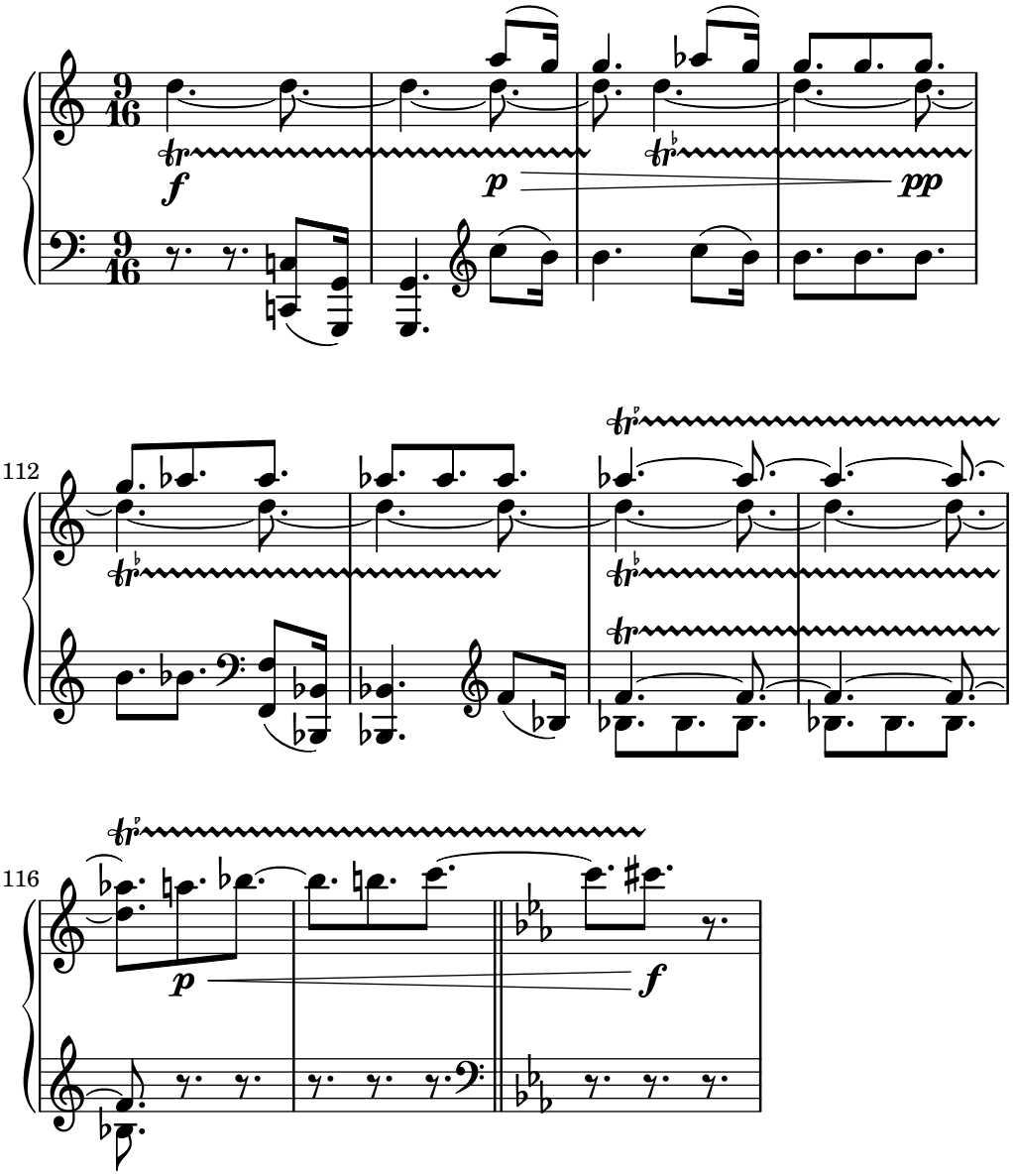
```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```



### Vedi anche

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

## 1.5 Note simultanee




La polifonia in musica si riferisce alla coesistenza simultanea di più di una voce in un brano musicale. In LilyPond la polifonia si riferisce alla coesistenza di più voci sullo stesso rigo.

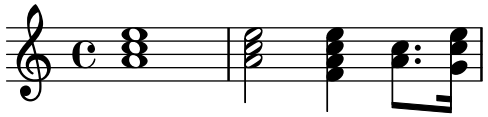
### 1.5.1 Una voce

In questa sezione vengono spiegate le note simultanee che fanno parte di un’unica voce.

## Note in un accordo

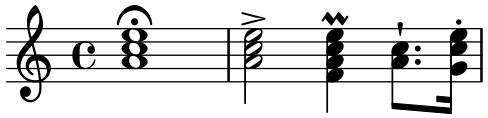
Un accordo si forma racchiudendo una serie di altezze tra < e > e può essere seguito da una durata, come accade per le semplici note.

```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```



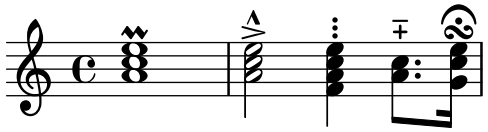
Proprio come per le note, si possono specificare le articolazioni da riferire all'accordo.

```
\relative {
  <a' c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
}
```



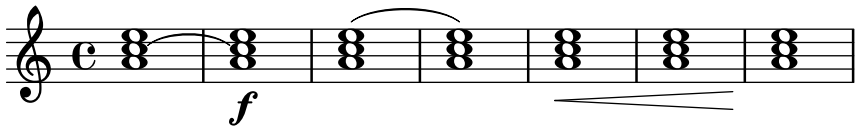
Si possono specificare abbellimenti e articolazioni per ogni nota che fa parte dell'accordo.

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



Tuttavia, alcuni elementi della notazione, come le dinamiche e le forcelle, devono essere attaccate all'accordo invece che alle sue singole note, altrimenti non appariranno. Altri elementi della notazione, come le diteggiature e le legature di portamento, saranno posizionate in modo nettamente diverso se attaccate alle note di un accordo invece che a un accordo intero o a singole note.
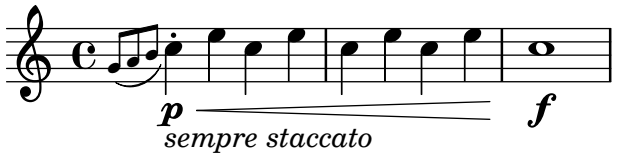
```
\relative {
  <a'\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>)
  <a c e>\< <a c e> <a c e>\!
}
```




Un accordo si comporta semplicemente come un contenitore di note, articolazioni e altri elementi. Di conseguenza, un accordo privo di note non ha una durata. Qualsiasi articolazione attaccata a un accordo vuoto si troverà nel momento musicale della nota o accordo seguenti e si combinerà con questi (possibilità più complesse di combinazione sono spiegate in [Espressioni simultanee], pagina 173):

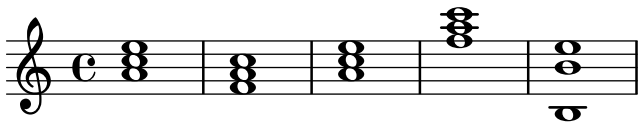
```
\relative {
```

```
  \grace { g'8( a b }
  <> ) \p \< -. -\markup \italic "sempre staccato"
  \repeat unfold 4 { c4 e }  c1\f
}
```




Si può usare la modalità relativa per indicare l'altezza degli accordi. La prima nota di ogni accordo è sempre relativa alla prima nota dell'accordo che lo precede oppure, se non c'è un accordo precedente, è relativa all'altezza dell'ultima nota che precede l'accordo. Le altezze di tutte le altre note dell'accordo sono relative alla nota che le precede *all'interno dell'accordo*.

```
\relative {
  <a' c e>1 <f a c> <a c e> <f' a c> <b, e b,>
}
```



Maggiori informazioni sugli accordi si trovano in Sezione 2.7 [Notazione per accordi], pagina 419.

## Vedi anche

Glossario Musicale: Sezione "accordo" in *Glossario Musicale*.

Manuale d'apprendimento: Sezione "Combinare le note negli accordi" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.7 [Notazione per accordi], pagina 419, [Articolazioni e abbellimenti], pagina 124, [Ottava relativa], pagina 2, Sezione 1.5.2 [Più voci], pagina 175.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

## Problemi noti e avvertimenti

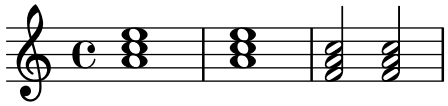
Gli accordi che contengono più di due altezze in uno spazio del rigo, come ad esempio '`<e f! fis!>`', presentano le teste di tali note sovrapposte. A seconda della situazione, si può migliorare l'aspetto con

- l'uso temporaneo di Sezione 1.5.2 [Più voci], pagina 175, '`<< f! \\ <e fis!> >>`',
- la trascrizione enarmonica di una o più altezze, '`<e f ges>`', oppure
- i [Cluster], pagina 175.

## Ripetizione di un accordo

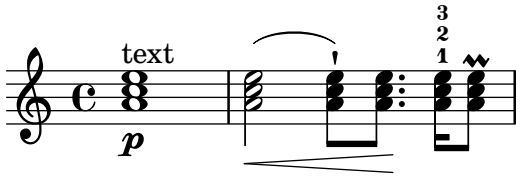
Per inserire la musica più rapidamente, si può usare una scorciatoia che ripete l'accordo precedente. Il simbolo di ripetizione dell'accordo è `q`:

```
\relative {
  <a' c e>1 q <f a c>2 q
}
```



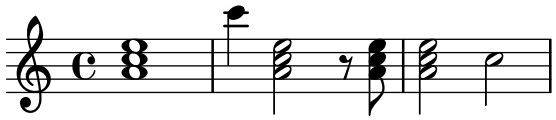
Come nel caso dei normali accordi, il simbolo di ripetzione dell'accordo si può usare con le durate, le articolazioni, i testi a margine, le legature di portamento, le travature, etc. dato che solo le altezze dell'accordo precedente vengono duplicate.

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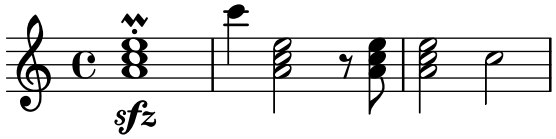



Il simbolo di ripetizione dell'accordo ricorda sempre l'ultimo accordo inserito, quindi è possibile inserire l'accordo più recente anche se nel frattempo sono state inserite altre note (senza accordi) o pause.

```
\relative {
  <a' c e>1 c'4 q2 r8 q8 |
  q2 c, |
}
```



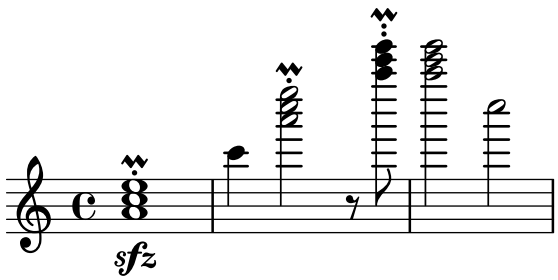
Tuttavia questo simbolo non conserva le dinamiche, le articolazioni o gli abbellimenti dell'accordo precedente.

```
\relative {
  <a'-. c\prall e>1\sfz c'4 q2 r8 q8 |
  q2 c, |
}
```




Per far sì che alcuni elementi siano conservati, si può invocare esplicitamente la funzione `\chordRepeats` con un'ulteriore argomento che indica una lista di *tipi di evento* da mantenere, a meno che eventi di quel tipo non siano già presenti nell'accordo `q` stesso.
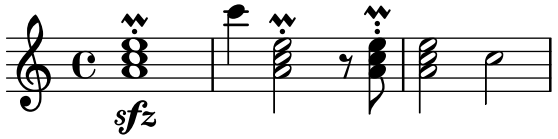
```
\relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c, |
}
```




In questo esempio l'uso di `\chordRepeats` all'interno di un blocco `\relative` produce risultati indesiderati: gli eventi di un accordo, una volta espansi, non si distinguono più per essere stati inseriti come accordi normali, quindi `\relative` assegna un'ottava basata sul contesto corrente.

Dato che `\relative` annidati non si influenzano l'un l'altro, si può usare un altro `\relative` dentro `\chordRepeats` per stabilire le relazioni di ottava prima di espandere gli accordi ripetuti. In questo caso l'intero contenuto del `\relative` più interno non influenza quello esterno; ecco perché in questo esempio la nota finale è stata specificata con un'ottava diversa.

```
\new Voice
\relative c'' {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c |
}
```




Le interazioni con `\relative` si verificano solo con chiamate esplicite di `\chordRepeats`: l'espansione implicita all'inizio della creazione della partitura viene fatta in un momento in cui tutti i `\relative` sono stati già elaborati.

## Vedi anche

Guida alla notazione: Sezione 2.7 [Notazione per accordi], pagina 419, [Articolazioni e abbellimenti], pagina 124.
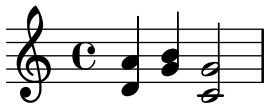
File installati: `ly/chord-repetition-init.ly`.

## Espressioni simultanee
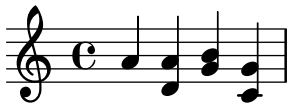
Una o più espressioni musicali racchiuse tra due coppie di parentesi uncinate sono considerate simultanee. Se la prima espressione inizia con una nota singola o se l'intera espressione simultanea appare esplicitamente all'interno di una voce, sarà posta in un solo rigo; altrimenti gli elementi dell'espressione simultanea saranno messi in righi separati.

Gli esempi seguenti mostrano espressioni simultanee su un rigo:

```
\new Voice {  % voce singola esplicita
  << \relative { a'4 b g2 }
     \relative { d'4 g c,2 } >>
}
```
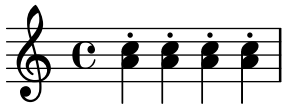


```
\relative {
  % prima nota singola
  a' << \relative { a'4 b g }
        \relative { d'4 g c, } >>
}
```



Questo può essere utile se le sezioni simultanee hanno durate identiche, ma i tentativi di collegare note con durate diverse allo stesso gambo causerà degli errori. Le note, le articolazioni e le modifiche delle proprietà in una *singola* voce (‘`Voice`’) sono raccolte e create secondo l’ordine della musica:
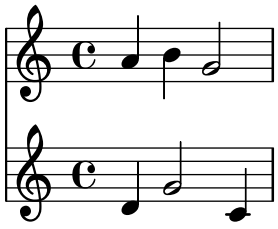
```
\relative {
  <a' c>4-.  <>-. << c a >>  << { c-. <c a> } { a s-. } >>
}
```



Per poter inserire gambi o travature multiple e variare le durate o altre proprietà di note riferibili allo stesso momento musicale, occorre usare più voci.

L’esempio seguente mostra come le espressioni simultanee possano generare implicitamente righi multipli:

```
% nessuna singola nota precede l'espressione simultanea
<< \relative { a'4 b g2 }
   \relative { d'4 g2 c,4 } >>
```



In questo caso le durate diverse non causano problemi perché sono interpretate in voci diverse.

## Problemi noti e avvertimenti

Se le note appartenenti a due o più voci, senza che sia stato specificato uno spostamento, hanno i gambi nella stessa direzione, il messaggio

```
attenzione: questa voce ha bisogno di un'impostazione \voiceXx o \shiftXx
```

apparirà durante la compilazione del file. Si può evitare con:
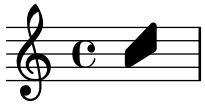
```
\override NoteColumn.ignore-collision = ##t
```

Tuttavia, questo comando non si limita a eliminare l’avvertimento, ma impedisce qualsiasi risoluzione delle collisioni, e potrebbe comportare altri effetti indesiderati (vedi anche i *Problemi noti* in [Risoluzione delle collisioni], pagina 179).

### Cluster

Un cluster prescrive l'esecuzione simultanea di tutti i suoni compresi in un determinato intervallo. Può essere rappresentato come un involucro che contiene le note che ne fanno parte. Si inserisce applicando la funzione `\makeClusters` a una sequenza di accordi, ad esempio:

```
\relative \makeClusters { <g' b>2 <c g'> }
```



Si possono inserire insieme sullo stesso rigo le normali note e i cluster, anche contemporaneamente. In tal caso non viene fatto alcun tentativo di evitare automaticamente collisioni tra le note normali e i cluster.

### Vedi anche

Glossario Musicale: Sezione "cluster" in *Glossario Musicale*.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "ClusterSpanner" in *Guida al Funzionamento Interno*, Sezione "ClusterSpannerBeacon" in *Guida al Funzionamento Interno*, Sezione "Cluster_spanner_engraver" in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

I cluster hanno un buon aspetto solo se durano almeno per due accordi; altrimenti appaiono troppo stretti.

I cluster non hanno un gambo e non possono indicare delle durate da soli, ma la lunghezza del cluster è determinata dalle durate degli accordi che lo definiscono. Più cluster distinti devono essere separati da una pausa.

I cluster non generano output MIDI.

## 1.5.2 Più voci
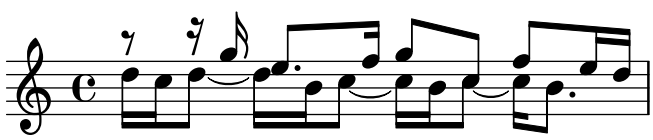
Questa sezione presenta le note simultanee in più voci o più righi.

### Polifonia su un solo rigo

#### *Istanziare esplicitamente le voci*

La struttura di base necessaria per ottenere più voci indipendenti in un solo rigo è illustrata nell'esempio seguente:

```
\new Staff <<
  \new Voice = "prima"
    \relative { \voiceOne r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \new Voice= "seconda"
    \relative { \voiceTwo d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Le voci sono istanziate esplicitamente e vengono contrassegnate da dei nomi. I comandi `\voiceOne` ... `\voiceFour` impostano le voci in modo che la prima e terza voce abbiano i

gambi in su, la seconda e la quarta voce i gambi in giù, le teste di nota della terza e quarta voce siano spostate orizzontalmente e le pause in ciascuna voce siano spostate automaticamente per evitare collisioni. Il comando `\oneVoice` ripristina tutte le impostazioni della voce alle direzioni neutrali predefinite.
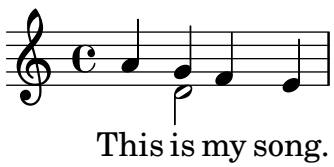
### *Passaggi polifonici temporanei*

Un passaggio polifonico temporaneo si può creare col seguente costrutto:

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

In questo esempio la prima espressione all'interno di un passaggio polifonico temporaneo è posta nel contesto `Voice` che era in uso immediatamente prima del passaggio polifonico e quello stesso contesto `Voice` continua dopo la sezione temporanea. Le altre espressioni comprese nelle parentesi uncinate vengono assegnate a voci temporanee distinte. Questo permette di assegnare il testo a una voce che continua prima, durante e dopo una sezione polifonica:

```
\relative <<
  \new Voice = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
  This is my song.
  }
>>
```




I comandi `\voiceOne` e `\voiceTwo` sono necessari per definire le impostazioni di ogni voce.
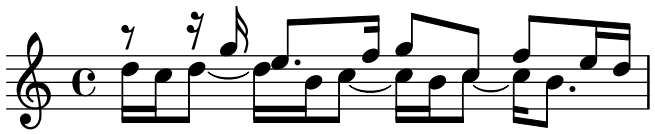
### *Il costrutto con la doppia barra inversa (backslash)*

Il costrutto `<< {...} \\ {...} >>`, in cui due (o più) espressioni sono separate da due barre inverse (backslash), si comporta diversamente dal costrutto simile privo delle due barre: *tutte* le espressioni in questo costrutto vengono assegnate a nuovi contesti `Voice`. Questi nuovi contesti `Voice` vengono creati implicitamente e ad essi vengono assegnati dei nomi prestabiliti `"1"`, `"2"`, etc.

Il primo esempio potrebbe essere riscritto nel modo seguente:
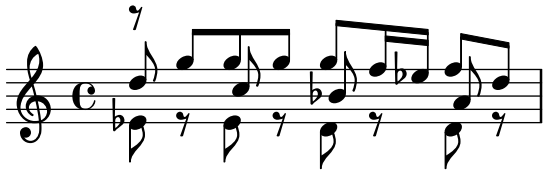
```
<<
```

```
  \relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \\
  \relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Questa sintassi si usa quando non importa che le voci temporanee siano create e poi eliminate. A queste voci create implicitamente vengono assegnate le impostazioni equivalenti all'uso dei comandi `\voiceOne` ... `\voiceFour`, nell'ordine in cui appaiono nell'input.

Nell'esempio seguente, la voce intermedia ha i gambi in su, dunque viene inserita in terza posizione in modo che diventi la terza voce, che ha i gambi in su. Si usano le pause spaziatrici per evitare il raddoppio delle pause ordinarie.

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



In tutti i brani, a eccezione di quelli più semplici, è consigliabile creare contesti `Voice` espliciti come è spiegato in Sezione "Contesti e incisori" in *Manuale di Apprendimento* e Sezione "Definire esplicitamente le voci" in *Manuale di Apprendimento*.

### *Ordine delle voci*
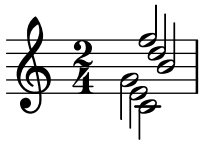
Quando si inseriscono più voci nel file di input, conviene usare il seguente ordine:

```
Voce 1: la più alta
Voce 2: la più bassa
Voce 3: la seconda più alta
Voce 4: la seconda più bassa
Voce 5: la  terza più alta
Voce 6: la terza più bassa
etc.
```

Sebbene possa sembrare controintuitivo, ciò semplifica il processo automatico di formattazione. Si noti che le voci con numero dispari hanno i gambi in su, quelle con numero pari hanno i gambi in giù:
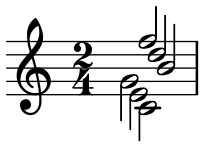
```
\new Staff <<
  \time 2/4
  { f''2 }  % 1: la più alta
  \\
  { c'2  }  % 2: la più bassa
```

```
  \\
  { d''2 }  % 3: seconda più alta
  \\
  { e'2  }  % 4: seconda più bassa
  \\
  { b'2  }  % 5: terza più alta
  \\
  { g'2  }  % 6: terza più bassa
>>
```



Quando si desidera inserire le voci in un ordine diverso, conviene usare il comando `\voices`:

```
\new Staff \voices 1,3,5,6,4,2 <<
  \time 2/4
  { f''2 }  % 1: la più alta
  \\
  { d''2 }  % 3: seconda più alta
  \\
  { b'2  }  % 5: terza più alta
  \\
  { g'2  }  % 6: terza più bassa
  \\
  { e'2  }  % 4: seconda più bassa
  \\
  { c'2  }  % 2: la più bassa
>>
```
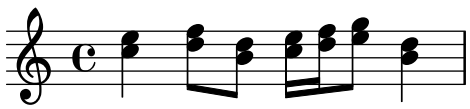


> **Nota:** Il testo e gli estensori (come le legature di portamento e di valore, le forcelle, etc.) non possono 'attraversare' le voci.

### *Durate identiche*

Nel caso speciale in cui si desideri inserire sezioni musicali parallele con il medesimo ritmo, si possono combinare in un unico contesto `Voice`, formando dunque degli accordi. Per farlo, vanno racchiuse in un semplice costrutto musicale simultaneo all'interno di una voce esplicita:

```
\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>
```



Questo metodo produce strane travature e avvertimenti se le sezioni musicali non hanno lo stesso ritmo.

### Comandi predefiniti

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

### Vedi anche

Manuale d'apprendimento: Sezione "Le voci contengono la musica" in *Manuale di Apprendimento*, Sezione "Definire esplicitamente le voci" in *Manuale di Apprendimento*.

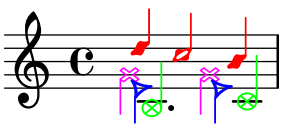
Guida alla notazione: [Righi delle percussioni], pagina 400, [Pause invisibili], pagina 61, [Gambi], pagina 232.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

### Stili di voce

Si possono assegnare colori e forme diverse a ciascuna voce per facilitarne l'identificazione:

```
<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \\
  \relative { \voiceTwoStyle e'2 e }
  \\
  \relative { \voiceThreeStyle b2. c4 }
  \\
  \relative { \voiceFourStyle g'2 g }
>>
```



Il comando `\voiceNeutralStyle` permette di ripristinare l'aspetto predefinito.

### Comandi predefiniti

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

### Vedi anche

Manuale d'apprendimento: Sezione "Sento le Voci" in *Manuale di Apprendimento*, Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.
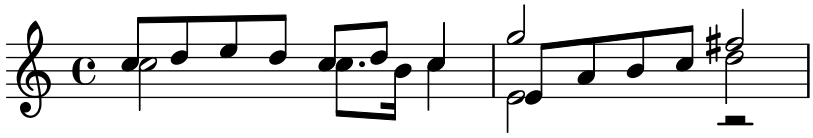
Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

### Risoluzione delle collisioni

Le teste di note che si trovano in voci diverse ma hanno stessa altezza, stessa testa e direzione del gambo opposta vengono unite automaticamente; invece, le note che hanno la stessa testa o la stessa direzione del gambo non vengono unite. Le pause opposte a un gambo in una voce diversa vengono spostate verticalmente. L'esempio seguente mostra tre diverse circostanze, sul primo e terzo movimento della prima battuta e sul primo movimento della seconda battuta, in cui l'unione automatica delle teste di nota non funziona.
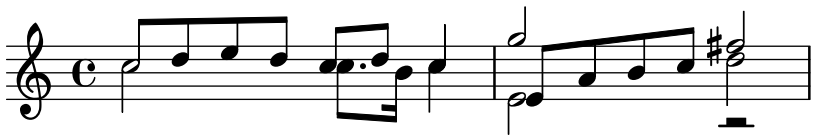
```
<<
  \relative {
```

```
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Note con teste diverse possono essere unite come è mostrato sotto. In questo esempio le teste delle note nel primo battito della prima battuta sono unite:

```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```
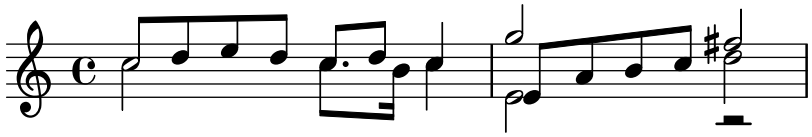


Le minime e le semiminime, invece, non sono unite, perché sarebbe difficile distinguerle.

Anche le teste di note con diversi punti, come nel terzo movimento della prima battuta, possono essere unite:
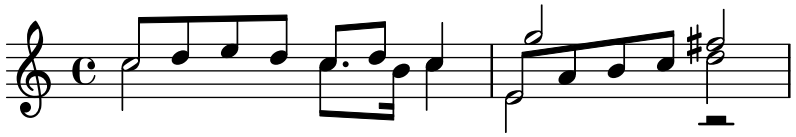
```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    g'2 fis
```

```
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



La minima e la croma all'inizio della seconda misura sono unite per errore, perché l'unione automatica non riesce a completare correttamente l'unione quando tre o più note sono allineate sulla stessa colonna di note: in questo caso la testa di nota unita non è corretta. Per far sì che l'unione selezioni la testa di nota corretta, occorre applicare il comando `\shift` alla nota che non deve essere unita. In questo esempio si usa `\shiftOn` per spostare il Sol superiore ($g$) fuori dalla colonna e di conseguenza `\mergeDifferentlyHeadedOn` funziona correttamente.

```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```
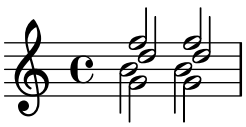


Il comando `\shiftOn` permette (senza forzare) lo spostamento delle note in una voce. Quando si applica `\shiftOn` a una voce, una nota o accordo in quella voce vengono spostati solo se il suo gambo altrimenti entrerebbe in collisione col gambo di un'altra voce, e solo se i gambi che collidono puntano nella stessa direzione. Il comando `\shiftOff` impedisce che si verifichi questo tipo di spostamento.

Per impostazione predefinita, le voci più esterne (solitamente la prima e la seconda voce) hanno specificato `\shiftOff`, mentre le voci più interne (terza e successive) hanno specificato `\shiftOn`. Quando si applica uno spostamento, le voci con i gambi in su (voci dispari) vengono spostate a destra, e le voci con i gambi in giù (voci pari) vengono spostate a sinistra.

Ecco un esempio che aiuta a visualizzare come un'espressione simultanea abbreviata viene espansa internamente.

> **Nota:** Attenzione: con tre o più voci, l'ordine verticale delle voci nel file di input non deve essere lo stesso dell'ordine verticale delle voci del rigo!

```
\new Staff \relative {
  %% inserimento abbreviato
  <<
    { f''2  }  % 1: più alta
    \\
    { g,2 }  % 2: più bassa
    \\
    { d'2 }  % 3: più alta centrale
    \\
    { b2  }  % 4: più  bassa centrale
  >>
  %% espansione interna dell'input precedente
  <<
    \new Voice = "1" { \voiceOne   \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo   \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn  d'2 } % sposta a destra
    \new Voice = "4" { \voiceFour  \shiftOn  b2  } % sposta a sinistra
  >>
}
```



Due ulteriori comandi, `\shiftOnn` e `\shiftOnnn`, mettono a disposizione altri livelli di spostamento che possono essere specificati in modo temporaneo per risolvere delle collisioni in situazioni complesse – vedi Sezione "Esempio musicale" in *Manuale di Apprendimento*.

Le note vengono unite solo se presentano opposta direzione dei gambi (come accade, ad esempio, nella prima o seconda voce o quando i gambi sono impostati esplicitamente in direzioni opposte).

## Comandi predefiniti

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.
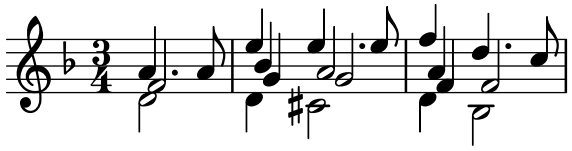
## Frammenti di codice selezionati

*Voci ulteriori per evitare le collisioni*

In alcuni casi di musica polifonica complessa sono necessarie delle voci ulteriori per evitare le collisioni tra note. Se servono più di quattro voci parallele, si possono aggiungere altre voci definendo una variabile con la funzione Scheme function `context-spec-music`.

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

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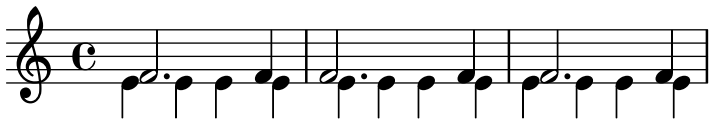


*Spostare le note puntate in polifonia*

Quando una nota puntata della voce più alta viene spostata per evitare una collisione con una nota di un'altra voce, il comportamento predefinito è spostare la nota più alta a destra. Tale comportamento può essere modificato tramite la proprietà `prefer-dotted-right` di `NoteCollision`.

```
\new Staff \relative c' <<
  {
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##f
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##t
    f2. f4
```

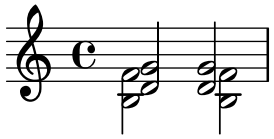
```
  }
  \\
  { e4 e e e e e e e e e e e }
>>
```



*Forzare lo spostamento orizzonatale delle note*

Quando il motore tipografico non riesce a risolvere una situazione, si può usare la sintassi che sovrascrive le decisioni tipografiche. L'unità di misura usata è lo spazio del rigo.

```
\relative c' <<
  {
    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn.force-hshift = #1.7
    <b f'>2
  }
>>
```



## Vedi anche

Glossario Musicale: Sezione "polifonia" in *Glossario Musicale*.

Manuale d'apprendimento: Sezione "Note simultanee" in *Manuale di Apprendimento*, Sezione "Le voci contengono la musica" in *Manuale di Apprendimento*, Sezione "Esempio musicale" in *Manuale di Apprendimento*.

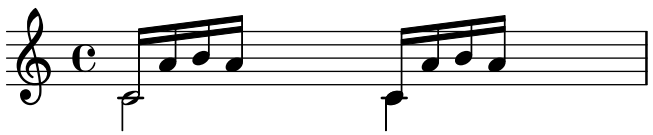
Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "NoteColumn" in *Guida al Funzionamento Interno*, Sezione "NoteCollision" in *Guida al Funzionamento Interno*, Sezione "RestCollision" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se si usa `\override NoteColumn.ignore-collision = ##t`, le note con teste diverse che si trovano in voci diverse saranno unite in modo non corretto.
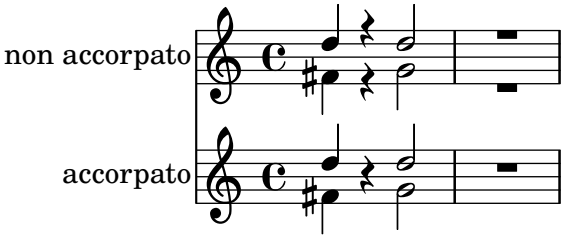
```
\mergeDifferentlyHeadedOn
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
\override NoteColumn.ignore-collision = ##t
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
```



### Accorpare le pause

In caso di voci multiple, di solito si accorpano le pause che capitano in entrambe. Per farlo si usa l'incisore `Merge_rests_engraver`.

```
voiceA = \relative { d''4 r d2 | R1 | }
voiceB = \relative { fis'4 r g2 | R1 | }
\score {
  <<
    \new Staff \with {
      instrumentName = "non accorpato"
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
    \new Staff \with {
      instrumentName = "accorpato"
      \consists "Merge_rests_engraver"
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
  >>
}
```




Impostando la proprietà di contesto `suspendRestMerging` su `##t` è possibile disattivare temporaneamente l'accorpamento delle pause.

### Combinazione automatica delle parti

La combinazione automatica delle parti si usa per combinare in un unico rigo due parti musicali separate. Ciò è utile soprattutto quando si scrivono partiture orchestrali. Viene stampata una sola voce se le due parti musicali sono identiche, ma nei punti in cui sono diverse viene aggiunta una seconda voce. Le direzioni dei gambi sono impostate in su e in giù in base alla voce di appartenenza, mentre le sezioni solistiche e a *due* sono a loro volta identificate e contrassegnate.

La sintassi per la combinazione automatica delle parti è:

`\partCombine` *`espressione-musicale1 espressione-musicale2`*
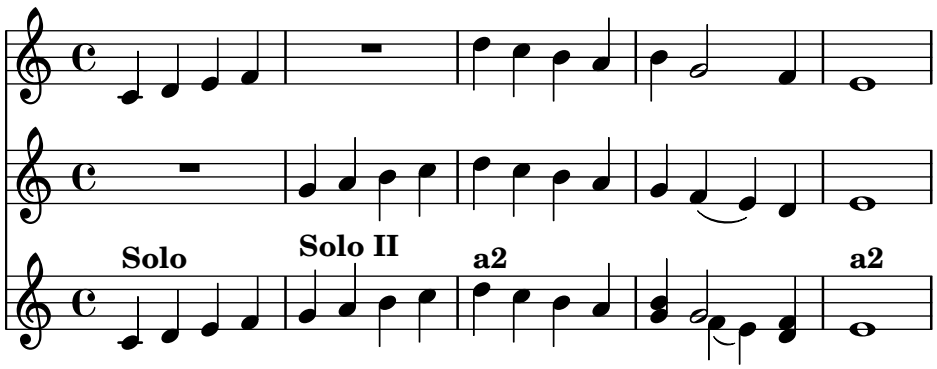
L'esempio seguente illustra il funzionamento di base: le parti sono poste su un unico rigo in modo polifonico e le direzioni dei gambi sono regolate di conseguenza. Si usano le stesse variabili per le parti indipendenti e il rigo combinato.

```
instrumentOne = \relative {
  c'4 d e f |
  R1 |
  d'4 c b a |
```

```
  b4 g2 f4 |
  e1 |
}

instrumentTwo = \relative {
  R1 |
  g'4 a b c |
  d4 c b a |
  g4 f( e) d |
  e1 |
}

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



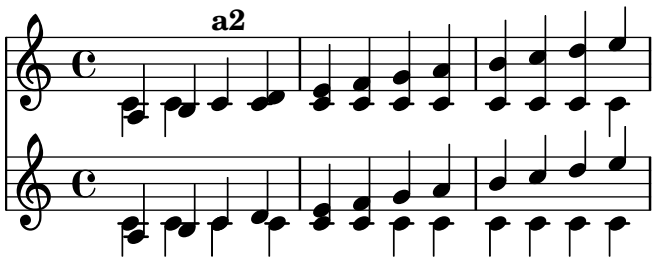

Entrambe le parti hanno note identiche nella terza misura, dunque viene stampata una sola nota. Le direzioni dei gambi e delle legature di portamento e di valore sono impostate automaticamente, a seconda che l'esecuzione delle parti sia solistica o all'unisono. Quando si rende necessario, in caso di polifonia, la prima parte (nel contesto `one`) ha i gambi in "su", mentre la seconda (nel contesto `two`) ha sempre i gambi in "giù". In caso di parti solistiche, la prima e seconda parte sono contrassegnate rispettivamente con "Solo" e "Solo II". Le parti (*a due*) all'unisono sono contrassegnate con la scritta "a2".

Il comportamento predefinito del combinatore delle parti è quello di unire due note della stessa altezza in una nota *a due*, combinare in un accordo note della stessa durata e separate da un intervallo inferiore alla nona e separare in voci distinte le note con un intervallo superiore alla nona (o quando le voci collidono). Tale comportamento può essere modificato con un argomento opzionale di una coppia di numeri dopo il comando `\partCombine`: il primo indica l'intervallo in cui le note iniziano a essere combinate (il valore predefinito è zero) e il secondo dove le note vengono divise in note distinte. Impostando il secondo argomento su zero, il combinatore delle parti divide le note che hanno un intervallo di una seconda o più; impostandolo su uno, divide le note di una terza o più, e così via.

```
instrumentOne = \relative {
  a4 b c d |
  e f g a |
  b c d e |
}
```

```
instrumentTwo = \relative {
  c'4 c c c |
  c c c c |
  c c c c |
}

<<
  \new Staff \partCombine \instrumentOne \instrumentTwo
  \new Staff \partCombine #'(2 . 3) \instrumentOne \instrumentTwo
>>
```




Entrambi gli argomenti di `\partCombine` sono interpretati come contesti `Voice` separati, dunque se la musica viene inserita in modo relativo *entrambe* le parti devono contenere una funzione `\relative`, ovvero:

```
\partCombine
  \relative ... espressione-musicale1
  \relative ... espressione-musicale2
```

Un blocco `\relative` che racchiude un `\partCombine` non ha effetto sulle altezze di *espressione-musicale1* e *espressione-musicale2*.

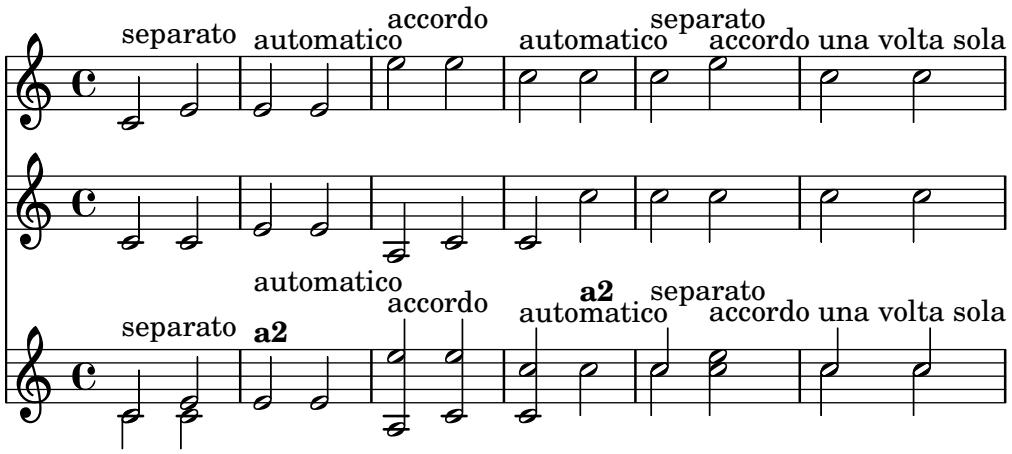
Nelle partiture professionali, spesso le voci sono tenute separate per lunghi passaggi anche se alcune note sono le stesse in entrambe le voci e potrebbero essere stampate come unisono. Combinare le note in un accordo o mostrare una voce come solista, dunque, non è la scelta ideale, perché la funzione `\partCombine` considera ogni nota individualmente. In questo caso si può sovrascrivere la funzione `\partCombine` con uno dei comandi elencati sotto. Tutti i comandi devono essere preceduti da `\once` per applicarli soltanto alla nota successiva dell'espressione musicale.

- `\partCombineApart` mantiene le note su due voci distinte, anche se potrebbero essere combinate in un accordo o in un unisono.
- `\partCombineChords` unisce le note in un accordo.
- `\partCombineUnisono` unisce entrambe le voci come "unisono".
- `\partCombineSoloI` stampa soltanto la prima voce e la contrassegna come un "Solo".
- `\partCombineSoloII` stampa soltanto la seconda voce e la contrassegna come un "Solo".
- `\partCombineAutomatic` termina le funzioni dei comandi precedenti e ripristina il funzionamento predefinito di `\partCombine`.

```
instrumentOne = \relative c' {
  \partCombineApart c2^"separato" e |
  \partCombineAutomatic e2^"automatico" e |
  \partCombineChords e'2^"accordo" e |
  \partCombineAutomatic c2^"automatico" c |
  \partCombineApart c2^"separato" \once \partCombineChords e^"accordo una volta sola"
  c2 c |
}
```

```
instrumentTwo = \relative {
  c'2 c |
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partCombine \instrumentOne \instrumentTwo }
>>
```




### *Uso di \partCombine col testo vocale*

Il comando `\partCombine` non è progettato per funzionare col testo vocale; al punto che se una delle voci è nominata in modo esplicito per poterle assegnare del testo, l'unione delle parti smette di funzionare. Tuttavia, questo risultato si può ottenere usando un contesto `NullVoice`. Vedi [Polifonia con testo in comune], pagina 297.

## Frammenti di codice selezionati

*Combinare due parti sullo stesso rigo*

Lo strumento di unione delle parti (il comando `\partCombine`) permette di combinare varie parti sullo stesso rigo. Indicazioni testuali come "solo" e "a2" sono aggiunte automaticamente; per toglierele basta impostare la proprietà `printPartCombineTexts` su `f`. Per le partiture vocali (inni), non c'è bisogno di aggiungere i testi "solo/a2", quindi dovrebbero essere disattivati. Tuttavia potrebbe convenire non usarlo se c'è una qualche parte solista, perché non verrebbe indicata. In tali casi è preferibile usare la notazione polifonica normale.

Questo frammento illustra i tre modi con cui due parti possono essere stampate su uno stesso rigo: normale polifonia, `\partCombine` senza testo e `\partCombine` con testo.

```
%% Combining pedal notes with clef changes

musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
```

```
  c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

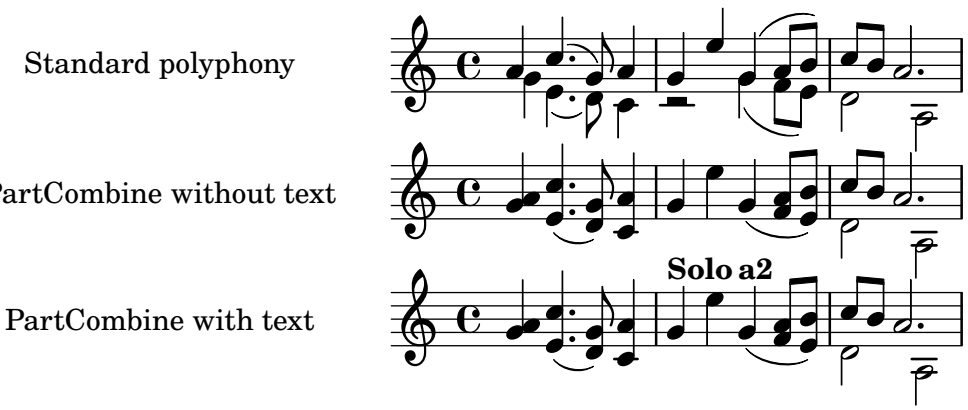
\score {
    <<
    \new Staff \with { instrumentName = "Standard polyphony" }

      << \musicUp \\ \musicDown >>

    \new Staff \with {
      instrumentName = "PartCombine without text"
      printPartCombineTexts = ##f
    }

    \partCombine \musicUp \musicDown

    \new Staff \with { instrumentName = "PartCombine with text" }
      \partCombine \musicUp \musicDown
    >>
  \layout {
    indent = 6.0\cm
    \context {
      \Score
      \override SystemStartBar.collapse-height = #30
    }
  }
}
```
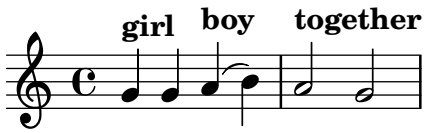



*Modificare le indicazioni testuali di partCombine*

Quando si usa la funzionalità di combinazione automatica delle parti, si può modificare il testo delle sezioni soliste e dell'unisono:

```
\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
    }
>>
```




## Vedi anche

Glossario Musicale: Sezione “a due” in *Glossario Musicale*, Sezione “parte” in *Glossario Musicale*.

Guida alla notazione: Sezione 1.6.3 [Scrittura delle parti], pagina 210.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “PartCombineMusic” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Tutte le funzioni `\partCombine...` possono accettare soltanto due voci.

Le funzioni `\partCombine...` non possono essere inserite all’interno di un blocco `\tuplet` o `\relative`.

Se `printPartCombineTexts` è attivo e le due voci eseguono le stesse note “in modo discontinuo” nella stessa misura, potrebbe apparire il testo `a2` più di una volta in quella misura.

`\partCombine` sa soltanto quando una nota inizia in una voce (`Voice`); non può, ad esempio, ricordare se una nota in una voce è già iniziata quando combina le note già iniziate nell’altra voce. Questo può portare a vari problemi inattesi, tra cui la stampa non corretta dei segni “Solo” e “Unisono”.

`\partCombine` tiene tutti gli estensori (legature di portamento e di valore, forcelle, etc.) nella stessa voce, quindi se uno di questi estensori inizia o termina in una voce diversa potrebbe essere stampato incorrettamente o non essere stampato affatto.

Se la funzione `\partCombine` non riesce a combinare le due espressioni musicali (ovvero quando le due voci hanno durate diverse), assegnerà alle voci, internamente, nomi personalizzati: rispettivamente `one` e `two`. Ciò significa che se c’è un “passaggio” a un contesto `Voice` nominato diversamente, gli eventi in quel contesto verranno ignorati.

Consultare i *Problemi noti e avvertimenti* in [Intavolature predefinite], pagina 350, se si usa `\partCombine` con l’intavolatura, e la *Nota* in [Travature automatiche], pagina 85, se si usa la disposizione automatica delle travature.

## Scrivere la musica in parallelo
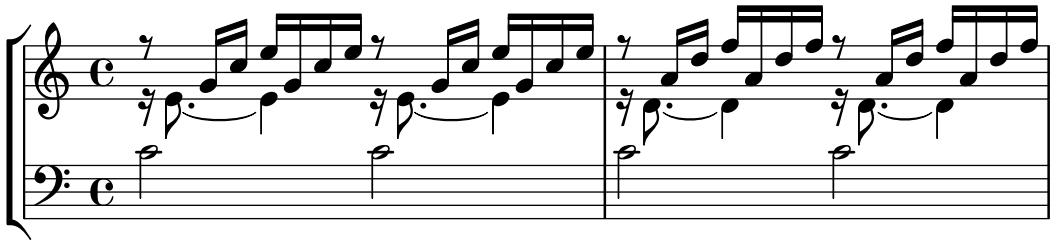
La musica che contiene parti diverse può essere messa in parallelo nel codice di input. La funzione `\parallelMusic` accetta una lista contenente i nomi di un insieme di variabili da creare e un'espressione musicale. Il contenuto delle misure alternate nell'espressione diventa il valore delle rispettive variabili, in modo che possano essere usate successivamente per stampare la musica.

> **Nota:** L'uso dei controlli di battuta `|` è obbligatorio e le misure devono avere la stessa durata.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Battuta 1
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~   4              r16 e'8.~   4              |
  c'2                        c'2                        |

  % Battuta 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
  r16 d'8.~   4              r16 d'8.~   4              |
  c'2                        c'2                        |

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



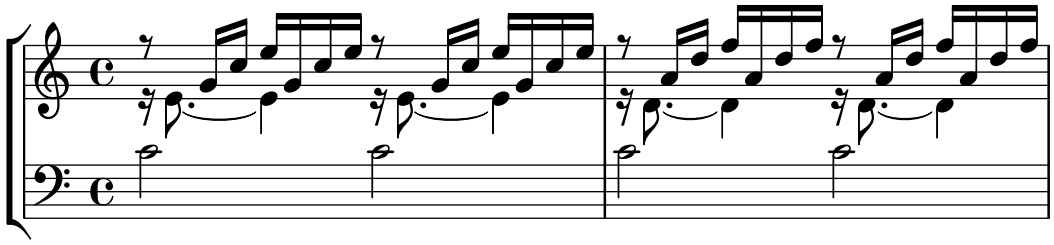
L'uso del modo relativo è permesso. Attenzione: il comando `\relative` non deve essere messo dentro `\parallelMusic`. Le note sono relative alla nota precedente della voce, non a quella precedente nell'input. In altre parole, le note relative di `voiceA` ignorano le note in `voiceB`.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Battuta 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ 4        r16 e8.~  4         |
  c2                c                   |

  % Battuta 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  4        r16 d8.~  4        |
  c2                 c                  |

 }
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
```

```
  \new Staff \relative c' { \clef bass \voiceC }
>>
```



Questo è molto utile nella musica per pianoforte. Questo esempio combina sezioni di quattro battute consecutive con quattro variabili:

```
global = {
  \key g \major
  \time 2/4
}
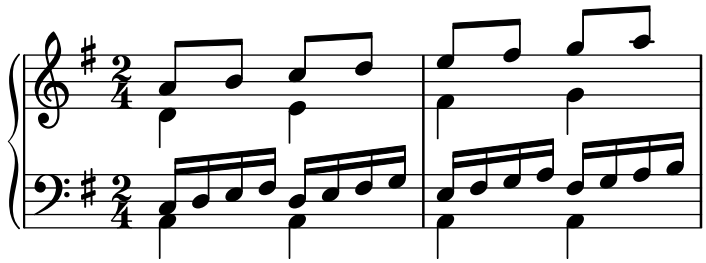
\parallelMusic voiceA,voiceB,voiceC,voiceD {
  % Battuta 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g |
  a4          a         |

  % Battuta 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
       >>
     }
```
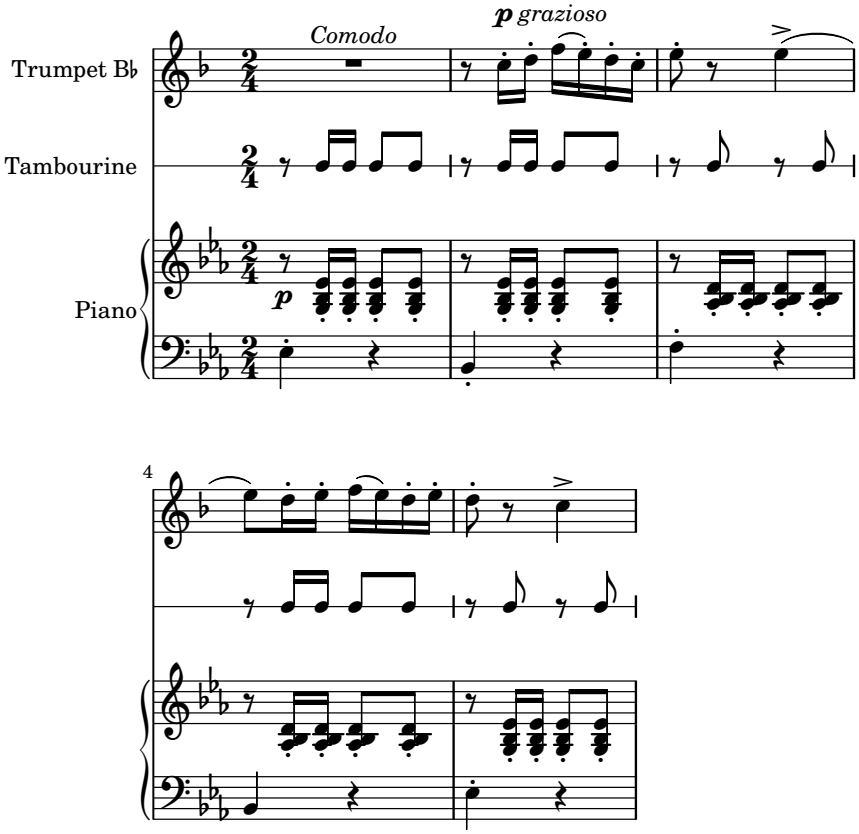
```
  >>
}
```



### Vedi anche

Manuale d'apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

## 1.6 Notazione del rigo




Questa sezione spiega come modificare l'aspetto del rigo, come stampare partiture multirigo e come aggiungere indicazioni di tempo e citazioni in corpo più piccolo nel rigo.

### 1.6.1 Aspetto del rigo
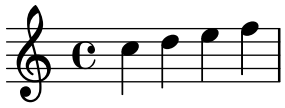
Questa sezione presenta i diversi metodi per creare e raggruppare i righi.

### Istanziare nuovi righi

Il *rigo musicale* si crea con i comandi `\new` o `\context`. Ulteriori dettagli in Sezione 5.1.2 [Creazione e citazione di un contesto], pagina 594.

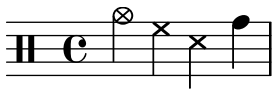
Il contesto di base del rigo è `Staff`:

```
\new Staff \relative { c''4 d e f }
```



Il contesto `DrumStaff` crea un rigo di cinque linee impostato per una tipica batteria. Ogni strumento viene mostrato con un simbolo diverso. Gli strumenti si inseriscono nella modalità percussioni, che si attiva col comando `\drummode`: ogni strumento viene indicato con un nome. Ulteriori dettagli in [Righi delle percussioni], pagina 400.

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



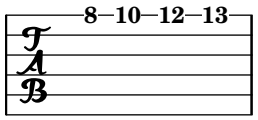
`RhythmicStaff` crea un rigo con una sola linea che mostra soltanto i valori ritmici dell'input. Le durate reali vengono mantenute. Ulteriori dettagli in [Mostrare i ritmi della melodia], pagina 83.

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



`TabStaff` crea un'intavolatura (o tablatura) con sei corde nell'accordatura standard per chitarra. Ulteriori dettagli in [Intavolature predefinite], pagina 350.
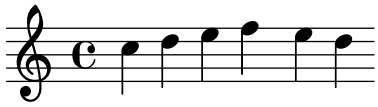
```
\new TabStaff \relative { c''4 d e f }
```




Ci sono due contesti del rigo specifici per la notazione di musica antica, `MensuralStaff` e `VaticanaStaff`, descritti in [Contesti predefiniti], pagina 442.

Il contesto `GregorianTranscriptionStaff` crea un rigo per il canto gregoriano moderno. Non mostra le stanghette delle battute.

```
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }
```



Si possono creare nuovi contesti per un singolo rigo, come è spiegato dettagliatamente in Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

## Vedi anche

Glossario musicale: Sezione “rigo” in *Glossario Musicale*,

Guida alla notazione: Sezione 5.1.2 [Creazione e citazione di un contesto], pagina 594, [Righi delle percussioni], pagina 400, [Mostrare i ritmi della melodia], pagina 83, [Intavolature predefinite], pagina 350, [Contesti predefiniti], pagina 442, [Simbolo del rigo], pagina 201, [Contesti del canto gregoriano], pagina 452, [Contesti mensurali], pagina 444, Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

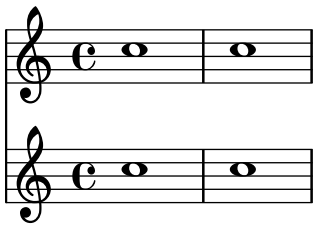
Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “GregorianTranscriptionStaff” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “TabStaff” in *Guida al Funzionamento Interno*, Sezione “MensuralStaff” in *Guida al Funzionamento Interno*, Sezione “VaticanaStaff” in *Guida al Funzionamento Interno*, Sezione “StaffSymbol” in *Guida al Funzionamento Interno*.

## Raggruppare i righi

Esistono vari contesti per raggruppare insieme singoli righi in modo da formare sistemi multirigo. Ogni contesto di raggruppamento imposta il comportamento delle stanghette e lo stile del segno che delimita l’inizio del sistema.

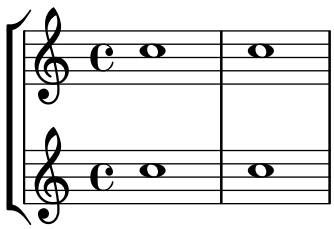
Se non si specifica alcun contesto, vengono usate le proprietà predefinite: il gruppo inizia con una linea verticale e le stanghette non sono collegate.

```
<<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Nel contesto `StaffGroup`, il gruppo inizia con una parentesi quadra e le stanghette attraversano tutti i righi.

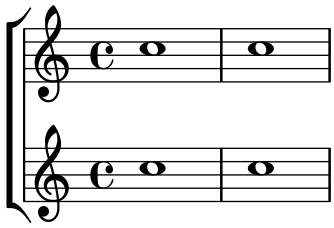
```
\new StaffGroup <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Nel contesto `ChoirStaff`, il gruppo inizia con una parentesi quadra, ma le stanghette non sono collegate.

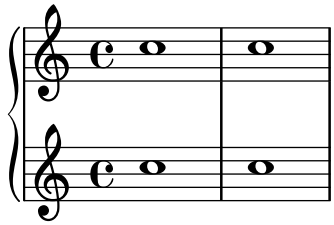
```
\new ChoirStaff <<
```

```
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



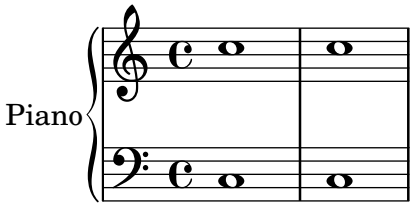
Nel contesto `GrandStaff`, il gruppo inizia con una parentesi graffa e le stanghette sono collegate da rigo a rigo.

```
\new GrandStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Il contesto `PianoStaff` è identico a `GrandStaff`, con l'unica differenza che permette di mostrare il nome dello strumento direttamente. Ulteriori dettagli in [Nomi degli strumenti], pagina 210.

```
\new PianoStaff \with { instrumentName = "Piano" }
<<
  \new Staff \relative { c''1 c }
  \new Staff \relative { \clef bass c1 c }
>>
```




Ogni contesto per il gruppo di righi imposta la proprietà `systemStartDelimiter` su uno dei seguenti valori: `SystemStartBar`, `SystemStartBrace` o `SystemStartBracket`. È presente anche un quarto delimitatore, `SystemStartSquare`, ma deve essere indicato esplicitamente.

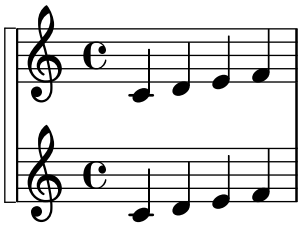
Si possono definire nuovi contesti di gruppi di rigo. I dettagli sono spiegati in Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

## Frammenti di codice selezionati

*Usare una parentesi quadra all'inizio di un gruppo di righi*

Si può usare il segno `SystemStartSquare` (uno dei segni che delimitano l'inizio del sistema) impostandolo esplicitamente in un contesto `StaffGroup` o `ChoirStaff`.

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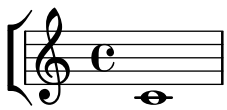


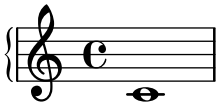
*Mostrare la parentesi anche se c'è un solo rigo nel sistema*

Se c'è un solo rigo in uno dei tipi di rigo `ChoirStaff` o `StaffGroup`, la parentesi e la stanghetta iniziale non appaiono. Si può modificare questo comportamento predefinito sovrascrivendo `collapse-height` e impostando un valore inferiore al numero di linee del rigo.

Nei contesti `PianoStaff` e `GrandStaff`, dove i sistemi iniziano con una parentesi graffa invece di una parentesi quadra, occorre impostare un'altra proprietà, come si vede nel secondo sistema dell'esempio.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
```



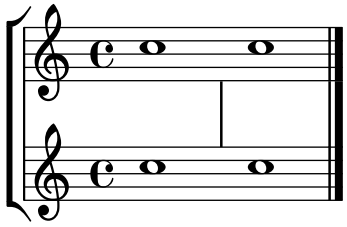


*Formattazione mensurale (stanghette tra i righi)*

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta con `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



## Vedi anche

Glossario musicale: Sezione “graffa” in *Glossario Musicale*, Sezione “parentesi quadra” in *Glossario Musicale*, Sezione “accollatura” in *Glossario Musicale*.

Guida alla notazione: [Nomi degli strumenti], pagina 210, Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.
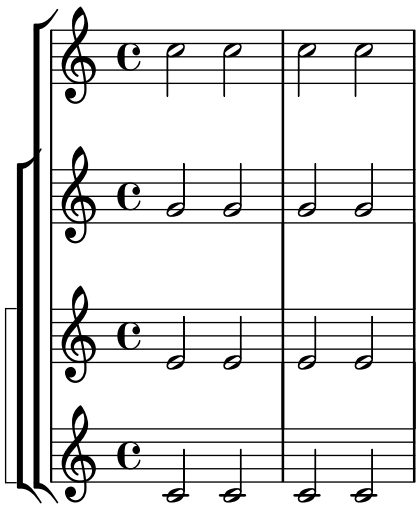
Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “GrandStaff” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento Interno*, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.

## Gruppi di righi annidati

I contesti dei gruppi di righi possono essere annidati fino a qualsiasi livello. In questo caso, ogni contesto inferiore crea una nuova parentesi accanto alla parentesi del gruppo superiore.

```
\new StaffGroup <<
  \new Staff \relative { c''2 c | c2 c }
  \new StaffGroup <<
    \new Staff \relative { g'2 g | g2 g }
    \new StaffGroup \with {
```

```
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff \relative { e'2 e | e2 e }
      \new Staff \relative { c'2 c | c2 c }
    >>
  >>
>>
```



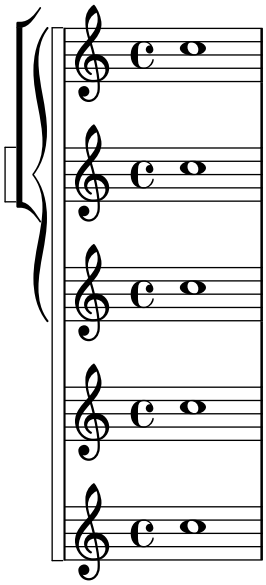
Si possono definire nuovi gruppi di righi annidati. Ulteriori dettagli in Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

## Frammenti di codice selezionati

*Annidare i righi*

Si può usare la proprietà `systemStartDelimiterHierarchy` per creare gruppi di righi annidati più complessi. Il comando `\set StaffGroup.systemStartDelimiterHierarchy` prende come argomento una lista alfabetica dell'insieme di righi prodotti. Prima di ogni rigo si può assegnare un delimitatore di inizio del sistema. Deve essere racchiuso tra parentesi e collega tutti i righi compresi tra le parentesi. Gli elementi nella lista possono essere omessi, ma la prima parentesi quadra collega sempre tutti i righi. Le possibilità sono `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` e `SystemStartSquare`.

```
\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = #4
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                            (SystemStartSquare b)  ) c ) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



## Vedi anche

Guida alla notazione: [Raggruppare i righi], pagina 195, [Nomi degli strumenti], pagina 210, Sezione 5.1.6 [Definizione di nuovi contesti], pagina 607.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

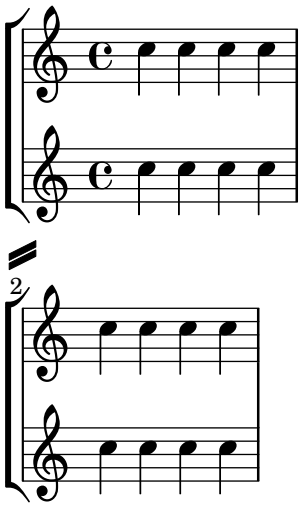
Guida al funzionamento interno: Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento Interno*, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.

## Separare i sistemi

Se il numero di sistemi per pagina cambia di pagina in pagina, è consuetudine separare i sistemi con un segno separatore. Per impostazione predefinita questo segno è disattivo, ma può essere attivato con un’opzione in `\paper`.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
```

```
    % i seguenti comandi servono soltanto alla formattazione di questa documentazione
    paper-width = 100\mm
    paper-height = 100\mm
    tagline = ##f
  }
}
```



## Vedi anche

Guida alla notazione: Sezione 4.1 [Formattazione della pagina], pagina 539.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.
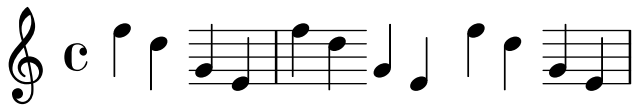
### 1.6.2 Modificare singoli righi

Questa sezione spiega come modificare gli attributi specifici di un rigo, per esempio il numero di linee o la dimensione del rigo. Vengono descritti anche i metodi per iniziare e finire un rigo e per impostare le sezioni ossia.

## Simbolo del rigo

I comandi `\stopStaff` e `\startStaff` servono a fermare o (ri)avviare le linee del rigo, per impedire che appaiano in un punto della partitura.

```
\relative {
  \stopStaff f''4 d \startStaff g, e
  f'4 d \stopStaff g, e
  f'4 d \startStaff g, e
}
```



## Comandi predefiniti

`\startStaff`, `\stopStaff`. Le linee di un rigo appartengono all’oggetto `StaffSymbol` (che comprende i tagli addizionali) e si possono modificare tramite le proprietà di `StaffSymbol`; però queste modifiche devono essere fatte prima che il rigo sia (ri)avviato.

Si può cambiare il numero di linee del rigo:
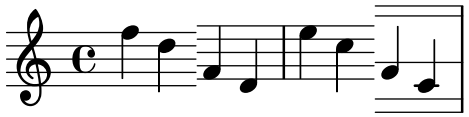
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-count = #2
  \startStaff g, e |

  f'4 d \stopStaff
  \revert Staff.StaffSymbol.line-count
  \startStaff g, e |
}
```



Si può cambiare anche la posizione di ogni linea del rigo. Un elenco di numeri definisce la posizione di ogni linea. I valori consueti sono `0` per la linea centrale e `(-4 -2 0 2 4)` per le altre. La linea del rigo appare solo se è presente il suo valore, quindi questo comando permette di variare anche il numero delle linee, oltre alla loro posizione.
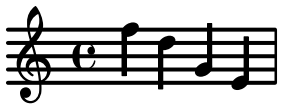
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
  \startStaff g, e |
  f'4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
  \startStaff g, e |
}
```



Per conservare le tipiche direzioni dei gambi (nella metà inferiore del rigo i gambi puntano in su, mentre in quella superiore sono rivolti in giù), occorre allineare la linea centrale (o lo spazio) del rigo personalizzato alla posizione della linea centrale normale (0). Potrà essere necessario regolare la posizione della chiave e del Do centrale per adattarsi alle nuove linee. Si veda [Chiave], pagina 17.

Si può modificare lo spessore della linea del rigo. Per impostazione predefinita, questa modifica ha effetto anche sui tagli addizionali e sui gambi.
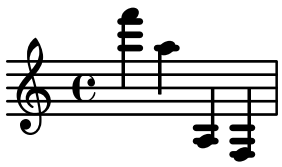
```
\new Staff \with {
  \override StaffSymbol.thickness = #3
} \relative {
  f''4 d g, e
}
```



È anche possibile impostare lo spessore dei tagli addizionali in modo indipendente dalle linee del rigo.

```
\new Staff \with {
```

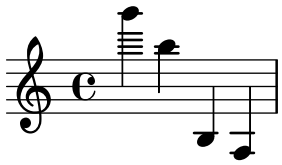
```
  \override StaffSymbol.thickness = #2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f'''4 a, a,, f
}
```



Il primo valore viene moltiplicato per lo spessore della linea del rigo, il secondo per la spaziatura del rigo; la somma dei due valori definisce il nuovo valore dello spessore del taglio addizionale.
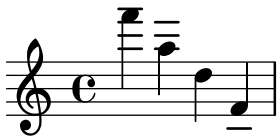
Si possono modificare le posizioni verticali dei tagli addizionali:

```
\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f'''4 a, a,, f
}
```
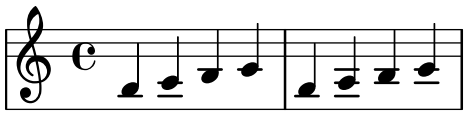


Si possono far apparire ulteriori tagli addizionali sopra o sotto le teste delle note, a seconda della posizione corrente relativa alle altre teste, anch'esse con i propri tagli addizionali.

```
\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
} \relative {
  f'''4 a, d, f,
}
```



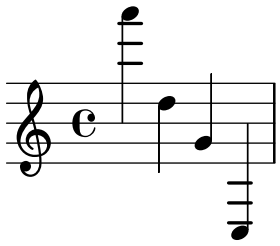
Si possono far apparire i tagli addizionali anche dentro il rigo quando servono delle linee personalizzate. L'esempio mostra la posizione predefinita dei tagli addizionali quando la proprietà `ledger-position` è impostata e quando non lo è. Nell'esempio il comando `\stopStaff` serve ad annullare il comando `\override` per l'oggetto `StaffSymbol`.

```
\relative d' {
  \override Staff.StaffSymbol.line-positions = #'(-8 0 2 4)
  d4 e f g
  \stopStaff
  \startStaff
  \override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
  d4 e f g
}
```



Si può cambiare la distanza tra le linee del rigo. Tale modifica ha effetto anche sulla spaziatura della linea.

```
\new Staff \with {
  \override StaffSymbol.staff-space = #1.5
} \relative {
  f'''4 d, g, e,
}
```
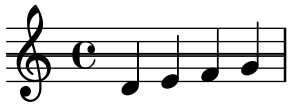


## Frammenti di codice selezionati

*Rendere alcune linee del rigo più spesse delle altre*

In ambito didattico può essere utile rendere più spesso una linea del rigo (per esempio, la linea centrale, o per sottolineare la linea della chiave di Sol). Per farlo si possono aggiungere altre linee e posizionarle molto vicino alla linea che deve essere evidenziata, usando la proprietà `line-positions` dell'oggetto `StaffSymbol`.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



## Vedi anche

Glossario musicale: Sezione "linea" in *Glossario Musicale*, Sezione "taglio addizionale" in *Glossario Musicale*, Sezione "rigo (o pentagramma)" in *Glossario Musicale*.

Guida alla notazione: [Chiave], pagina 17.

Frammenti: Sezione "Staff notation" in *Frammenti di codice*.
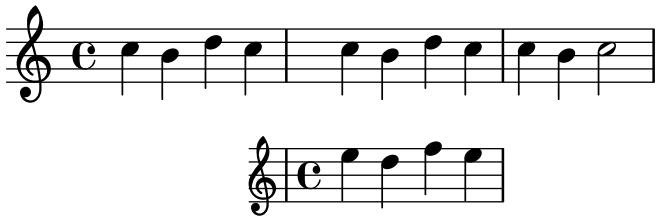
Guida al funzionamento interno: Sezione "StaffSymbol" in *Guida al Funzionamento Interno*, Sezione "staff-symbol-interface" in *Guida al Funzionamento Interno*.

## Righi ossia

I righi *ossia* si possono creare aggiungendo un nuovo rigo simultaneo nel punto giusto:

```
\new Staff \relative {
  c''4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
```

```
  >>
  c4 b c2
}
```

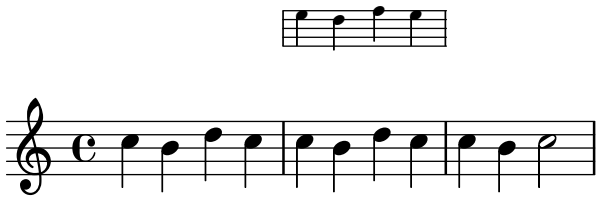


Tuttavia, questo esempio non produce quel che normalmente si desidera. Per creare righi ossia che siano sopra il rigo originale, non abbiano indicazione di tempo né chiave e abbiano un tipo di carattere più piccolo, sono necessarie delle modifiche manuali. Il Manuale d'apprendimento descrive una tecnica specifica per ottenere questo risultato, a partire da Sezione "Annidare le espressioni musicali" in *Manuale di Apprendimento*.

L'esempio seguente usa la proprietà `alignAboveContext` per allineare il rigo ossia. Questo metodo conviene quando sono necessari solo pochi righi ossia.

```
\new Staff = "main" \relative {
  c''4 b d c
  <<
    { c4 b d c }
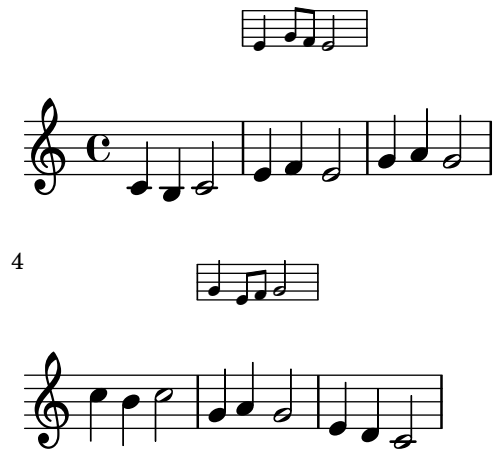
    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = "main"
      \magnifyStaff #2/3
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```



Se si hanno molti righi ossia isolati, è meglio creare un contesto `Staff` vuoto con un *identificativo del contesto* specifico; i righi ossia possono essere creati *chiamando* questo contesto e usando `\startStaff` e `\stopStaff` nei punti richiesti. I vantaggi di questo metodo sono più evidenti se il brano è più lungo del seguente esempio.

```
<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    \magnifyStaff #2/3
  }
  { \stopStaff s1*6 }
```
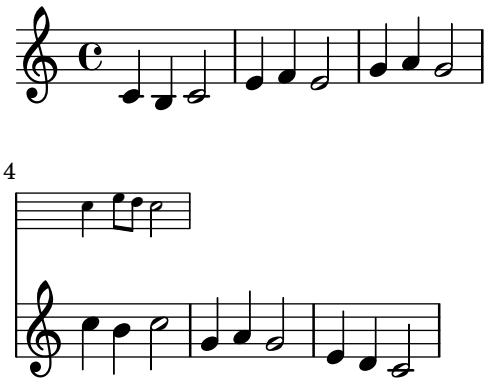
```
  \new Staff \relative {
    c'4 b c2
    <<
      { e4 f e2 }
      \context Staff = "ossia" {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = "ossia" {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```



Altrimenti si può usare il comando `\RemoveAllEmptyStaves` per creare i righi ossia. Questo metodo conviene quando i righi ossia si trovano subito dopo un'interruzione di linea. Ulteriori informazioni su `\RemoveAllEmptyStaves` si trovano in [Nascondere i righi], pagina 208.

```
<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    \magnifyStaff #2/3
    \RemoveAllEmptyStaves
  } \relative {
    R1*3
    c''4 e8 d c2
  }
  \new Staff \relative {
    c'4 b c2
    e4 f e2
    g4 a g2 \break
```

```
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>
```
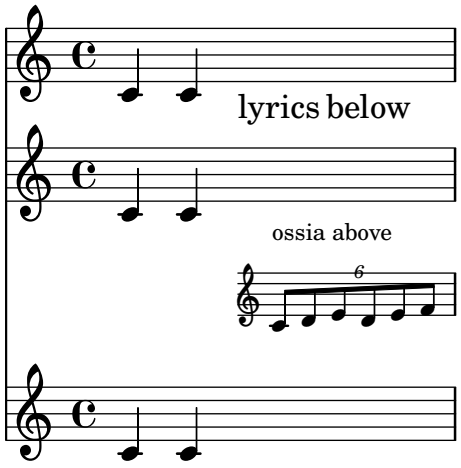


## Frammenti di codice selezionati

*Allineare verticalmente gli ossia e il testo vocale*

Questo frammento mostra come usare le proprietà di contesto `alignBelowContext` e `alignAboveContext` per controllare il posizionamento del testo vocale e degli ossia.

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

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3
          c8[^"ossia above" d e d e f]
        }
      }
    >>
  }
>>
```




## Vedi anche

Glossario musicale: Sezione “ossia” in *Glossario Musicale*, Sezione “rigo (pentagramma)” in *Glossario Musicale*, Sezione “rigo temporaneo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Annidare le espressioni musicali” in *Manuale di Apprendimento*, Sezione “Dimensione degli oggetti” in *Manuale di Apprendimento*, Sezione “Lunghezza e spessore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: [Nascondere i righi], pagina 208.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StaffSymbol” in *Guida al Funzionamento Interno*.

## Nascondere i righi

Le linee del rigo si possono nascondere togliendo l’incisore `Staff_symbol_engraver` dal contesto `Staff`. Altrimenti si può usare `\stopStaff`.

```
\new Staff \with {
  \remove "Staff_symbol_engraver"
}
\relative { a''8 f e16 d c b a2 }
```

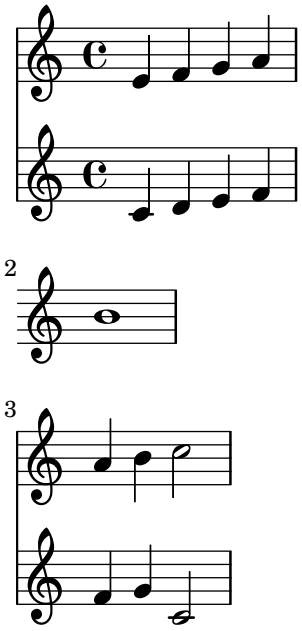


I righi vuoti si possono nascondere (per la cosiddetta ‘partitura alla francese’) applicando il comando `\RemoveEmptyStaves` a un contesto, che può essere fatto globalmente (in un blocco `\layout`) oppure soltanto per righi specifici (in un blocco `\with`). Questo comando toglie tutti i righi vuoti di una partitura eccetto quelli nel primo sistema. Per nascondere anche quelli del primo sistema usare il comando `\RemoveAllEmptyStaves`. I contesti supportati sono `Staff`, `RhythmicStaff` e `VaticanaStaff`.

> **Nota:** Un rigo viene considerato vuoto quando contiene soltanto pause multiple, pause, salti, pause spaziatrici o una combinazione di questi elementi.

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

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



`\RemoveAllEmptyStaves` si può usare anche per creare sezioni ossia per un rigo. I dettagli si trovano in [Righi ossia], pagina 204.

## Comandi predefiniti

`\RemoveEmptyStaves`, `\RemoveAllEmptyStaves`.

## Vedi anche

Glossario musicale: Sezione “rigo temporaneo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.5 [Modifica delle impostazioni predefinite di un contesto], pagina 602, [Simbolo del rigo], pagina 201, [Righi ossia], pagina 204, [Note nascoste], pagina 229, [Pause invisibili], pagina 61, Sezione 5.4.7 [Visibilità degli oggetti], pagina 638.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “Staff_symbol_engraver” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Se si toglie l'incisore `Staff_symbol_engraver` vengono nascoste anche le stanghette. Se si forza la visibilità delle stanghette, potrebbero verificarsi degli errori di formattazione. In questo caso, conviene usare i seguenti comandi invece di togliere l'incisore:

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

Per i problemi noti e gli avvertimenti relativi a `\Staff` `\RemoveEmptyStaves` si veda Sezione 5.1.5 [Modifica delle impostazioni predefinite di un contesto], pagina 602.
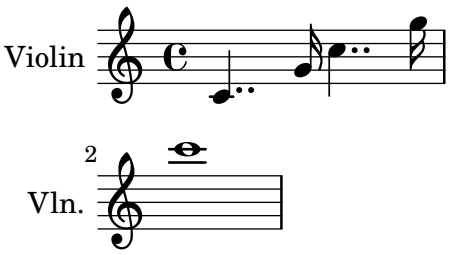
## 1.6.3 Scrittura delle parti

Questa sezione spiega come inserire in una partitura le indicazioni di tempo e i nomi degli strumenti. Mostra anche come citare altre voci e come formattare le citazioni in corpo più piccolo.

### Nomi degli strumenti
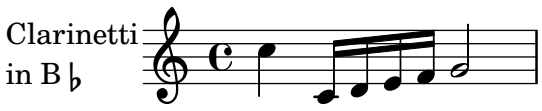
I nomi degli strumenti possono essere fatti apparire, alla sinistra dei righi, nei contesti `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` e `ChoirStaff`. Il valore di `instrumentName` viene usato per il primo rigo e quello di `shortInstrumentName` per tutti i righi successivi.

```
\new Staff \with {
  instrumentName = "Violin "
  shortInstrumentName = "Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}
```




Si può usare `\markup` per creare nomi più complessi:
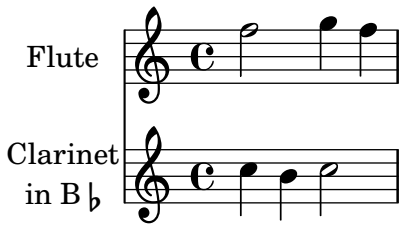
```
\new Staff \with {
  instrumentName = \markup {
    \column { "Clarinetti"
      \line { "in B" \smaller \flat }
    }
  }
} \relative {
  c''4 c,16 d e f g2
}
```




Se due o più contesti del rigo sono raggruppati insieme, i nomi degli strumenti, sia quello normale che quello abbreviato, vengono centrati automaticamente. Per allineare al centro i nomi degli strumenti che vanno a capo, occorre usare `\center-column`:
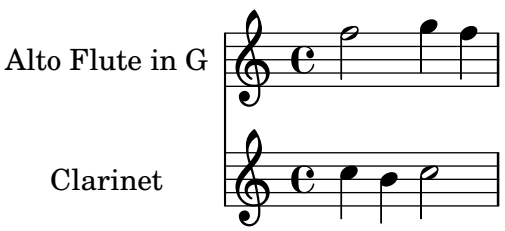
```
<<
```

```
  \new Staff \with {
    instrumentName = "Flute"
  } \relative {
    f''2 g4 f
}
  \new Staff \with {
    instrumentName = \markup {
      \center-column { "Clarinet"
        \line { "in B" \smaller \flat }
      }
    }
  } \relative { c''4 b c2 }
>>
```

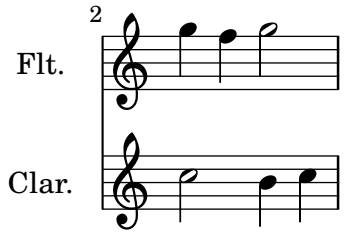



Tuttavia, se i nomi degli strumenti sono lunghi, potranno essere centrati solo aumentando i valori di `indent` e `short-indent`. Ulteriori dettagli su queste impostazioni si trovano in [Variabili \paper per spostamenti e indentazioni], pagina 547.

```
<<
  \new Staff \with {
    instrumentName = "Alto Flute in G"
    shortInstrumentName = "Flt."
  } \relative {
    f''2 g4 f \break
    g4 f g2
  }
  \new Staff \with {
    instrumentName = "Clarinet"
    shortInstrumentName = "Clar."
  } \relative {
    c''4 b c2 \break
    c2 b4 c
  }
>>

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






Per impostare i nomi degli strumenti in altri contesti (come `ChordNames` o `FiguredBass`), si deve aggiungere l'incisore `Instrument_name_engraver` a quel contesto. Ulteriori dettagli in Sezione 5.1.4 [Modifica dei componenti aggiuntivi di un contesto], pagina 600.

`shortInstrumentName` può essere cambiato all'interno di un brano, insieme a altre impostazioni necessarie al nuovo strumento. Tuttavia, di `instrumentName` apparirà solo la prima definizione e le modifiche successive saranno ignorate:

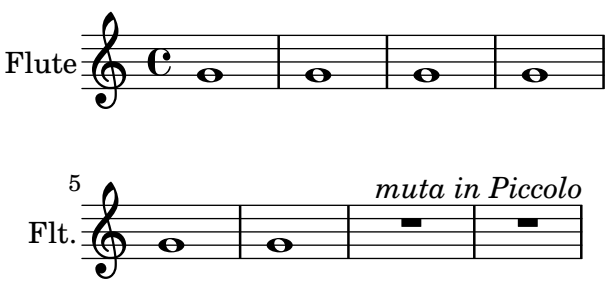
```
prepPiccolo = <>^\markup \italic { muta in Piccolo }

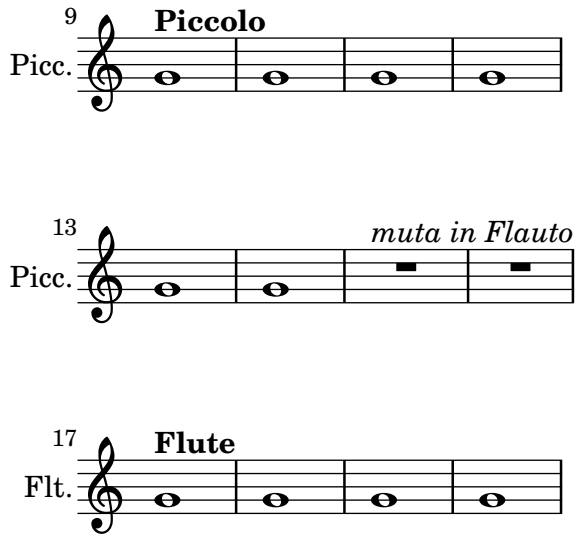
prepFlute = <>^\markup \italic { muta in Flauto }

setPiccolo = {
  <>^\markup \bold { Piccolo }
  \transposition c''
}

setFlute = {
  <>^\markup \bold { Flute }
  \transposition c'
}

\new Staff \with {
  instrumentName = "Flute"
  shortInstrumentName = "Flt."
}
\relative {
  g'1 g g g \break
  g1 g \prepPiccolo R R \break
  \set Staff.instrumentName = "Piccolo"
  \set Staff.shortInstrumentName = "Picc."
  \setPiccolo
  g1 g g g \break
  g1 g \prepFlute R R \break
  \set Staff.instrumentName = "Flute"
  \set Staff.shortInstrumentName = "Flt."
  \setFlute
  g1 g g g
}
```






## Vedi anche

Guida alla notazione: [Variabili `\paper` per spostamenti e indentazioni], pagina 547, Sezione 5.1.4 [Modifica dei componenti aggiuntivi di un contesto], pagina 600.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “InstrumentName” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## Citare altre voci

È molto comune che una voce usi le stesse note di un’altra voce. Per esempio, il primo e il secondo violino che suonano la stessa frase durante un particolare passaggio del brano. Per evitare di reinserire la musica di nuovo per la seconda voce, si può far sì che una voce *citi* l’altra.

Il comando `\addQuote`, usato nell’ambito di più alto livello, definisce un flusso musicale da cui poter citare i frammenti.
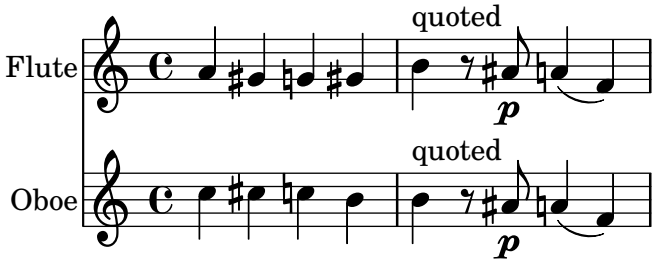
Il comando `\quoteDuring` serve a indicare il punto in cui inizia la citazione. È seguito da due argomenti: il nome della voce citata, come è definito da `\addQuote`, e un’espressione musicale per la durata della citazione.

```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
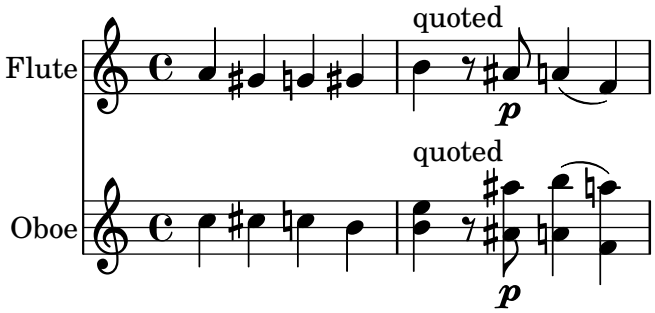



Se l'espressione musicale usata in `\quoteDuring` contiene note invece di pause spaziatrici o multiple, la citazione apparirà in forma polifonica e potrebbe causare risultati indesiderati.

```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { e4 r8 ais b4 a }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```




Se un comando `\unfoldRepeats` in un'espressione musicale deve essere stampato quando si usa `\quoteDuring`, allora anch'esso deve contenere il suo comando `\unfoldRepeats`;

```
fluteNotes = \relative {
  \repeat volta 2 { a'4 gis g gis }
}

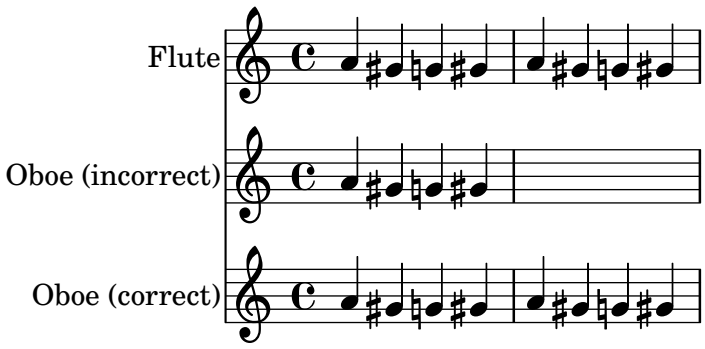
oboeNotesDW = \relative {
  \repeat volta 2 \quoteDuring "incorrect" { s1 }
}

oboeNotesW = \relative {
  \repeat volta 2 \quoteDuring "correct" { s1 }
}


\addQuote "incorrect" { \fluteNotes }
```

```
\addQuote "correct" { \unfoldRepeats \fluteNotes }

\score {
  \unfoldRepeats
  <<
    \new Staff \with { instrumentName = "Flute" }
    \fluteNotes
    \new Staff \with { instrumentName = "Oboe (incorrect)" }
    \oboeNotesDW
    \new Staff \with { instrumentName = "Oboe (correct)" }
    \oboeNotesW
  >>
}
```
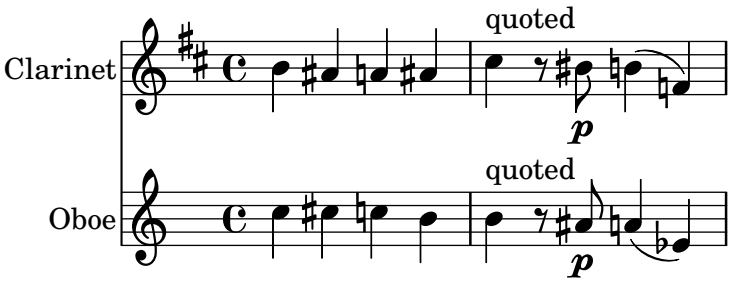



Il comando `\quoteDuring` usa le impostazioni `\transposition` sia della parte citata sia di quella che cita per produrre delle note per la parte che cita che abbiano la stessa altezza di quelle nella parte citata.

```
clarinetNotes = \relative c'' {
  \transposition bes
  \key d \major
  b4 ais a ais | cis4^"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "clarinet" { s1 }
}

\addQuote "clarinet" { \clarinetNotes }


\score {
  <<
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
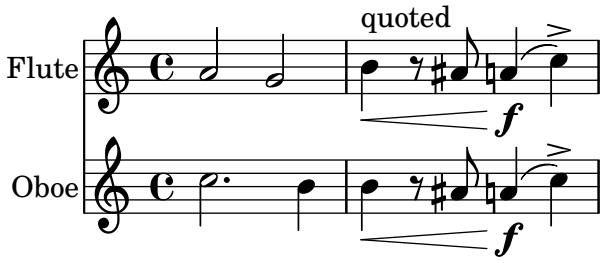



La musica citata include tutte le articolazioni, dinamiche, annotazioni, etc. presenti nel frammento citato. È possibile scegliere quali di questi oggetti far apparire usando la proprietà di contesto `quotedEventTypes`.

```
fluteNotes = \relative {
  a'2 g2 |
  b4\<^"quoted" r8 ais a4\f( c->)
 }

oboeNotes = \relative {
  c''2. b4 |
  \quoteDuring "flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \set Score.quotedEventTypes = #'(note-event articulation-event
                                     crescendo-event rest-event
                                     slur-event dynamic-event)
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
 }
```




Le citazioni possono anche essere contrassegnate; si veda [Uso delle etichette], pagina 515.

## Vedi anche

Guida alla notazione: [Trasporto strumentale], pagina 27, [Uso delle etichette], pagina 515.

File installati: `scm/define-event-classes.scm`.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Music classes” in *Guida al Funzionamento Interno*, Sezione “QuoteMusic” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.
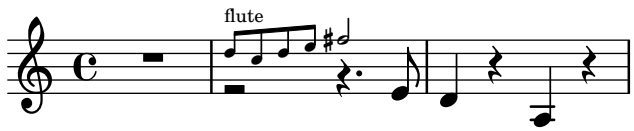
## Problemi noti e avvertimenti

Solo il contenuto della prima voce in un comando `\addQuote` sarà preso in considerazione per la citazione; quindi se l'espressione musicale contiene comandi `\new` o `\context Voice`, il loro contenuto non verrà citato. La citazione degli abbellimenti non è supportata e potrebbe causare il crash di LilyPond; la citazione di terzine annidate potrebbe produrre una notazione mediocre.

## Formattazione delle notine
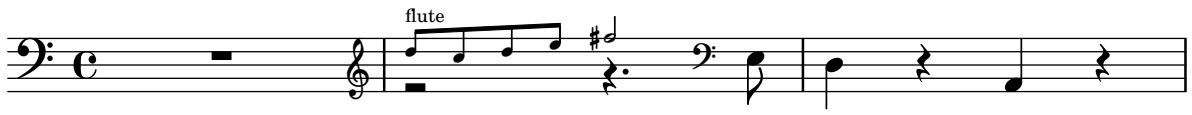
Il modo più semplice per formattare le notine è creare esplicitamente un contesto `CueVoice` all'interno della parte.

```
\relative {
  R1
  <<
    { e'2\rest r4. e8 }
    \new CueVoice {
      \stemUp d'8^"flute" c d e fis2
    }
  >>
  d,4 r a r
}
```




Si può usare il comando `\cueClef` all'interno di un contesto `CueVoice` esplicito se è richiesto un cambiamento di chiave; in questo modo la chiave apparirà nella dimensione giusta per le notine. Si può poi usare il comando `\cueClefUnset` per tornare alla chiave originale, di nuovo nella dimensione giusta.
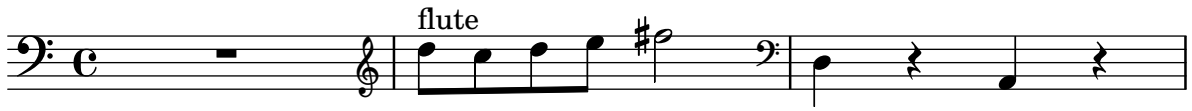
```
\relative {
  \clef "bass"
  R1
  <<
    { e'2\rest r4. \cueClefUnset e,8 }
    \new CueVoice {
      \cueClef "treble" \stemUp d''8^"flute" c d e fis2
    }
  >>
  d,,4 r a r
}
```




I comandi `\cueClef` e `\cueClefUnset` si possono usare anche senza un esplicito contesto `CueVoice`.

```
\relative {
  \clef "bass"
  R1
  \cueClef "treble"
```

```
  d''8^"flute" c d e fis2
  \cueClefUnset
  d,,4 r a r
}
```




Per posizionamenti complessi delle notine, per esempio includere la trasposizione o inserire delle notine da varie sorgenti musicali, si possono usare i comandi `\cueDuring` o `\cueDuringWithClef`. Questi sono delle varianti più specializzate di `\quoteDuring`, introdotto in [Citare altre voci], pagina 213, nella sezione precedente.

La sintassi è:

```
\cueDuring nomecitazione #direzione musica
```

e

```
\cueDuringWithClef nomecitazione #direzione #chiave musica
```

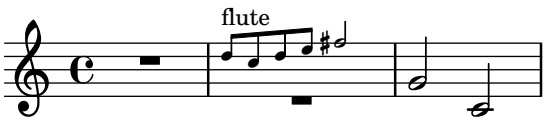
La musica delle misure che corrispondono a *`nomecitazione`* viene aggiunta in un contesto `CueVoice` e si colloca in simultanea con *`musica`*, creando quindi una situazione polifonica. La *`direzione`* prende l'argomento `UP` o `DOWN`, e corrisponde alla prima e alla seconda voce rispettivamente, determinando come le notine appaiono in relazione all'altra voce.

```
fluteNotes = \relative {
  r2. c''4 | d8 c d e fis2 | g2 d |
}

oboeNotes = \relative c'' {
  R1
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}
```
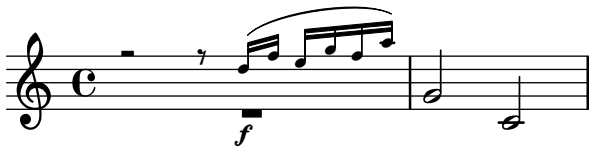



È possibile controllare quali aspetti della musica vengono citati con `\cueDuring` impostando la proprietà `quotedCueEventTypes`. Il suo valore predefinito è `'(note-event rest-event tie-event beam-event tuplet-span-event)`, che significa che vengono citati solo note, pause, legature di valore, travature e gruppi irregolari, ma non le articolazioni, le indicazioni dinamiche, il testo a margine, etc.

**Nota:** Quando una voce inizia con `\cueDuring`, come nell'esempio seguente, il contesto `Voice` deve essere dichiarato esplicitamente, altrimenti l'intera espressione musicale appartiene al contesto `CueVoice`.

```
oboeNotes = \relative {
  r2 r8 d''16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboeNotes }

\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring "oboe" #UP { R1 }
  g2 c,
}
```
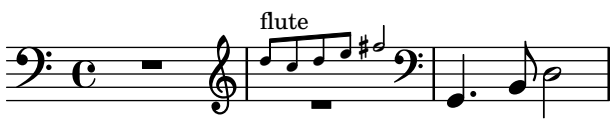


Si può usare il comando `\markup` per mostrare il nome dello strumento citato. Se le citazioni in corpo più piccolo richiedono un cambio di chiave, si può fare manualmente, ma anche il ripristino della chiave originale dovrà essere fatto manualmente al termine delle citazioni.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \clef treble
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```




Altrimenti si può usare la funzione `\cueDuringWithClef`. Questo comando prende un ulteriore argomento per specificare il cambio di chiave da usare per le citazioni in corpo più piccolo ma mostrerà automaticamente la chiave originale appena le citazioni sono finite.
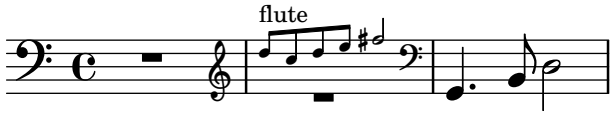
```
fluteNotes = \relative {
```

```
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  <>^\markup { \tiny "flute" }
  \cueDuringWithClef "flute" #UP "treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```




Come `\quoteDuring`, `\cueDuring` prende in considerazione la trasposizione degli strumenti. Le citazioni in corpo più piccolo vengono mostrate nelle altezze necessarie allo strumento che riprende la citazione per riprodurre gli stessi suoni dello strumento citato.

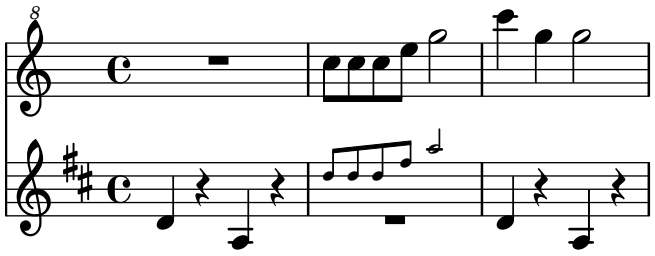
Per trasporre le citazioni in corpo più piccolo in modo diverso, si usa `\transposedCueDuring`. Questo comando prende un ulteriore argomento per specificare (in modalità assoluta) l'altezza da usare nella partitura per rappresentare il Do centrale in intonazione reale. È utile nel caso di citazioni da uno strumento che ha un registro completamente diverso.

```
piccoloNotes = \relative {
  \clef "treble^8"
  R1
  c'''8 c c e g2
  c4 g g2
}

bassClarinetNotes = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring "piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccoloNotes }

<<
  \new Staff \piccoloNotes
  \new Staff \bassClarinetNotes
>>
```



Il comando `\killCues` toglie le citazioni in corpo più piccolo da un'espressione musicale, in modo che la stessa espressione musicale possa essere usata per produrre sia la parte strumentale con le citazioni in corpo più piccolo sia l'intera partitura. Il comando `\killCues` toglie soltanto le note e gli eventi citati da `\cueDuring`. Altre annotazioni relative alle citazioni in corpo più piccolo, come i cambi di chiave e il nome che identifica lo strumento sorgente, possono essere contrassegnate per includerle in modo selettivo nella partitura; si veda [Uso delle etichette], pagina 515.

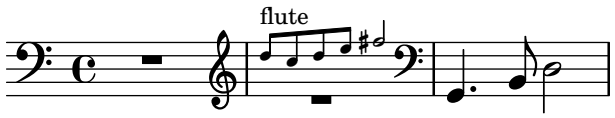
```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

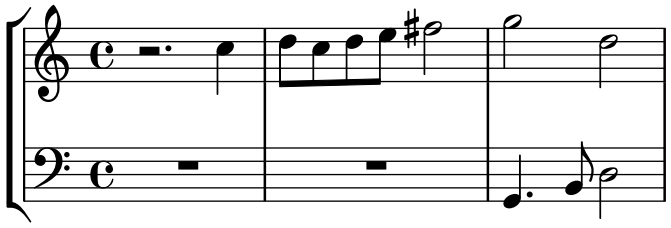
bassoonNotes = \relative c {
  \clef bass
  R1
  \tag #'part {
    \clef treble
    <>^\markup { \tiny "flute" }
  }
  \cueDuring "flute" #UP { R1 }
  \tag #'part \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>
```





### Vedi anche

Guida alla notazione: [Citare altre voci], pagina 213, [Trasporto strumentale], pagina 27, [Nomi degli strumenti], pagina 210, [Chiave], pagina 17, [Suggerimenti musicali], pagina 313, [Uso delle etichette], pagina 515.

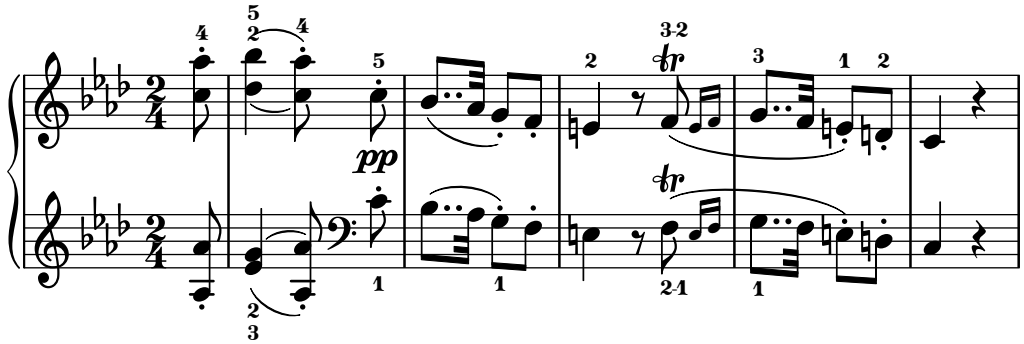
Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “CueVoice” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Quando si usa `\cueDuring`, si possono verificare delle collisioni tra le pause nel contesto `Voice` e quelle in `CueVoice`. Quando si usa `\cueDuringWithClef` o `\transposedCueDuring`, l’argomento ulteriore richiesto da entrambi deve venire dopo la citazione e la direzione.

## 1.7 Note editoriali




Questa sezione tratta dei vari modi con cui cambiare l’aspetto delle note e aggiungere un’analisi o un accento didattico.

### 1.7.1 Interne al rigo

Questa sezione spiega come aggiungere enfasi agli elementi interni al rigo.

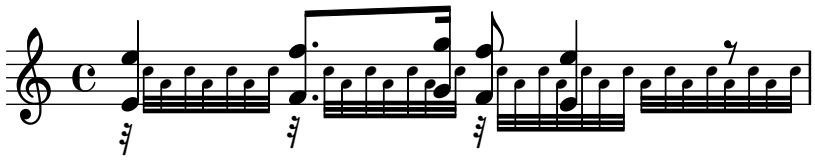
### Scelta della dimensione del tipo di carattere

> **Nota:**
> Per le dimensioni del testo, leggere [Scelta del tipo di carattere e della dimensione], pagina 248.
> Per la dimensione del rigo, leggere Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.
> Per le citazioni in corpo piccolo, leggere [Formattazione delle notine], pagina 217.
> Per i righi ossia, leggere [Righi ossia], pagina 204.

Per modificare la dimensione di un elemento della notazione senza cambiare anche la dimensione del rigo, si può specificare un fattore di ingrandimento col comando `\magnifyMusic`:

```
\new Staff <<
```

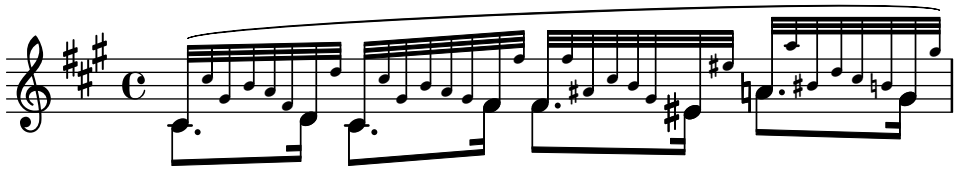
```
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



L'`\override` in questo esempio serve a eludere un difetto del programma, spiegato in "Known issues and warnings" alla fine di questa sezione.

Se la testa di una nota di dimensione normale è accorpata con una più piccola, potrebbe essere necessario ripristinare la dimensione della nota più piccola (con '`\once \normalsize`') in modo che i gambi e le alterazioni siano allineati correttamente:

```
\new Staff <<
  \key fis \minor
  \mergeDifferentlyDottedOn
  \new Voice \relative {
    \voiceOne
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      \once \normalsize cis'32( cis' gis b a fis \once \normalsize d d'
      \once \normalsize cis, cis' gis b a gis \once \normalsize fis fis'
      \once \normalsize fis, fis' ais, cis b gis \once \normalsize eis eis'
      \once \normalsize a, a' bis, d cis b \once \normalsize gis gis')
    }
  }
  \new Voice \relative {
    \voiceTwo
    cis'8. d16 cis8. fis16 fis8. eis16 a8. gis16
  }
>>
```



Il comando `\magnifyMusic` non è adatto per le citazioni in corpo piccolo, gli abbellimenti o i righi ossia, per i quali esistono metodi di inserimento più appropriati. È invece utile quando la dimensione della notazione cambia in una singola parte strumentale su un rigo e quando gli
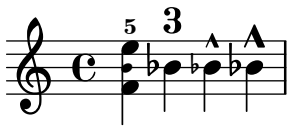
abbellimenti non sono la scelta appropriata, come nei passaggi di tipo cadenza o in casi simili agli esempi precedenti. Impostando il valore di `\magnifyMusic` su 0.63 si duplicano le dimensioni del contesto `CueVoice`.

> **Nota:** Il comando `\magnifyMusic` *non* deve essere usato quando si ridimensiona anche il rigo. Maggiori informazioni in Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

### *Ridimensionare oggetti della formattazione individualmente*

Un singolo oggetto della formattazione può essere ridimensionato coi comandi `\tweak` o `\override` per regolare la sua proprietà `font-size`:

```
\relative {
  % ridimensiona una testa di nota
  <f' \tweak font-size -4 b e>-5
  % ridimensiona una diteggiatura
  bes-\tweak font-size 0 -3
  % ridimensiona un'alterazione
  \once \override Accidental.font-size = -4 bes!-^
  % ridimensiona un'articolazione
  \once \override Script.font-size = 4 bes!-^
}
```
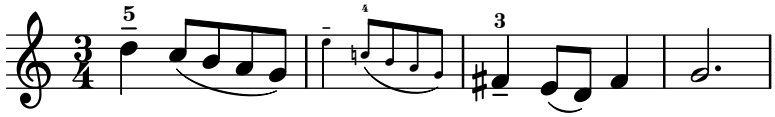


Il valore predefinito di `font-size` per ogni oggetto della formattazione è elencato nella Guida al funzionamento interno. La proprietà `font-size` può essere impostata solo per quegli oggetti che supportano l'interfaccia di formattazione `font-interface`. Se `font-size` non è specificato nella lista 'Standard settings' dell'oggetto, il suo valore è 0. Si veda Sezione "All layout objects" in *Guida al Funzionamento Interno*.

### *Capire la proprietà* `fontSize`

La proprietà di contesto `fontSize` regola la dimensione relativa di tutti gli elementi della notazione basati su un glifo in un contesto:

```
\relative {
  \time 3/4
  d''4---5 c8( b a g) |
  \set fontSize = -6
  e'4-- c!8-4( b a g) |
  \set fontSize = 0
  fis4---3 e8( d) fis4 |
  g2.
}
```



Il valore di `fontSize` è un numero che indica la dimensione relativa alla dimensione standard dell'altezza del rigo corrente. Il valore predefinito di `fontSize` è 0; aggiungendo 6 a qualsiasi

valore di `fontSize` si raddoppia la dimensione dei glifi e togliendo 6 si dimezza. Ogni punto aumenta la dimensione di circa il 12%.

Dato che le unità logaritmiche della proprietà `font-size` non sono del tutto intuitive, viene fornita per comodità la funzione Scheme `magnification->font-size`. Per esempio, per ridurre la notazione musicale al 75% della dimensione predefinita si usa:

```
\set fontSize = #(magnification->font-size 0.75)
```

La funzione Scheme `magstep` fa l'opposto: converte un valore di `font-size` in un fattore di ingrandimento.
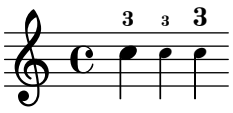
La proprietà `fontSize` avrà effetto soltanto sugli elementi della notazione che sono disegnati con glifi, come le teste di nota, le alterazioni, i segni, etc. Non modificherà la dimensione del rigo stesso né ridimensionerà proporzionalmente gambi, travature o la spaziatura orizzontale. Per ridimensionare gambi, travature e spaziatura orizzontale insieme alla dimensione degli elementi della notazione (senza cambiare la dimensione del rigo), si usa il comando `\magnifyMusic` presentato prima. Per ridimensionare tutto, compreso il rigo, leggere Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

Ogni volta che la *proprietà di contesto* `fontSize` è impostata, il suo valore viene aggiunto al valore della *proprietà del grob* `font-size` per i singoli oggetti di formattazione, prima che siano stampati i glifi. Ciò può creare confusione quando si impostano individualmente le proprietà `font-size` mentre è impostato anche `fontSize`:

```
% il valore predefinito di font-size per NoteHead è 0
% il valore predefinito di font-size per Fingering è -5
c''4-3

\set fontSize = -3
% la dimensione effettiva per NoteHead è ora -3
% la dimensione effettiva per Fingering è ora -8
c''4-3

\override Fingering.font-size = 0
% la dimensione effettiva per Fingering è ora -3
c''4-3
```
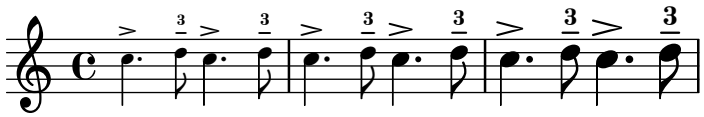



Sono anche disponibili le seguenti scorciatoie:

| **Comando** | **Equivalente a** | **Dimensione relativa** |
|---|---|---|
| `\teeny` | `\set fontSize = -3` | 71% |
| `\tiny` | `\set fontSize = -2` | 79% |
| `\small` | `\set fontSize = -1` | 89% |
| `\normalsize` | `\set fontSize = 0` | 100% |
| `\large` | `\set fontSize = 1` | 112% |
| `\huge` | `\set fontSize = 2` | 126% |

```
\relative c'' {
  \teeny
  c4.-> d8---3
  \tiny
  c4.-> d8---3
  \small
```

```
  c4.-> d8---3
  \normalsize
  c4.-> d8---3
  \large
  c4.-> d8---3
  \huge
  c4.-> d8---3
}
```



La modifica della dimensione del tipo di carattere si ottiene ridimensionando la dimensione, tra quelle predefinite, più vicina a quella desiderata. La dimensione standard (per `font-size = 0`) dipende dall'altezza standard del rigo: per un rigo di 20pt, viene scelto un tipo di carattere di 11pt.

## Comandi predefiniti

`\magnifyMusic`, `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

## Vedi anche

Guida alla notazione: [Scelta del tipo di carattere e della dimensione], pagina 248, Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552, [Formattazione delle notine], pagina 217, [Righi ossia], pagina 204.

File installati: `ly/music-functions-init.ly`, `ly/property-init.ly`.

Frammenti: Sezione "Editorial annotations" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "font-interface" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti
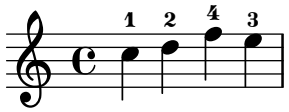
Attualmente ci sono due bug che impediscono una corretta spaziatura orizzontale quando si usa `\magnifyMusic`. C'è un solo modo per eludere questi bug e non funziona in tutte le circostanze. Nell'esempio seguente, sostituire la variabile *mag* con un valore a piacere. Si può provare anche a togliere uno o entrambi i comandi `\newSpacingSection` e/o i comandi `\override` e `\revert`:

```
\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [music]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}
```

## Indicazioni di diteggiatura
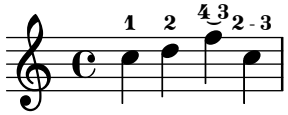
Le indicazioni di diteggiatura si inseriscono con '*nota-numero*':

```
\relative { c''4-1 d-2 f-4 e-3 }
```
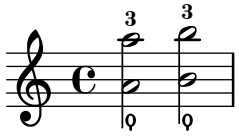


Si può usare il testo incluso dentro `\markup` o tra virgolette per indicare un cambio di dito.

```
\relative {
  c''4-1 d-2 f\finger \markup \tied-lyric "4~3" c\finger "2 - 3"
}
```
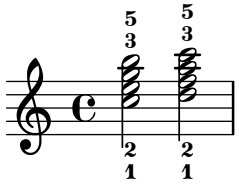


Si può aggiungere il simbolo del pollice per indicare che una nota deve essere suonata col pollice (ad esempio, nella musica per violoncello).

```
\relative { <a'_\thumb a'-3>2 <b_\thumb b'-3> }
```



È possibile indicare la diteggiatura di ogni singola nota di un accordo specificandola dopo ciascuna altezza.

```
\relative {
  <c''-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
}
```



Le indicazioni di diteggiatura possono essere poste sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

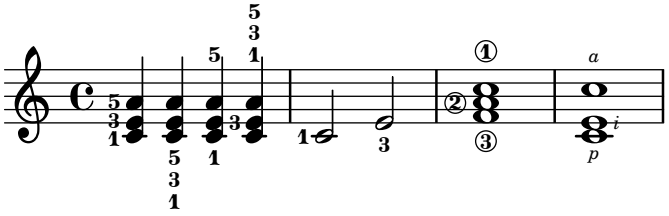
## Frammenti di codice selezionati

*Controllare il posizionamento delle diteggiature di un accordo*

Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi `<>` anche per le note singole. Si può impostare in modo simile l'orientamento dei numeri di corda e delle diteggiature della mano destra.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
```
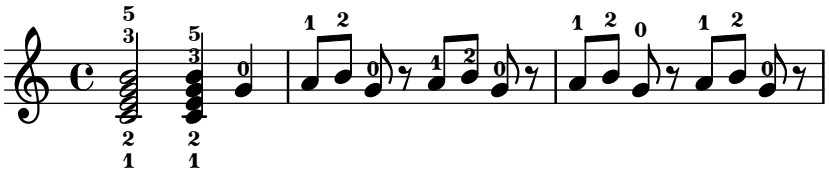
```
  <e-3>2
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger #1 e\rightHandFinger #2 c'\rightHandFinger #4 >
}
```



*Far sì che la diteggiatura appaia dentro il rigo*

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo; questo comportamento tuttavia può essere disabilitato. Occorre fare attenzione alle situazioni in cui le diteggiature e i gambi sono rivolti nella stessa direzione: normalmente le diteggiature evitano soltanto i gambi con travature. Questa impostazione predefinita può essere cambiata in modo da evitare tutti i gambi oppure nessuno. L'esempio seguente mostra queste due opzioni, così come tornare al comportamento predefinito.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 g'-0
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

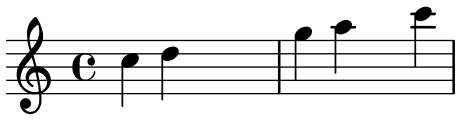
Frammenti: Sezione "Editorial annotations" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "FingeringEvent" in *Guida al Funzionamento Interno*, Sezione "fingering-event" in *Guida al Funzionamento Interno*, Sezione "Fingering_engraver" in *Guida al Funzionamento Interno*, Sezione "New_fingering_engraver" in *Guida al Funzionamento Interno*, Sezione "Fingering" in *Guida al Funzionamento Interno*.

## Note nascoste
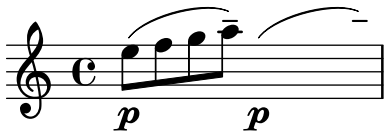
Le note nascoste (o invisibili o trasparenti) possono essere utili nella preparazione di esercizi di teoria e composizione.

```
\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
  b
  \unHideNotes
  c
}
```



Questo comando rende invisibili le teste, i gambi e le code delle note, e le pause. Le travature sono invisibili se iniziano su una nota nascosta. Mentre gli oggetti attaccati a note invisibili sono comunque visibili.

```
\relative c'' {
  e8(\p f g a)--
  \hideNotes
  e8(\p f g a)--
}
```




## Comandi predefiniti

`\hideNotes`, `\unHideNotes`.

## Vedi anche

Manuale d'apprendimento: Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: [Pause invisibili], pagina 61, Sezione 5.4.7 [Visibilità degli oggetti], pagina 638, [Nascondere i righi], pagina 208.

Frammenti: Sezione "Editorial annotations" in *Frammenti di codice*.

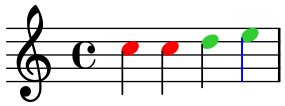
Guida al funzionamento interno: Sezione "Note_spacing_engraver" in *Guida al Funzionamento Interno*, Sezione "NoteSpacing" in *Guida al Funzionamento Interno*.

## Colorare gli oggetti

Si possono assegnare dei colori a ciascun oggetto. I nomi dei colori validi sono elencati nell'Sezione A.7 [Elenco dei colori], pagina 678.

```
\override NoteHead.color = #red
```
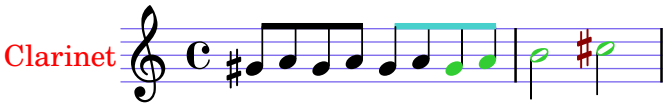
```
c''4 c''
\override NoteHead.color = #(x11-color 'LimeGreen)
d''
\override Stem.color = #blue
e''
```



Si può accedere all'intera gamma di colori definita per X11 con la funzione Scheme `x11-color`. La funzione prende un argomento, che può essere un simbolo nella forma `'FooBar` o una stringa nella forma `"FooBar"`. La prima forma è più veloce da scrivere e più efficiente. Tuttavia, la seconda forma permette di accedere ai colori X11 attraverso la forma del nome che ha più di una parola.

La funzione `x11-color`, se non riesce a comprendere il parametro, restituisce il colore nero.
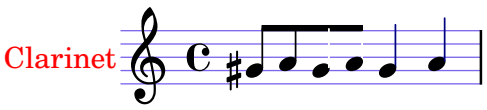
```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
    }
  }
  \relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  gis8 a
  \override Beam.color = #(x11-color "medium turquoise")
  gis a
  \override Accidental.color = #(x11-color 'DarkRed)
  gis a
  \override NoteHead.color = #(x11-color "LimeGreen")
  gis a
  % this is deliberate nonsense; note that the stems remain black
  \override Stem.color = #(x11-color 'Boggle)
  b2 cis
}
```




I colori RGB esatti si specificano con la funzione Scheme `rgb-color`.

```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
    }
  }
\relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  \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
}
```




## Vedi anche

Guida alla notazione: Sezione A.7 [Elenco dei colori], pagina 678, Sezione 5.3.4 [Il comando tweak], pagina 619.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Un colore X11 non ha necessariamente la stessa identica tonalità di un normale colore dal nome simile.
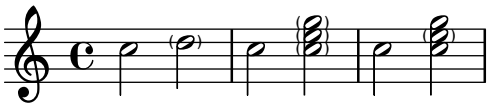
Non tutti i colori X11 sono distinguibili in un browser web. Per esempio, un browser potrebbe non mostrare alcuna differenza tra `LimeGreen` e `ForestGreen`. Per il web si consiglia di usare i colori normali (ovvero `blue`, `green`, `red`).

Le note in un accordo non possono essere colorate separatamente con un `\override`; al suo posto si usa `\tweak` o l’equivalente `\single\override`, vedi Sezione 5.3.4 [Il comando tweak], pagina 619.

## Parentesi
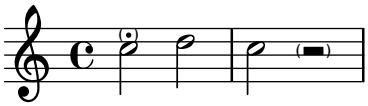
Gli oggetti possono essere messi tra parentesi se si usa il comando `\parenthesize` prima dell’evento musicale. Se precede un accordo, viene messa tra parentesi ogni nota dell’accordo. Si possono mettere tra parentesi anche singole note di un accordo.

```
\relative {
  c''2 \parenthesize d
  c2 \parenthesize <c e g>
  c2 <c \parenthesize e g>
}
```



Si possono mettere tra parentesi anche oggetti diversi dalle note. Per le articolazioni è necessario usare un trattino prima del comando `\parenthesize`.

```
\relative {
  c''2-\parenthesize -. d
  c2 \parenthesize r
}
```



## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Parenthesis_engraver” in *Guida al Funzionamento Interno*, Sezione “ParenthesesItem” in *Guida al Funzionamento Interno*, Sezione “parentheses-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se si mette tra parentesi un accordo, viene creata una parentesi per ogni nota dell’accordo invece di una sola grande parentesi per l’intero accordo.

## Gambi

Per ogni nota viene creato automaticamente un oggetto `Stem` (gambo). Vale anche per le semibrevi e le pause, anche se i loro gambi sono resi invisibili.

I gambi si possono posizionare sopra o sotto, vedi Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

## Comandi predefiniti
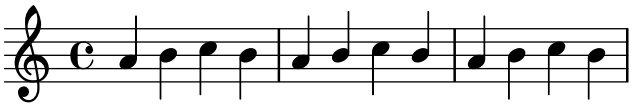
`\stemUp`, `\stemDown`, `\stemNeutral`.

## Frammenti di codice selezionati

*Direzione predefinita dei gambi sulla linea centrale del rigo*

La direzione predefinita dei gambi sulla linea centrale del rigo si imposta con la proprietà `neutral-direction` dell’oggetto `Stem`.

```
\relative c'' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}
```
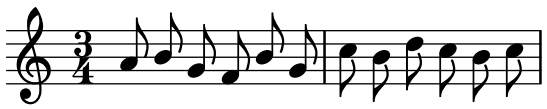


*Cambiare automaticamente la direzione del gambo della nota centrale in base alla melodia*

LilyPond può modificare la direzione del gambo della nota centrale di un rigo in modo che segua la melodia: occorre aggiungere l’incisore `Melody_engraver` al contesto Voice e sovrascrivere la proprietà `neutral-direction` di Stem.

```
\relative c'' {
  \time 3/4
  a8 b g f b g |
  c  b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
```

```
    \override Stem.neutral-direction = #'()
  }
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Stem_engraver” in *Guida al Funzionamento Interno*, Sezione “Stem” in *Guida al Funzionamento Interno*, Sezione “stem-interface” in *Guida al Funzionamento Interno*.
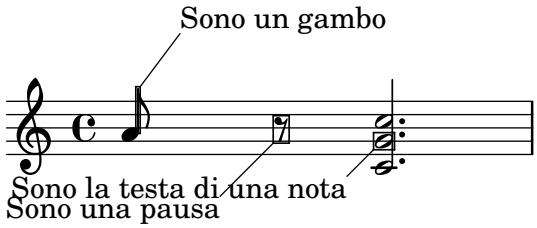
### 1.7.2 Esterne al rigo

Questa sezione spiega come dare risalto agli elementi nel rigo attraverso delle note esterne al rigo.

## Nuvoletta di aiuto

Si possono contrassegnare e nominare gli elementi della notazione tramite una nuvoletta quadrata. La sua funzione principale è spiegare la notazione.

```
\relative c'' {
  \new Voice \with { \consists "Balloon_engraver" }
  {
    \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
    a8
    \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
    r
    <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
  }
}
```

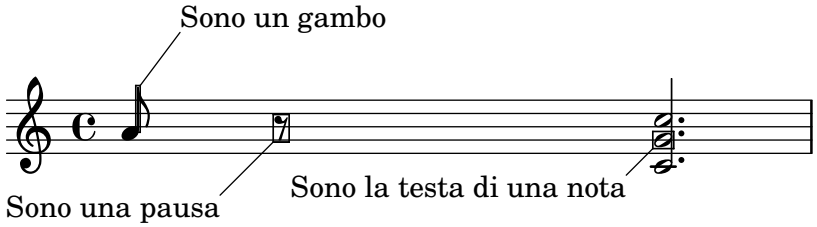



Ci sono due funzioni musicali, `balloonGrobText` e `balloonText`; la prima si usa nella forma `\once \override` per attaccare del testo a un qualsiasi oggetto grafico (grob), mentre la seconda viene usata come `\tweak`, solitamente all’interno degli accordi, per attaccare del testo a una singola nota.

Il testo nella nuvoletta influenza la spaziatura delle note, ma è possibile modificare questo comportamento:

```
\relative c'' {
  \new Voice \with { \consists "Balloon_engraver" }
  {
    \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
```

```
    a8
    \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
    r
    \balloonLengthOn
    <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
  }
}
```




## Comandi predefiniti

`\balloonLengthOn`, `\balloonLengthOff`.

## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Balloon_engraver” in *Guida al Funzionamento Interno*, Sezione “BalloonTextItem” in *Guida al Funzionamento Interno*, Sezione “balloon-interface” in *Guida al Funzionamento Interno*.

## Linee della griglia

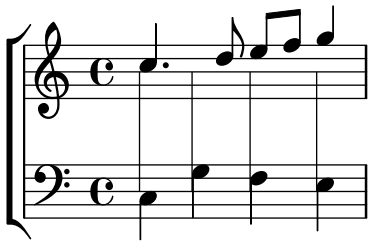
Si possono disegnare delle linee verticali tra i righi sincronizzate con le note.

Si deve usare l’incisore `Grid_point_engraver` per creare le estremità delle linee, mentre l’incisore `Grid_line_span_engraver` serve a disegnare le linee. Per impostazione predefinita, le linee della griglia sono allineate orizzontalmente sotto e sul lato sinistro delle teste di nota. Le linee si estendono a partire dalle linee centrali di ciascun rigo. `gridInterval` deve specificare la durata che separa le linee.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #(ly:make-moment 1/4)
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative {
      \stemUp
      c''4. d8 e8 f g4
    }
    \new Staff \relative {
```

```
      \clef bass
      \stemDown
      c4 g' f e
    }
  >>
}
```



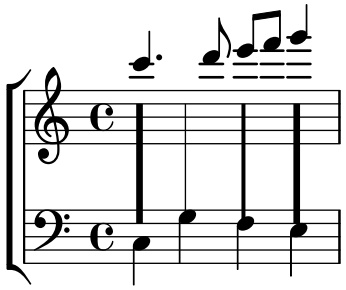
## Frammenti di codice selezionati

*Modificare l'aspetto delle linee della griglia*

L'aspetto delle linee della griglia può essere modificato sovrascrivendo alcune delle loro proprietà.

```
\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
    }
  }
}
```



## Vedi anche
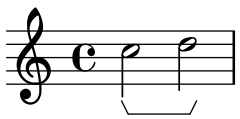
Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Grid_line_span_engraver” in *Guida al Funzionamento Interno*, Sezione “Grid_point_engraver” in *Guida al Funzionamento Interno*, Sezione “GridLine” in *Guida al Funzionamento Interno*, Sezione “GridPoint” in *Guida al Funzionamento Interno*, Sezione “grid-line-interface” in *Guida al Funzionamento Interno*, Sezione “grid-point-interface” in *Guida al Funzionamento Interno*.

## Parentesi analitiche

Nell’analisi musicale si usano le parentesi per indicare la struttura dei brani musicali. Sono supportate delle semplici parentesi orizzontali.

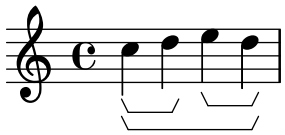
```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}
```



Le parentesi analitiche si possono annidare.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
```

```
  }
}
\relative {
  c''4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```
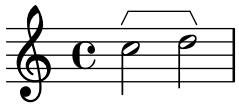


## Frammenti di codice selezionati

*Parentesi analitiche sopra il rigo*

Delle semplici parentesi analitiche orizzontali vengono aggiunte, per impostazione predefinita, sotto il rigo. L'esempio seguente mostra un modo per posizionarle sopra il rigo.

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

\relative c'' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```



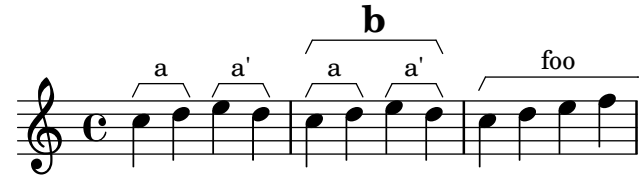
*Parentesi analitiche con etichette*

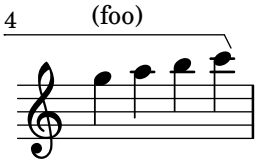
Si può aggiungere del testo alle parentesi analitiche tramite la proprietà `text` del grob `HorizontalBracketText`. L'aggiunta di vari frammenti di testo alle parentesi che iniziano nello stesso momento musicale richiede l'uso del comando `\tweak`. Dopo un'interruzione di linea il testo viene messo tra parentesi.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}

{
  \once\override HorizontalBracketText.text = "a"
    c''\startGroup d''\stopGroup
```

```
    \once\override HorizontalBracketText.text = "a'"
    e''\startGroup d''\stopGroup |
  c''-\tweak HorizontalBracketText.text
        \markup \bold \huge "b" \startGroup
     -\tweak HorizontalBracketText.text "a" \startGroup
    d''\stopGroup
    e''-\tweak HorizontalBracketText.text "a'" \startGroup
    d''\stopGroup\stopGroup |
  c''-\tweak HorizontalBracketText.text foo \startGroup
    d'' e'' f'' | \break
  g'' a'' b'' c'''\stopGroup
}
```

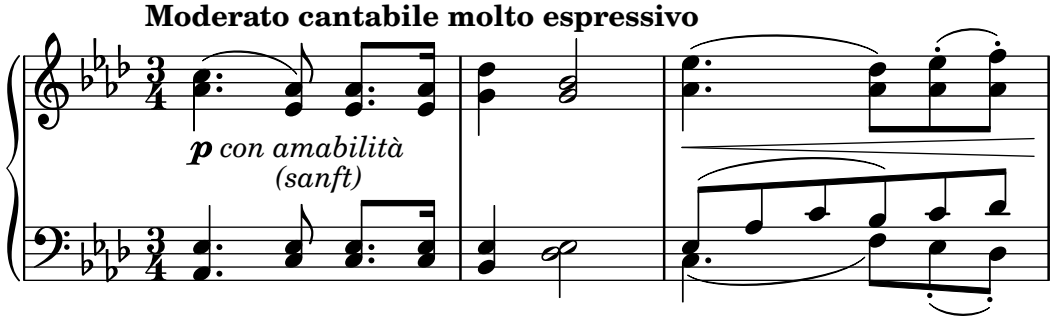


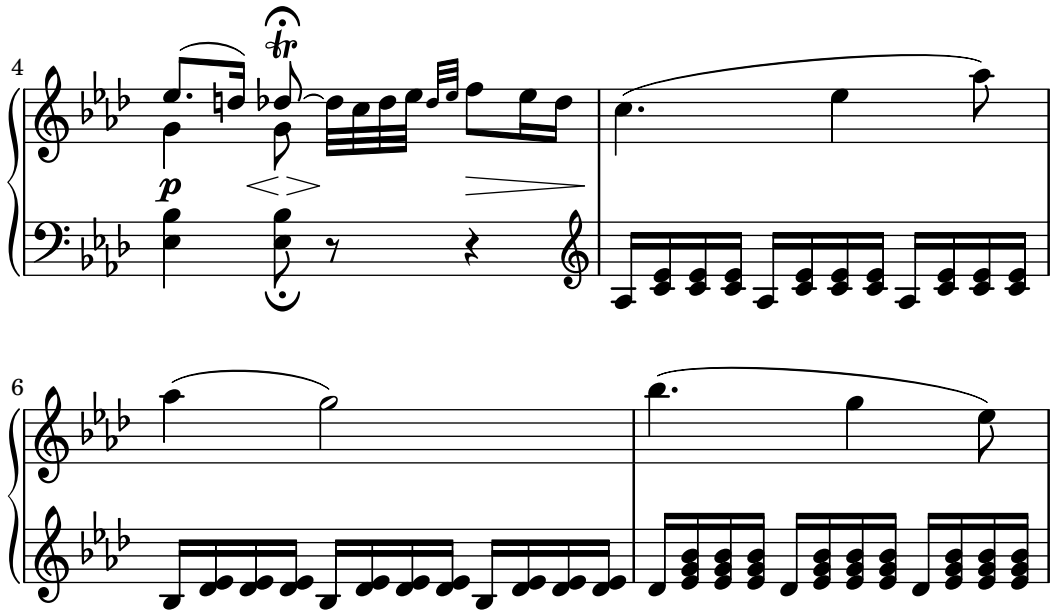




### Vedi anche

Guida al funzionamento interno: Sezione “Horizontal_bracket_engraver” in *Guida al Funzionamento Interno*, Sezione “HorizontalBracket” in *Guida al Funzionamento Interno*, Sezione “horizontal-bracket-interface” in *Guida al Funzionamento Interno*, Sezione “HorizontalBracketText” in *Guida al Funzionamento Interno*, Sezione “horizontal-bracket-text-interface” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## 1.8 Testo





Questa sezione spiega come includere del testo (con vari tipi di formattazione) nelle partiture musicali.

Alcuni elementi testuali che non sono trattati qui sono discussi in altre sezioni specifiche: Sezione 2.1 [Musica vocale], pagina 267, ⟨undefined⟩ [Titles and headers], pagina ⟨undefined⟩.

### 1.8.1 Inserimento del testo
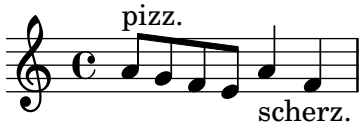
Questa sezione presenta vari modi di aggiungere del testo a una partitura.

> **Nota:** Per scrivere caratteri accentati e speciali (come quelli di altre lingue), basta inserire semplicemente i caratteri nel file LilyPond, purché il file sia salvato in formato UTF-8. Ulteriori informazioni in [Codifica del testo], pagina 519.

#### Scritte
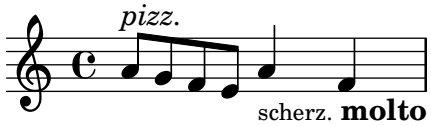
Si possono aggiungere a una partitura delle semplici indicazioni con del "testo tra virgolette", come mostrato nell'esempio seguente. Tali indicazioni possono essere posizionate sopra o sotto il rigo, usando la sintassi descritta in Sezione 5.4.2 [Direzione e posizionamento], pagina 630.

```
\relative { a'8^"pizz." g f e a4-"scherz." f }
```
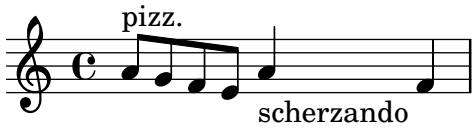



In realtà questa sintassi è una scorciatoia; si può specificare una formattazione del testo più complessa usando in modo esplicito un blocco `\markup`, come è spiegato in Sezione 1.8.2 [Formattazione del testo], pagina 246.

```
\relative {
  a'8^\markup { \italic pizz. } g f e
  a4_\markup { \tiny scherz. \bold molto } f }
```



Le indicazioni testuali, di norma, non influenzano la spaziatura delle note. Ma è possibile far sì che la loro larghezza venga presa in considerazione: nell'esempio seguente la prima stringa di testo non influenza la spaziatura, mentre la seconda sì.

```
\relative {
  a'8^"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}
```




Oltre alle scritte, si possono attaccare alle note anche le articolazioni. Ulteriori informazioni in [Articolazioni e abbellimenti], pagina 124.

Per maggiori informazioni sull'ordinamento relativo delle scritte e delle articolazioni si veda Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

## Comandi predefiniti

`\textLengthOn`, `\textLengthOff`.

## Vedi anche

Manuale d'apprendimento: Sezione "Posizionamento degli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Articolazioni e abbellimenti], pagina 124.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

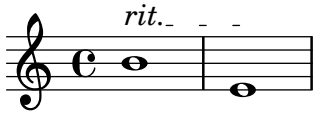
Per verificare che le scritte e il testo vocale siano entro i margini occorrono ulteriori calcoli. Nei casi in cui è richiesta un'esecuzione leggermente più veloce, usare

```
\override Score.PaperColumn.keep-inside-line = ##f
```

## Estensori del testo

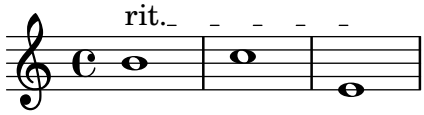
Alcune indicazioni esecutive, per esempio *rallentando* o *accelerando*, appaiono in forma testuale e vengono estese lungo molteplici note con delle linee punteggiate. Tali oggetti, chiamati "estensori" (spanner), si creano collegando due note con la seguente sintassi:

```
\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
```



La stringa testuale da stampare viene impostata attraverso le proprietà dell'oggetto. Per impostazione predefinita, appare in corsivo, ma si può ottenere una formattazione diversa tramite i blocchi `\markup`, come è spiegato in Sezione 1.8.2 [Formattazione del testo], pagina 246.

```
\relative {
  \override TextSpanner.bound-details.left.text =
    \markup { \upright "rit." }
  b'1\startTextSpan c
  e,\stopTextSpan
}
```




Lo stile della linea, così come la stringa testuale, può essere definito come una proprietà dell'oggetto. Questa sintassi è descritta in Sezione 5.4.8 [Stili della linea], pagina 644.

## Comandi predefiniti

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.

## Problemi noti e avvertimenti

LilyPond è capace di gestire un solo estensore del testo per ogni voce.
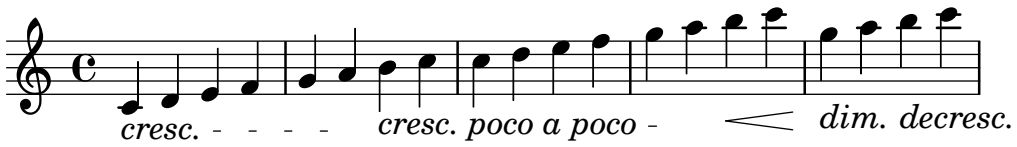
## Frammenti di codice selezionati

*Estensore testuale della dinamica personalizzato*

Si possono definire estensori testuali personalizzati che fanno uso delle forcine e dei crescendo testuali. `\<` e `\>` generano le forcine, `\cresc` etc. generano gli estensori testuali.

```
% Some sample text dynamic spanners, to be used as postfix operators
crpoco =
#(make-music 'CrescendoEvent
             'span-direction START
             'span-type 'text
             'span-text "cresc. poco a poco")

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decresc c4\!
}
```



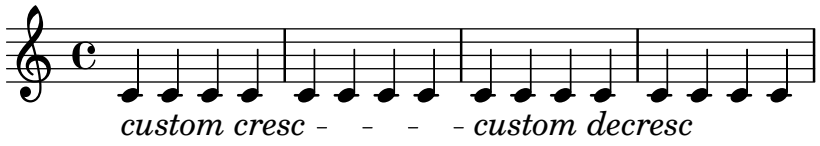

*Estensore testuale della dinamica personalizzato*

Funzioni postfix per estensori testuali personalizzati del crescendo. Gli estensori devono iniziare sulla prima nota della misura; e bisogna usare `-\mycresc`, altrimenti l'inizio dell'estensore viene assegnato alla nota successiva.

```
% Two functions for (de)crescendo spanners where you can explicitly
```

```
% give the spanner text.
mycresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'DecrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4\! c4 c4
}
```




## Vedi anche

Guida alla notazione: Sezione 5.4.8 [Stili della linea], pagina 644, [Dinamiche], pagina 126, Sezione 1.8.2 [Formattazione del testo], pagina 246.

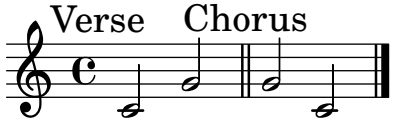
Frammenti: Sezione “Text” in *Frammenti di codice*, Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextSpanner” in *Guida al Funzionamento Interno*.

## Indicazioni testuali
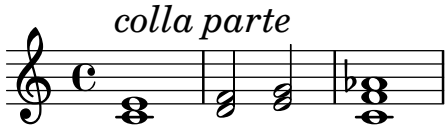
Si possono aggiungere vari elementi testuali a una partitura tramite la sintassi descritta in [Segni di chiamata], pagina 114:

```
\relative {
  \mark "Verse"
  c'2 g'
  \bar "||"
  \mark "Chorus"
  g2 c,
  \bar "|."
}
```
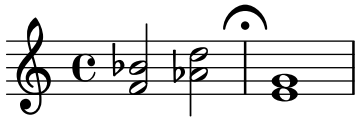


Questa sintassi permette di porre del testo sopra una stanghetta; una formattazione del testo più complessa è possibile grazie al blocco `\markup`, come è spiegato in Sezione 1.8.2 [Formattazione del testo], pagina 246:

```
\relative {
  <c' e>1
  \mark \markup { \italic { colla parte } }
  <d f>2 <e g>
  <c f aes>1
}
```
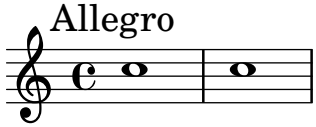



Questa sintassi permette anche di stampare segni speciali, come coda, segno o corona, se si specifica il nome appropriato del simbolo, come è spiegato in [Notazione musicale nel blocco markup], pagina 257:
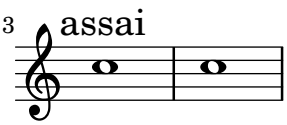
```
\relative {
  <bes' f>2 <aes d>
  \mark \markup { \musicglyph "scripts.ufermata" }
  <e g>1
}
```



Tali oggetti vengono posizionati soltanto sopra il rigo superiore della partitura; a seconda che siano inseriti alla fine o a metà di una battuta, possono trovarsi sopra la stanghetta o tra le note. Se inserito prima di un'interruzione di linea, l'indicazione apparirà all'inizio della linea successiva.

```
\relative c'' {
  \mark "Allegro"
  c1 c
  \mark "assai" \break
  c  c
}
```







## Comandi predefiniti
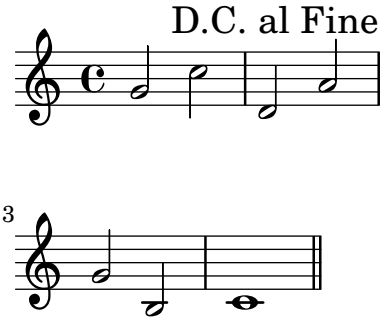
`\markLengthOn`, `\markLengthOff`.

## Frammenti di codice selezionati

*Posizionare le indicazioni alla fine di una linea*

È possibile posizionare le indicazioni alla fine della linea corrente, invece che all'inizio della linea successiva. In tali casi, può essere preferibile allineare l'estremità destra dell'indicazione alla stanghetta.

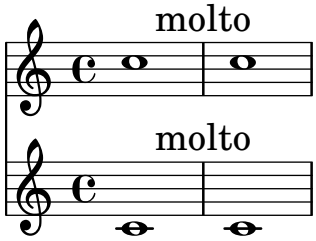
```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility =
    #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X =
    #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}
```




*Stampare le indicazioni su ogni rigo*

Sebbene le indicazioni testuali siano di norma collocate solo sopra il rigo più alto, è possibile farle apparire su ogni rigo.

```
\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"
    }
  }
}
```




## Vedi anche

Guida alla notazione: [Segni di chiamata], pagina 114, Sezione 1.8.2 [Formattazione del testo], pagina 246, [Notazione musicale nel blocco markup], pagina 257, Sezione A.8 [Il font Emmentaler], pagina 679.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MarkEvent” in *Guida al Funzionamento Interno*, Sezione “Mark_engraver” in *Guida al Funzionamento Interno*, Sezione “RehearsalMark” in *Guida al Funzionamento Interno*.

## Testo separato

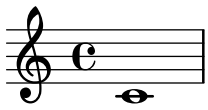
Un blocco `\markup` può esistere di per sé, fuori da qualsiasi blocco `\score`, come un’ “espressione di livello superiore”. Questa sintassi è descritta in Sezione 3.1.5 [Struttura del file], pagina 484.

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

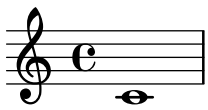
Tomorrow, and tomorrow, and tomorrow...

Ciò permette di stampare il testo in modo autonomo dalla musica, ed è utile soprattutto quando il file di input contiene vari brani musicali, come è spiegato in Sezione 3.1.2 [Molteplici partiture in un libro], pagina 481.

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Blocchi di testo separati possono essere estesi per molte pagine, rendendo possibile la realizzazione di documenti o libri interamente con LilyPond. Questa funzionalità, e la sintassi specifica che richiede, è descritta in [Testo formattato su più pagine], pagina 260.

## Comandi predefiniti

`\markup`, `\markuplist`.

## Frammenti di codice selezionati

*Testo separato su due colonne*

Il testo separato può essere disposto su varie colonne con i comandi di `\markup`:

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column \italic {
      \line { O sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }
    \hspace #1
  }
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

*O sacred feast*
*in which Christ is received,*
*the memory of His Passion is renewed,*
*the mind is filled with grace,*
*and a pledge of future glory is given to us.*
*Amen.*

## Vedi anche

Guida alla notazione: Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione 3.1.5 [Struttura del file], pagina 484, Sezione 3.1.2 [Molteplici partiture in un libro], pagina 481, [Testo formattato su più pagine], pagina 260.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## 1.8.2 Formattazione del testo

Questa sezione presenta la formattazione del testo basilare e quella avanzata, usando la sintassi specifica della modalità `\markup`.

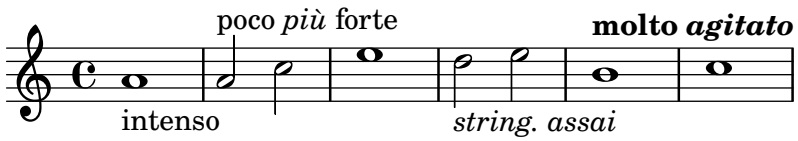
## Introduzione al testo a margine

Un blocco `\markup` permette di comporre del testo con un'ampia sintassi chiamata "modalità markup".

La sintassi di markup è simile alla solita sintassi di LilyPond: un'espressione `\markup` viene racchiusa tra parentesi graffe `{... }`. Una singola parola viene considerata un'espressione minima, e quindi non è necessario racchiuderla tra parentesi.
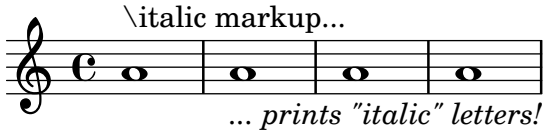
Diversamente dalle indicazioni testuali "tra virgolette", i blocchi `\markup` possono contenere espressioni o comandi di markup annidati, inseriti col carattere di barra inversa `\`. Tali comandi hanno effetto solo sulla prima espressione che segue.

```
\relative {
  a'1-\markup intenso
  a2^\markup { poco \italic più forte  }
  c e1
  d2_\markup { \italic "string. assai" }
  e
  b1^\markup { \bold { molto \italic  agitato } }
  c
}
```



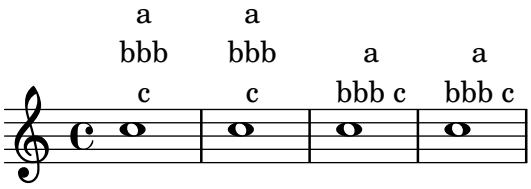

Un blocco `\markup` può contenere anche stringhe di testo tra virgolette. Tali stringhe vengono trattate come espressioni testuali minime, e quindi qualsiasi comando di markup o carattere speciale (come `\` e `#`) apparirà alla lettera senza influenzare la formattazione del testo. Le stesse doppie virgolette possono essere stampate facendole precedere da una barra inversa.

```
\relative {
  a'1^"\italic markup..."
  a_\markup { \italic "... prints \"italic\" letters!" }
  a a
}
```




Perché sia trattata come un'espressione distinta, una lista di parole deve essere racchiusa tra virgolette doppie o preceduta da un comando. Il modo in cui le espressioni musicali sono definite influenza il modo in cui saranno sistemate, centrate e allineate; nell'esempio seguente, la seconda espressione di `\markup` viene trattata nello stesso modo della prima:
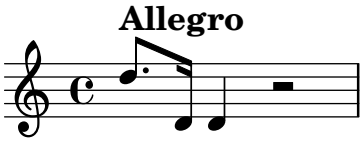
```
\relative c'' {
  c1^\markup { \center-column { a bbb c } }
  c1^\markup { \center-column { a { bbb c } } }
  c1^\markup { \center-column { a \line { bbb c } } }
  c1^\markup { \center-column { a "bbb c" } }
}
```




I markup possono essere salvati in delle variabili, che possono poi essere attaccate direttamente alle note:

```
allegro = \markup { \bold \large Allegro }

{
  d''8.^\allegro
  d'16 d'4 r2
}
```




Una lista completa dei comandi specifici di `\markup` si trova in Sezione A.11 [Comandi per markup], pagina 707.

## Vedi anche

Guida alla notazione: Sezione A.11 [Comandi per markup], pagina 707.

Frammenti: Sezione “Text” in *Frammenti di codice*.

File installati: `scm/markup.scm`.
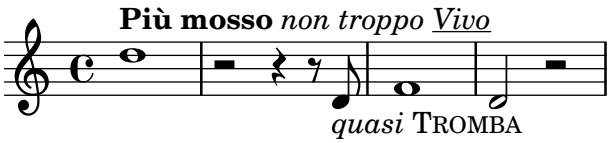
## Problemi noti e avvertimenti

I messaggi di errore di sintassi relativi alla modalità markup possono essere poco chiari.

## Scelta del tipo di carattere e della dimensione

La modalità markup permette di cambiare il tipo di carattere:

```
\relative {
  d''1^\markup {
    \bold { Più mosso }
    \italic { non troppo \underline Vivo }
  }
  r2 r4 r8
  d,_\markup { \italic quasi \smallCaps Tromba }
  f1 d2 r
}
```

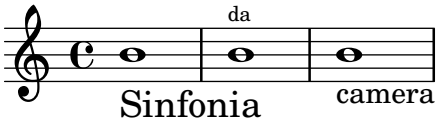



Si può modificare la dimensione del tipo di carattere, rispetto alla dimensione globale del rigo, in vari modi.

Si può impostare su una dimensione predefinita.

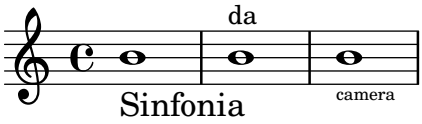
```
\relative b' {
  b1_\markup { \huge Sinfonia }
```

```
  b1^\markup { \teeny da }
  b1-\markup { \normalsize camera }
}
```



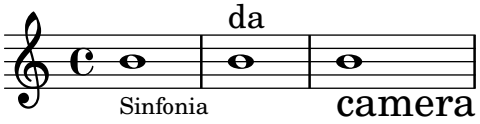

oppure in modo proporzionale rispetto al valore precedente.

```
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6  camera }
}
```



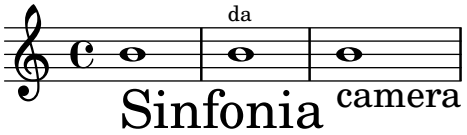

Può essere aumentata o diminuita rispetto al valore impostato per la dimensione globale del rigo.

```
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}
```




Si può impostare anche su una dimensione fissa (in punti), indipendentemente dalla dimensione globale del rigo.

```
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}
```




Se il testo contiene degli spazi, è meglio racchiuderlo tutto tra virgolette, in modo che la dimensione di ciascun spazio sia adatta alla dimensione degli altri caratteri.

```
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }
```

**Sinfonia da camera**

**Sinfonia da camera**

# Sinfonia da camera

È possibile stampare il testo come pedice o apice. Per impostazione predefinita, questo appaiono in corpo più piccolo, ma si può usare anche un corpo 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)

La modalità di markup fornisce un modo semplice per scegliere famiglie di caratteri diverse. Se non specificato altrimenti, viene scelto automaticamente il carattere tipografico con grazie (il tipo romano); nell'ultima linea dell'esempio seguente non c'è differenza tra la prima e la seconda parola.

```
\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.

Alcune di queste famiglie di caratteri, usate per elementi specifici come i numeri o le dinamiche, non forniscono tutti i caratteri, come accennato in [Nuove indicazioni dinamiche], pagina 133, e [Indicazioni di ripetizione manuali], pagina 160.

Se usati all'interno di una parola, alcuni comandi che cambiano il tipo di carattere o la formattazione potrebbero produrre uno spazio vuoto indesiderato. Si può facilmente risolvere concatenando insieme gli elementi testuali:

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

**1st movement**
***p**, con dolce espressione*

Una lista completa dei comandi per cambiare il tipo di carattere o per usare tipi di carattere personalizzati si trova in Sezione A.11.1 [Font], pagina 707.

È possibile anche definire i propri gruppi di tipi di carattere, come è spiegato in Sezione 1.8.3 [Tipi di carattere], pagina 260.

## Comandi predefiniti

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

## Vedi anche

Guida alla notazione: Sezione A.11.1 [Font], pagina 707, [Nuove indicazioni dinamiche], pagina 133, [Indicazioni di ripetizione manuali], pagina 160, Sezione 1.8.3 [Tipi di carattere], pagina 260.

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

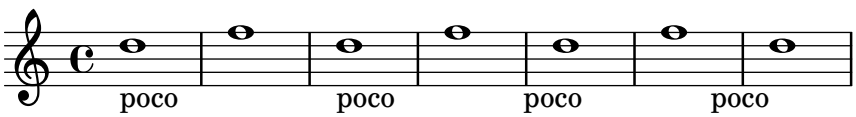
L’uso dei comandi di dimensionamento dei caratteri `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` e `\huge` produce una spaziatura della linea imprevedibile rispetto all’uso di `\fontsize`.

## Allineamento del testo

Questa sottosezione spiega come posizionare il testo nella modalità markup. Gli oggetti markup possono anche essere spostati interamente tramite la sintassi descritta in Sezione “Spostare gli oggetti” in *Manuale di Apprendimento*.

Gli oggetti di markup possono essere allineati in vari modi. Per impostazione predefinita, l’indicazione testuale è allineata rispetto al suo margine sinistro: nell’esempio seguente, non c’è differenza tra il primo e il secondo markup.

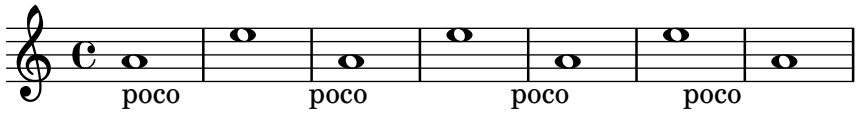
```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \left-align poco }
  f
  d-\markup { \center-align { poco } }
  f
  d-\markup { \right-align poco }
}
```




L’allineamento orizzontale può essere ritoccato usando un valore numerico:

```
\relative {
  a'1-\markup { \halign #-1 poco }
  e'
  a,-\markup { \halign #0 poco }
  e'
  a,-\markup { \halign #0.5 poco }
```

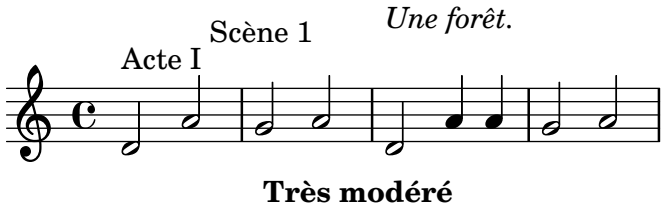
```
  e'
  a,-\markup { \halign #2 poco }
}
```




Alcuni oggetti possono avere proprie procedure di allineamento, e dunque non sono influenzate da questi comandi. È possibile spostare tali oggetti di markup tutti insieme, come mostrato ad esempio in [Indicazioni testuali], pagina 242.

L'allineamento verticale è un po' più complesso. Come si è detto prima, gli oggetti di markup possono essere spostati tutti insieme; tuttavia è anche possibile spostare elementi specifici all'interno di un blocco markup. In questo caso l'elemento da spostare deve essere preceduto da un *punto di riferimento*, che può essere un altro elemento markup o un oggetto invisibile. L'esempio seguente illustra queste due possibilità; l'ultimo markup in questo esempio non ha un punto di riferimento e di conseguenza non si muove.

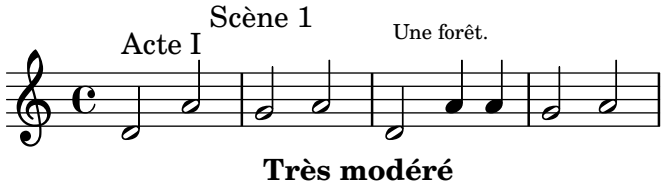
```
\relative {
  d'2^\markup {
    Acte I
    \raise #2 { Scène 1 }
  }
  a'
  g_\markup {
    \null
    \lower #4 \bold { Très modéré }
  }
  a
  d,^\markup {
    \raise #4 \italic { Une forêt. }
  }
  a'4 a g2 a
}
```




Alcuni comandi possono cambiare l'allineamento sia orizzontale che verticale degli oggetti testuali in modalità markup. Qualsiasi oggetto interessato da questi comandi deve essere preceduto da un punto di riferimento:

```
\relative {
  d'2^\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 oggetto markup può includere varie linee di testo. Nell'esempio seguente, ogni elemento o espressione viene posizionato sulla sua linea, allineato a sinistra o centrato:

```
\markup {
  \column {
    a
    "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a
    "b c"
    \line { d e f }
  }
}
```




Analogalmente, una lista di elementi o espressioni può essere distesa per riempire l'intera larghezza orizzontale della linea (se c'è un solo elemento, verrà centrato sulla pagina). Queste espressioni possono a loro volta includere del testo multilinea o una qualsiasi altra espressione di 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

Indicazioni testuali lunghe possono andare a capo automaticamente in base alla larghezza della linea specificata. Possono essere allineate a sinistra o giustificate, come mostra l'esempio seguente.

```
\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)*

Una lista completa dei comandi di allineamento del testo si trova in Sezione A.11.2 [Align], pagina 717.

## Vedi anche

Manuale d'apprendimento: Sezione "Spostare gli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione A.11.2 [Align], pagina 717, [Indicazioni testuali], pagina 242.

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione "Text" in *Frammenti di codice*.

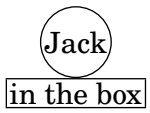
Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

## Notazione grafica nel blocco markup

Si possono aggiungere vari oggetti grafici a una partitura attraverso i comandi di markup.

Alcuni comandi di markup consentono di decorare gli elementi testuali con degli elementi grafici, come è illustrato nell'esempio seguente.

```
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
    }
    \null
    \rounded-box \bold Prelude
  }
}
```




Erik Satie [1866 - 1925]

**Prelude**

Alcuni comandi possono richiedere un aumento del padding intorno al testo; per farlo si usano dei comandi di markup, descritti in modo esaustivo in Sezione A.11.2 [Align], pagina 717.

```
\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

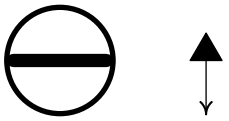
**Largo to Presto**

String quartet keeps very even time, Flute quartet keeps very uneven time.

Si possono produrre altri elementi grafici o simboli che non richiedono alcun testo. Come con qualsiasi espressione di markup, tali oggetti possono essere combinati.

```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

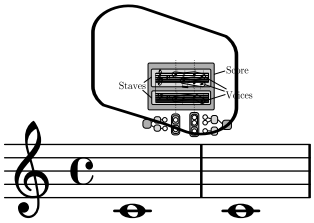
  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Le funzionalità grafiche avanzate comprendono la possibilità di includere file di immagini convertite nel formato Encapsulated PostScript (*eps*), oppure di inserire la grafica direttamente nel file di input, usando del codice PostScript nativo. In tal caso, può essere utile specificare esplicitamente la dimensione del disegno, come è mostrato sotto:

```
c'1^\markup {
  \combine
    \epsfile #X #10 "./context-example.eps"
    \with-dimensions #'(0 . 6) #'(0 . 10)
    \postscript "
      -2 3 translate
```

```
        2.7 2 scale
        newpath
        2 -1 moveto
        4 -2 4 1 1 arct
        4 2 3 3 1 arct
        0 4 0 3 1 arct
        0 0 1 -1 1 arct
        closepath
        stroke"
    }
c'
```




Una lista completa dei comandi specifici per la grafica si trova in Sezione A.11.3 [Graphic], pagina 732.

## Vedi anche

Guida alla notazione: Sezione A.11.2 [Align], pagina 717, Sezione 5.4.4 [Dimensioni], pagina 632, Sezione 1.7 [Note editoriali], pagina 222, Sezione A.11.3 [Graphic], pagina 732.

File installati: `scm/define-markup-commands.scm`, `scm/stencil.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

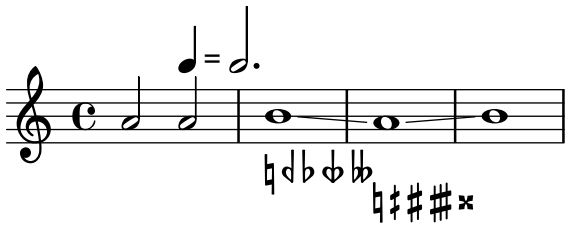
Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Notazione musicale nel blocco markup

Si possono aggiungere vari elementi della notazione musicale dentro un oggetto markup.
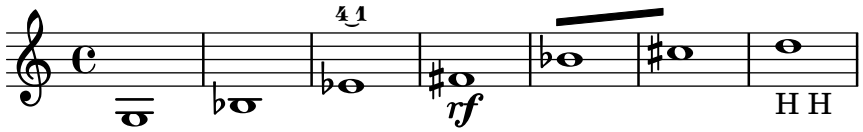
Per le note e le alterazioni esistono dei comandi markup appositi:

```
a'2 a'^\markup {
  \note {4} #1
  =
  \note-by-number #1 #1 #1.5
}
b'1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a'1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b'
```



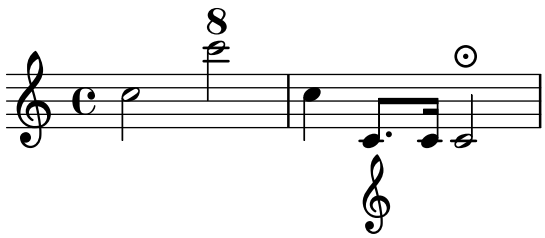
Anche altri oggetti della notazione possono essere stampati in modalità markup:

```
\relative {
  g1 bes
  ees\finger \markup \tied-lyric "4~1"
  fis_\markup { \dynamic rf }
  bes^\markup {
    \beam #8 #0.1 #0.5
  }
  cis
  d-\markup {
    \markalphabet #8
    \markletter #8
  }
}
```




Più in generale, qualsiasi simbolo musicale disponibile può essere incluso separatamente in un oggetto markup, come è illustrato sotto. Una lista completa di questi simboli e dei loro nomi si trova in Sezione A.8 [Il font Emmentaler], pagina 679.

```
\relative {
  c''2
  c'^\markup { \musicglyph "eight" }
  c,4
  c,8._\markup { \musicglyph "clefs.G_change" }
  c16
  c2^\markup { \musicglyph "timesig.neomensural94" }
}
```

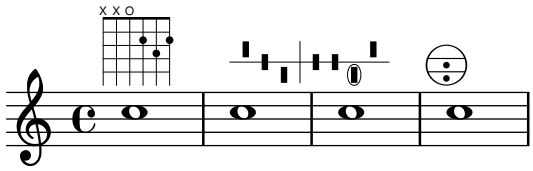



Un altro modo per stampare glifi non testuali è descritto in [Tipi di carattere in dettaglio], pagina 261. È utile per stampare parentesi di varie dimensioni.

La modalità markup supporta anche i diagrammi per strumenti specifici:

```
\relative {
  c''1^\markup {
    \fret-diagram-terse "x;x;o;2;3;2;"
  }
```
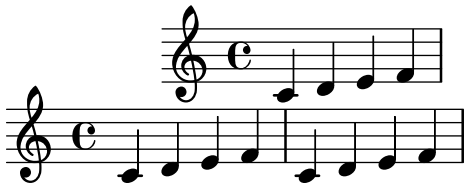
```
  c^\markup {
    \harp-pedal "^-v|--ov^"
  }
  c
  c^\markup {
    \combine
      \musicglyph "accordion.discant"
      \combine
        \raise #0.5 \musicglyph "accordion.dot"
        \raise #1.5 \musicglyph "accordion.dot"
  }
}
```



Questi diagrammi sono documentati in Sezione A.11.5 [Instrument Specific Markup], pagina 746.

È possibile annidare perfino un'intera partitura in un oggetto markup:

```
\relative {
  c'4 d^\markup {
    \score {
      \relative { c'4 d e f }
    }
  }
  e f |
  c d e f
}
```



Una lista completa dei comandi relativi alla notazione musicale si trova in Sezione A.11.4 [Music], pagina 740.

## Vedi anche

Guida alla notazione: Sezione A.11.4 [Music], pagina 740, Sezione A.8 [Il font Emmentaler], pagina 679, [Tipi di carattere in dettaglio], pagina 261.

File installati: `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

La spaziatura verticale di un blocco `\score` interno a un oggetto markup è regolata da `baseline-skip`. Le impostazioni in `\paper` vengono ignorate.

### Testo formattato su più pagine

Sebbene gli oggetti di markup standard non possano avere interruzioni, una specifica sintassi permette di inserire linee di testo che possono estendersi per varie pagine:

```
\markuplist {
  \justified-lines {
    Un testo molto lungo di linee giustificate.
    ...
  }
  \wordwrap-lines {
    Un altro paragrafo molto lungo.
    ...
  }
  ...
}
```

Un testo molto lungo di linee giustificate. ...

Un altro paragrafo molto lungo. ...

...

Questa sintassi accetta una lista di oggetti di markup, che possono essere

- il risultato di un comando \markuplist,
- una lista di markup,
- una lista di \markuplists.

Una lista completa dei comandi che si possono usare con \markuplist si trova in Sezione A.12 [Comandi per una lista di markup], pagina 761.

### Vedi anche

Guida alla notazione: Sezione A.12 [Comandi per una lista di markup], pagina 761.

Estendere LilyPond: Sezione “New markup list command definition” in *Estendere*.

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

### Comandi predefiniti

`\markuplist`.

## 1.8.3 Tipi di carattere

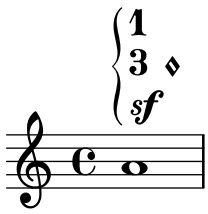
Questa sezione presenta il modo in cui sono gestiti i tipi di carattere e come possono essere modificati nelle partiture.

### Tipi di carattere in dettaglio

I tipi di carattere vengono gestiti attraverso varie librerie. FontConfig rileva i tipi di carattere disponibili nel sistema; i tipi selezionati sono riprodotti con Pango.

I tipi di carattere della notazione musicale possono essere descritti come un insieme di glifi specifici, ordinati in varie famiglie. La seguente sintassi permette di usare vari glifi *Feta* di LilyPond direttamente nella modalità markup:

```
a'1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup "brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup "noteheads.s0petrucci"
  }
}
```




Tuttavia, tutti questi glifi, ad eccezione delle graffe di varie dimensioni contenute in `fetaBraces`, sono già utilizzabili con la sintassi ben più semplice descritta in [Notazione musicale nel blocco markup], pagina 257.

Quando si usano i glifi contenuti in `fetaBraces`, la dimensione della graffa viene specificata dalla parte numerica del nome del glifo, in unità arbitrarie. Può essere specificato qualsiasi numero intero da `0` a `575` compresi, dove `0` corrisponde alla graffa più piccola. Il valore ottimale deve essere determinato per tentativi. Questi glifi sono tutte graffe sinistre; le graffe destre si possono ottenere con la rotazione, vedi Sezione 5.4.9 [Rotazione degli oggetti], pagina 645.

Sono disponibili tre famiglie di font: il font *roman* (con grazie), il font *sans* (senza grazie) e il font monospaziato *typewriter*.

Per il backend `svg`:

| **Famiglia** | **Font predefinito** |
|---|---|
| *roman* | `serif` |
| *sans* | `sans-serif` |
| *typewriter* | `monospace` |

`serif`, `sans-serif`, e `monospace` sono `generic-family` (famiglie generiche) nelle specifiche SVG e CSS.

Per gli altri backend:
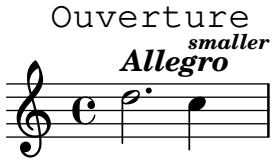
| **Famiglia** | **Font predefinito (alias)** | **Elenchi di definizione di alias** |
|---|---|---|
| *roman* | `LilyPond Serif` | TeX Gyre Schola, C059, Century SchoolBook URW, Century Schoolbook L, DejaVu Serif, ..., serif |
| *sans* | `LilyPond Sans Serif` | TeX Gyre Heros, Nimbus Sans, Nimbus Sans L, DejaVu Sans, ..., sans-serif |

| | | |
|---|---|---|
| *typewriter* | `LilyPond Monospace` | TeX Gyre Cursor, Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, DejaVu Sans Mono, ..., monospace |

`LilyPond Serif`, `LilyPond Sans Serif`, e `LilyPond Monospace` sono font alias definiti nel file di configurazione FontConfig specifico per LilyPond `00-lilypond-fonts.conf`. Se un carattere non esiste nel primo font elencato, il font successivo dell'elenco verrà usato al suo posto per quel carattere. I dettagli delle definizioni degli alias si trovano nel file `00-lilypond-fonts.conf` all'interno della directory di installazione.

Ogni famiglia può avere forme e serie differenti. L'esempio seguente illustra la possibilità di scegliere famiglie, forme, serie e dimensioni alternative. Il valore specificato per `font-size` è la modifica relativa alla dimensione predefinita.

```
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d''2.^\markup "Allegro"
\override Voice.TextScript.font-size = #-3
c''4^smaller
```




Una sintassi simile si usa nella modalità markup; tuttavia in questo caso è preferibile usare la sintassi più semplice spiegata in [Scelta del tipo di carattere e della dimensione], pagina 248:

```
\markup {
  \column {
    \line {
      \override #'((font-shape . italic) (font-size . 4))
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter)
      {
        \override #'(font-series . bold)
        re
        di
      }
      \override #'(font-family . sans)
      Creta
    }
  }
}
```




Quando si usano i font OpenType, sono disponibili le funzionalità dei font. Nota bene: non tutti i font OpenType hanno a disposizione tutte le funzioni. Se si richiede una funzionalità che non esiste nel font scelto, tale funzionalità viene semplicemente ignorata.

```
% Maiuscoletto vero
```

```
\markup { Stile normale: Hello HELLO }
\markup { \caps { Maiuscoletto: Hello } }
\markup { \override #'(font-features . ("smcp"))
          { Maiuscoletto vero: Hello } }

% Stili numerici
\markup { Stile numerico normale: 0123456789 }
\markup { \override #'(font-features . ("onum"))
          { Stile numerico vecchio: 0123456789 } }

% Alternative stilistiche
\markup { \override #'(font-features . ("salt 0"))
          { Alternative stilistiche 0: εφπρθ } }
\markup { \override #'(font-features . ("salt 1"))
          { Alternative stilistiche 1: εφπρθ } }

% Funzionalità multiple
\markup { \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Funzionalità multiple: Hello 0123456789 εφπρθ } }
```

Stile normale: Hello HELLO

Maiuscoletto: Hello

Maiuscoletto vero: Hello

Stile numerico normale: 0123456789

Stile numerico vecchio: 0123456789

Alternative stilistiche 0: εφπρθ

Alternative stilistiche 1: εφπρθ

Funzionalità multiple: Hello 0123456789 εφπρθ

L'elenco completo delle funzionalità del font OpenType è consultabile qui: `https://www.microsoft.com/typography/otspec/featurelist.htm`

Per identificare le funzionalità dei font OpenType leggere qui: `http://lists.gnu.org/archive/html/lilypond-devel/2017-08/msg00004.html`

Sebbene sia semplice passare a un tipo di carattere preconfigurato, è anche possibile usare altri tipi, come viene spiegato nelle sezioni successive: [Tipi di carattere per singolo oggetto], pagina 264, e [Tipi di carattere per l'intero documento], pagina 264.

## Vedi anche

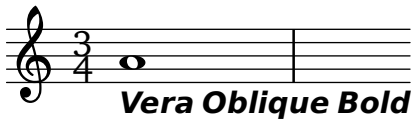
Guida alla notazione: Sezione A.8 [Il font Emmentaler], pagina 679, [Notazione musicale nel blocco markup], pagina 257, Sezione 5.4.9 [Rotazione degli oggetti], pagina 645, [Scelta del tipo di carattere e della dimensione], pagina 248, Sezione A.11.1 [Font], pagina 707.

### Tipi di carattere per singolo oggetto

Si può usare nella partitura qualsiasi tipo di carattere che sia installato nel sistema operativo e riconosciuto da FontConfig, usando la seguente sintassi:

```
\override Staff.TimeSignature.font-name = "Bitstream Charter"
\override Staff.TimeSignature.font-size = #2
\time 3/4

a'1_\markup {
  \override #'(font-name . "Bitstream Vera Sans,sans-serif, Oblique Bold")
    { Vera Oblique Bold }
}
```




*font-name* può essere definito da una lista separata da virgola di 'font' e una lista separata da spazi di 'stili'. Se il 'font' nella lista è installato e contiene il glifo richiesto, verrà usato, altrimenti sarà usato al suo posto il font *successivo*.

Lanciando lilypond con la seguente opzione si ottiene un elenco di tutti i tipi di carattere disponibili nel sistema operativo:

```
lilypond -dshow-available-fonts
```

### Vedi anche

Guida alla notazione: [Tipi di carattere in dettaglio], pagina 261, [Tipi di carattere per l'intero documento], pagina 264.

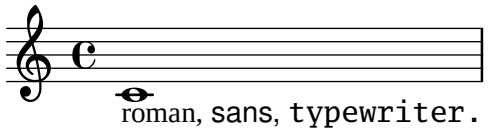
Frammenti: Sezione "Text" in *Frammenti di codice*.

### Tipi di carattere per l'intero documento

È possibile modificare i tipi di carattere usati come tipi predefiniti nelle famiglie *roman*, *sans* e *typewriter* specificandoli, in questo ordine, come è mostrato nell'esempio seguente, che ridimensiona automaticamente i caratteri col valore impostato per la dimensione globale del rigo. In modo analogo a [Tipi di carattere per singolo oggetto], pagina 264, si può indicare con una lista separata da virgole di 'font'. Gli 'stili' dei font, invece, non possono essere definiti. I tipi di carattere sono spiegati in [Tipi di carattere in dettaglio], pagina 261.

```
\paper  {
  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans,Nimbus Sans L"
                          "Luxi Mono"
                          (/ staff-height pt 20)))
}

\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```




Nota: `make-pango-font-tree` reimposta i font della notazione sui predefiniti Emmentaler.

La sintassi seguente permette di cambiare font specifici, lasciando gli altri ai valori predefiniti. L'esempio seguente produce lo stesso risultato dell'esempio precedente di `make-pango-font-tree`. Come con `make-pango-font-tree`, è possibile specificare un elenco separato da virgola di "font" per le categorie `roman`, `sans` e `typewriter`. Se non si cambia la dimensione del rigo dal valore predefinito di 20 pt, `#:factor (/ staff-height pt 20)` non è necessario.

```
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Times New Roman"
     #:sans "Nimbus Sans,Nimbus Sans L"
     #:typewriter "Luxi Mono"
     #:factor (/ staff-height pt 20) ; non necessario se la dimensione del rigo è prede
    ))
}
```

È possibile specificare anche i font della notazione. Il prossimo esempio ha anch'esso lo stesso risultato degli esempi precedenti, perché imposta i font della notazione predefiniti. Maggiori informazioni in Sezione 3.4.4 [Cambiare il tipo di carattere della notazione], pagina 523.

```
\paper {
  #(define fonts
    (set-global-fonts
     #:music "emmentaler"            ; predefinito
     #:brace "emmentaler"            ; predefinito
     #:roman "Times New Roman"
     #:sans "Nimbus Sans,Nimbus Sans L"
     #:typewriter "Luxi Mono"
     #:factor (/ staff-height pt 20) ; non necessario se la dimensione del rigo è prede
    ))
}
```

Nota: ogni chiamata della funzione `set-global-fonts` reimposta completamente i font sia della notazione principale che del testo. Se una categoria non è specificata, allora verrà usato il font predefinito per quella categoria. Ogni chiamata di `set-global-fonts` cambia i font di ciascun blocco `\book` che la segue. Se ci sono molteplici blocchi `\book` e si vogliono usare font diversi per ciascuno, basta ripetere `set-global-fonts`, in questo modo:

```
\paper {
  #(define fonts
    (set-global-fonts
     ...
    ))
}
\book {
  ...
}

\paper {
  #(define fonts
    (set-global-fonts
```

```
      ...
    ))
}
\book {
  ...
}
```

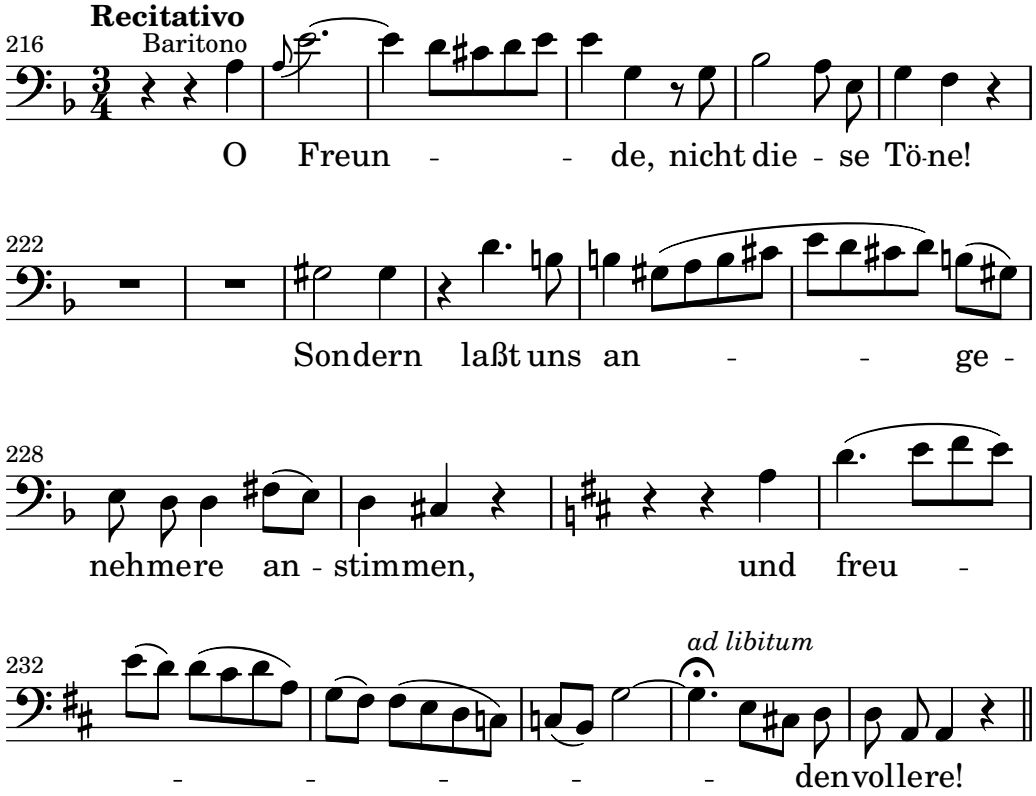
## Vedi anche

Guida alla notazione: [Tipi di carattere in dettaglio], pagina 261, [Tipi di carattere per singolo oggetto], pagina 264, [Scelta del tipo di carattere e della dimensione], pagina 248, Sezione A.11.1 [Font], pagina 707, Sezione 3.4.4 [Cambiare il tipo di carattere della notazione], pagina 523.

# 2 Notazione specialistica

Questo capitolo spiega come creare la notazione musicale per particolari tipi di strumento o per stili specifici.

## 2.1 Musica vocale




Questa sezione spiega come scrivere la musica vocale e assicurarsi che il testo sia allineato con le note della melodia.

### 2.1.1 Notazione comune per la musica vocale

Questa sezione tratta le questioni relative alla maggior parte delle tipologie di musica vocale.

#### Riferimenti per la musica vocale

Questa sezione indica dove trovare informazioni dettagliate sulle questioni comuni a qualsiasi tipo di musica vocale.

- La maggior parte degli stili di musica vocale usa il testo semplice per le parti vocali. Un'introduzione a questo tipo di notazione è disponibile in Sezione "Impostare canzoni semplici" in *Manuale di Apprendimento*.
- La musica vocale richiede solitamente l'uso della modalità `markup`, sia per il testo musicale che per altri elementi testuali (i nomi dei personaggi, etc.). Questa sintassi è spiegata in [Introduzione al testo a margine], pagina 247.
- L'*Ambitus* può essere aggiunto all'inizio dei righi per la voce, come è spiegato in [Ambitus], pagina 36.
- Le indicazioni dinamiche sono di norma posizionate sotto il rigo, ma nella musica corale sono posizionate solitamente sopra il rigo per evitare il testo, come è spiegato in [Struttura di una partitura corale], pagina 308.

## Vedi anche

Glossario musicale: Sezione “ambitus” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Impostare canzoni semplici” in *Manuale di Apprendimento*.

Guida alla notazione: [Introduzione al testo a margine], pagina 247, [Ambitus], pagina 36, [Struttura di una partitura corale], pagina 308.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

## Inserimento del testo vocale

Il testo vocale viene inserito in una speciale modalità di input, che può essere introdotta dai comandi `\lyricmode`, `\addlyrics` o `\lyricsto`. In questa speciale modalità, l'input `d` non viene analizzato come l'altezza *Re*, ma come una sillaba di una lettera. In altre parole, le sillabe si inseriscono come le note, ma le altezze sono sostituite dal testo.

Per esempio:

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

Ci sono due metodi principali per determinare il posizionamento orizzontale delle sillabe. Si può indicare la durata di ogni sillaba esplicitamente, come nell'esempio precedente; oppure si può lasciare che il testo sia allineato automaticamente a una melodia o a un'altra voce del brano, usando `\addlyrics` o `\lyricsto`. Il primo metodo è descritto sotto in [Durate manuali delle sillabe], pagina 274; il secondo in [Durate automatiche delle sillabe], pagina 271.

Una parola o sillaba del testo inizia con un carattere alfabetico (seguito da altri caratteri, vedi dopo) ed è terminata da uno spazio bianco o da un numero. I caratteri successivi al primo nella sillaba possono essere un qualsiasi carattere che non sia un numero o uno spazio bianco.

Dato che qualsiasi carattere che non sia un numero o uno spazio viene considerato come parte della sillaba, una parola è valida anche se finisce con `}`, causando spesso il seguente errore:
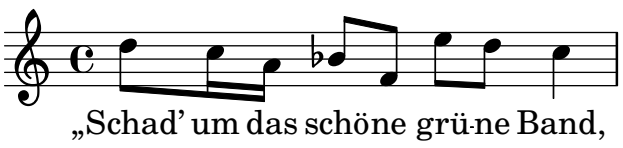
```
\lyricmode { lah lah lah}
```

In questo esempio, la parentesi `}` è inclusa nella sillaba finale, dunque la parentesi iniziale non viene chiusa e la compilazione del file probabilmente non riuscirà. Le parentesi devono quindi essere sempre circondate da uno spazio:

```
\lyricmode { lah lah lah }
```
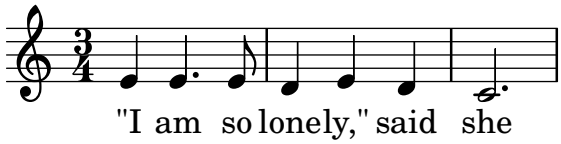
La punteggiatura, i caratteri accentati, quelli di lingue diverse dall'inglese e i caratteri speciali (come il simbolo del cuore o le virgolette oblique) possono essere inseriti direttamente nel file di input, purché sia salvato nella codifica UTF-8. Ulteriori informazioni in Sezione 3.3.3 [Caratteri speciali], pagina 519.

```
\relative { d''8 c16 a bes8 f e' d c4 }
\addlyrics { „Schad’ um das schö -- ne grü -- ne Band, }
```




Le virgolette normali possono essere usate nel testo vocale, ma devono essere precedute da un carattere di barra inversa e l'intera sillaba deve essere racchiusa tra altre virgolette. Per esempio,

```
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone -- "ly,\"" said she }
```




La definizione completa di come può iniziare una parola nella modalità testo è un po' più complessa. Può iniziare con un carattere alfabetico, `_`, `?`, `!`, `:`, `'`, i caratteri di controllo `^A` fino a `^F`, `^Q` fino a `^W`, `^Y`, `^^`, qualsiasi carattere a 8-bit con un codice ASCII superiore a 127 oppure una combinazione a due caratteri di una barra inversa seguita da `` ` ``, `'`, `"` o `^`.

Per avere un maggior controllo sull'aspetto del testo si può usare `\markup` all'interno del testo. Per una spiegazione delle molte opzioni disponibili leggere Sezione 1.8.2 [Formattazione del testo], pagina 246.
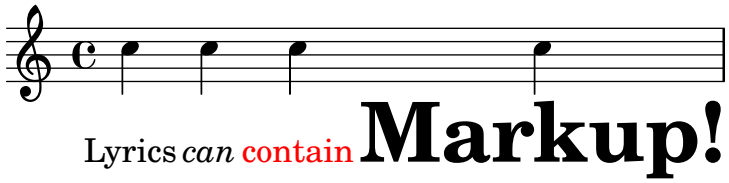
## Frammenti di codice selezionati

*Formattazione delle sillabe del testo vocale*

La modalità markup può essere usata per formattare le singole sillabe del testo vocale.

```
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>
```




## Vedi anche

Manuale di apprendimento: Sezione "Canzoni" in *Manuale di Apprendimento*.

Guida alla notazione: [Durate automatiche delle sillabe], pagina 271, Sezione 1.8.3 [Tipi di carattere], pagina 260, Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione 5.4.1 [Modalità di inserimento], pagina 629, [Durate manuali delle sillabe], pagina 274, Sezione 3.3.3 [Caratteri speciali], pagina 519.

Guida al funzionamento interno: Sezione "LyricText" in *Guida al Funzionamento Interno*.

Frammenti: Sezione "Text" in *Frammenti di codice*.

## Allineamento del testo alla melodia

Il testo vocale viene interpretato in `\lyricmode` e stampato in un contesto `Lyrics`, vedi Sezione 5.1.1 [Tutto sui contesti], pagina 592.

```
\new Lyrics \lyricmode { ... }
```

Due varianti di `\lyricmode` impostano un contesto associato usato per sincronizzare le sillabe del testo con la musica. Il più comodo `\addlyrics` segue immediatamente il contenuto musicale del contesto della voce con cui deve essere sincronizzato, creando implicitamente un contesto Lyrics. Il più versatile `\lyricsto` richiede sia di specificare il contesto della voce associata con un nome sia di creare esplicitamente un contesto Lyrics che contenga il testo. Maggiori dettagli in [Durate automatiche delle sillabe], pagina 271.

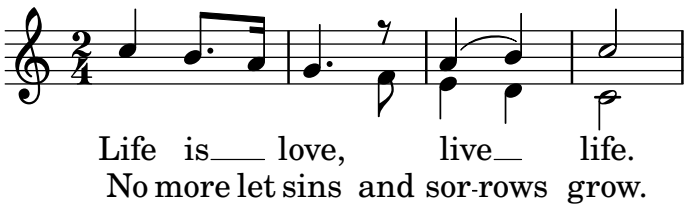
Ci sono due modi per allineare il testo a una melodia:

- Il testo può essere allineato automaticamente in modo che le durate delle sillabe siano prese da un'altra voce o (in circostanze speciali) da una melodia associata, usando `\addlyrics`, `\lyricsto` o impostando la proprietà `associatedVoice`. Ulteriori informazioni in [Durate automatiche delle sillabe], pagina 271.

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e4 d c2
    }
  >>

% prende le durate e l'allineamento dalle note in "one"
  \new Lyrics \lyricsto "one" {
    Life is __ _ love, live __ life.
  }

% prende le durate e l'allineamento dalle note in "one" inizialmente
% poi passa a "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two" % deve essere impostato prima della sillaba
    sins and sor -- rows grow.
  }
>>
```




La prima strofa mostra il modo normale di inserire il testo.

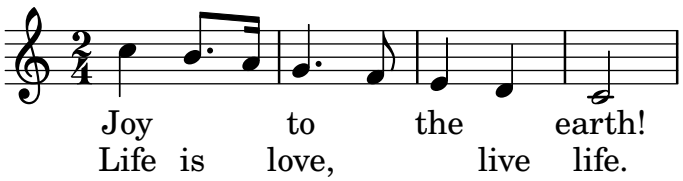
La seconda strofa mostra come cambiare la voce da cui prendere le durate per le sillabe del testo. Ciò è utile se le parole di strofe diverse si distribuiscono lungo le note in modo differente e tutte le durate sono disponibili nei contesti della voce. Ulteriori dettagli in Sezione 2.1.3 [Strofe], pagina 298.

- Il testo può essere allineato indipendentemente dalla durata delle note se le durate delle sillabe vengono specificate esplicitamente e inserite con `\lyricmode`.

```
<<
  \new Voice = "one" \relative {
    \time 2/4
    c''4 b8. a16 g4. f8 e4 d c2
  }
```

```
% usa la durata esplicita precedente di 2;
  \new Lyrics \lyricmode {
    Joy to the earth!
  }

% durate esplicite, impostate su un ritmo diverso
  \new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
  }
>>
```




La prima strofa non è allineata con le note perché le durate non sono state specificate, e il valore precedente di 2 viene usato per ogni parola.

La seconda strofa mostra come le parole possano essere allineate in modo del tutto indipendente dalle note. Ciò è utile se le parole di strofe diverse si distribuiscono lungo le note in modo differente e le durate necessarie non sono disponibili in un contesto musicale. Ulteriori dettagli in [Durate manuali delle sillabe], pagina 274. Questa tecnica è utile anche quando si imposta un dialogo sopra una musica, come si può vedere negli esempi in [Dialogo sopra la musica], pagina 317.

## Vedi anche

Manuale di apprendimento: Sezione “Allineare il testo alla melodia” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.1 [Tutto sui contesti], pagina 592, [Durate automatiche delle sillabe], pagina 271, Sezione 2.1.3 [Strofe], pagina 298, [Durate manuali delle sillabe], pagina 274, [Dialogo sopra la musica], pagina 317, [Durate manuali delle sillabe], pagina 274.

Guida al funzionamento interno: Sezione “Lyrics” in *Guida al Funzionamento Interno*.

## Durate automatiche delle sillabe

Il testo vocale può essere allineato automaticamente alle note di una melodia in tre modi:

- specificando il nome del contesto Voice contenente la melodia con `\lyricsto`,
- introducendo il testo con `\addlyrics` e inserendolo subito dopo il contesto Voice contenente la melodia,
- impostando la proprietà `associatedVoice`, l’allineamento del testo può essere cambiato su un contesto Voice con un nome diverso in qualsiasi momento musicale.
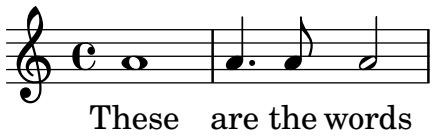
In tutti questi tre metodi è possibile disegnare dei trattini tra le sillabe di una parola e delle linee di estensione oltre la fine di una parola. Maggiori dettagli in [Estensori e trattini], pagina 279.

Il contesto `Voice` contenente la melodia al quale il testo si deve allineare non deve essere “morto”, altrimenti il testo successivo a quel punto verrà perduto. Questo può accadere se ci sono dei momenti in cui quella voce non ha nulla da fare. Metodi per mantenere attivi i contesti sono descritti in Sezione 5.1.3 [Conservazione di un contesto], pagina 598.

### Uso di `\lyricsto`

Il testo vocale può essere allineato a una melodia automaticamente specificando il nome del contesto voce con `\lyricsto`:

```
<<
  \new Voice = "melody" \relative {
    a'1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melody" {
    These are the words
  }
>>
```




In questo modo il testo viene allineato alle note del contesto `Voice` con quel nome, che deve già esistere. Ecco perché di solito il contesto `Voice` viene indicato prima, seguito dal contesto `Lyrics`. Il testo segue il comando `\lyricsto`. Il comando `\lyricsto` invoca la modalità testo automaticamente. Per impostazione predefinita, il testo viene posizionato sotto le note. Altri posizionamenti sono descritti in [Posizionamento verticale del testo], pagina 281.

### Uso di `\addlyrics`

Il comando `\addlyrics` è solo una comoda scorciatoia da usare per evitare di impostare il testo tramite una struttura più complessa.
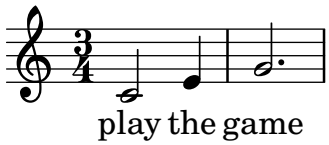
```
{ MUSICA }
\addlyrics { TESTO VOCALE }
```

equivale a

```
\new Voice = "blah" { MUSICA }
\new Lyrics \lyricsto "blah" { TESTO VOCALE }
```

Ecco un esempio,
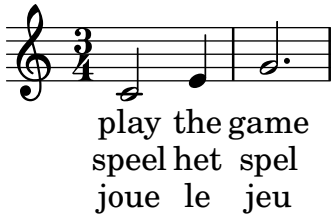
```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}
```




Si possono aggiungere più strofe aggiungendo più sezioni `\addlyrics`:

```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
```

```
}
```




Il comando `\addlyrics` non può gestire le impostazioni polifoniche. Inoltre non può essere usato per associare il testo alla voce per intavolatura (`TabVoice`). In questi casi bisogna usare `\lyricsto`.

## Uso di associatedVoice
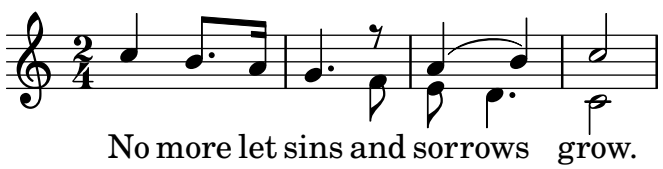
La melodia a cui allineare il testo vocale può essere cambiata impostando la proprietà `associatedVoice`,

```
\set associatedVoice = "lala"
```

Il valore della proprietà (qui: `"lala"`) deve essere il nome di un contesto `Voice`. Per ragioni tecniche, il comando `\set` deve essere posizionato una sillaba prima prima di quella alla quale si riferisce il cambio di voce.

Ecco un esempio che ne dimostra l'utilizzo:

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e8 d4. c2
    }
  >>
% inizialmente prende le note e le durate da "one"
% poi passa a "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two" % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```




## Vedi anche

Guida alla notazione: [Estensori e trattini], pagina 279, Sezione 5.1.3 [Conservazione di un contesto], pagina 598, [Posizionamento verticale del testo], pagina 281.
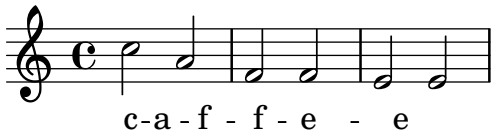
### Durate manuali delle sillabe

In alcune musiche vocali complesse, si potrebbe voler posizionare il testo in modo totalmente indipendente dalle note. In tali casi non bisogna usare `\lyricsto` o `\addlyrics` e nemmeno impostare `associatedVoice`. Le sillabe verranno invece inserite come se fossero note, ma col testo al posto delle altezze, e la durata di ogni sillaba sarà indicata esplicitamente dopo la sillaba.
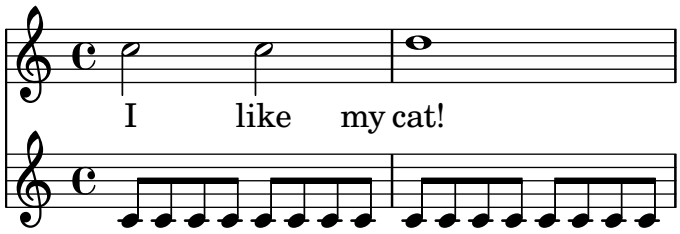
Si possono mostrare le linee tratteggiate tra le sillabe come sempre, mentre le linee di estensione non appaiono se non c'è una voce associata.

Ecco due esempi:

```
<<
  \new Voice = "melody" \relative {
    c''2 a f f e e
  }
  \new Lyrics \lyricmode {
    c4. -- a -- f -- f -- e2. -- e
  }
>>
```
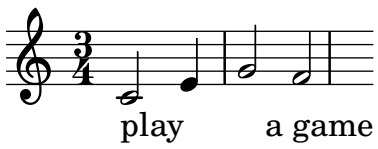



```
<<
  \new Staff {
    \relative {
      c''2 c2
      d1
    }
  }
  \new Lyrics {
    \lyricmode {
      I2 like4. my8 cat!1
    }
  }
  \new Staff {
    \relative {
      c'8 c c c c c c c
      c8 c c c c c c c
    }
  }
>>
```




Questa tecnica è utile quando si scrivono dialoghi , vedi [Dialogo sopra la musica], pagina 317.

Per cambiare l'allineamento delle sillabe, basta impostare la proprietà `self-alignment-X`:

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    c'2 e4 g2 f
  }
  \new Lyrics \lyricmode {
    \override LyricText.self-alignment-X = #LEFT
    play1 a4 game4
  }
>>
```




## Vedi anche

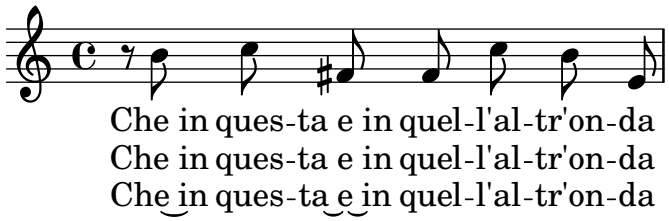
Guida alla notazione: [Dialogo sopra la musica], pagina 317.

Guida al funzionamento interno: Sezione "Lyrics" in *Guida al Funzionamento Interno*, Sezione "Voice" in *Guida al Funzionamento Interno*.

## Più sillabe in una nota

Per assegnare più di una sillaba a una singola nota mantenendo uno spazio tra le sillabe, occorre mettere la frase tra virgolette o usare il carattere `_`. Altrimenti si può usare il simbolo tilde (`~`) per ottenere una legatura di valore per il testo.

```
{
  \relative {
    \autoBeamOff
    r8 b' c fis, fis c' b e,
  }
  \addlyrics
  {
    \override LyricHyphen.minimum-distance = #1.0 % Ensure hyphens are visible
    Che_in ques -- ta_e_in quel -- l'al -- tr'on -- da
  }
  \addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
  \addlyrics { Che~in ques -- ta~e~in quel -- l'al -- tr'on -- da }
}
```




## Vedi anche

Guida al funzionamento interno: Sezione "LyricCombineMusic" in *Guida al Funzionamento Interno*.

### Più note in una sillaba

Talvolta, in particolare nella musica medievale e barocca, molte note vengono cantate in una sillaba; si chiama melisma (vedi Sezione "melisma" in *Glossario Musicale*). La sillaba di un melisma viene solitamente allineata a sinistra della prima nota del melisma.
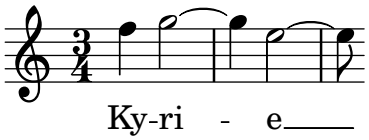
Quando un melisma si trova su una sillaba diversa dall'ultima in una parola, quella sillaba di solito viene collegata a quella successiva con una linea tratteggiata. Ciò si indica inserendo un doppio trattino `--`, subito dopo la sillaba.

Altrimenti, se un melisma si trova sull'ultima o unica sillaba in una parola, solitamente appare una linea di estensione dalla fine della sillaba all'ultima nota del melisma. Ciò si indica inserendo un doppio trattino basso, `__`, subito dopo la parola.

Esistono cinque modi per indicare i melismi:

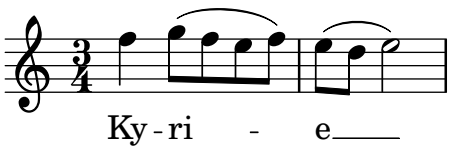
- I melismi vengono creati automaticamente sulle note legate insieme:

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g2 ~ |
    4 e2 ~ |
    8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```




- I melismi possono essere creati automaticamente dalla musica inserendo delle legature di portamento sulle note di ogni melisma. Questo è il modo più comune di inserire il testo:

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```
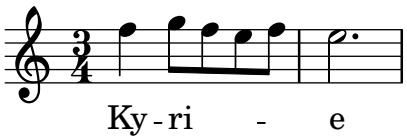



  Attenzione: le legature di frase non causano la creazione di melismi.

- Le note sono considerate un melisma se le loro travature sono disposte manualmente, purché la travatura automatica sia disabilitata. Vedi [Impostare il comportamento delle travature automatiche], pagina 88.
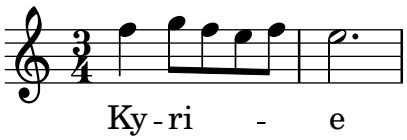
```
<<
```

```
  \new Voice = "melody" \relative {
    \time 3/4
    \autoBeamOff
    f''4 g8[ f e f]
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```




Ovviamente ciò non è adatto ai melismi su note più lunghe degli ottavi.
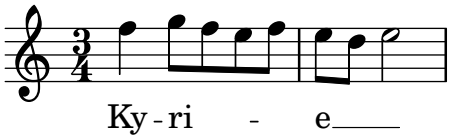
- Un gruppo di note privo di legature sarà trattato come un melisma se sono comprese tra `\melisma` `\melismaEnd`.

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8
    \melisma
    f e f
    \melismaEnd
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```
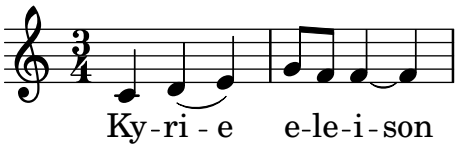



- Un melisma può essere definito interamente nel testo inserendo un solo trattino basso, _, per ogni nota ulteriore da aggiungere al melisma.

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 f e f
    e8 d e2
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ _ _ e __ _ _
  }
>>
```




È possibile avere legature di portamento e di valore e travature manuali nella melodia senza che indichino i melismi. Per farlo si imposta `melismaBusyProperties`:

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] f4 ~ 4
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e e -- le -- i -- son
  }
>>
```
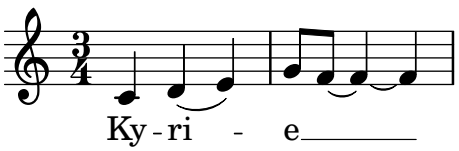



Altre impostazioni di `melismaBusyProperties` possono essere usate per includere o escludere selettivamente legature di valore e di portamento e travature dal rilevamento automatico del melisma; consultare `melismaBusyProperties` in Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

Altrimenti, se si vuole ignorare tutte le indicazioni di melisma, basta impostare `ignoreMelismata` su vero; vedi [Strofe con ritmi diversi], pagina 300.

Se un melisma è necessario in un passaggio in cui `melismaBusyProperties` è attivo, lo si può indicare inserendo un singolo trattino basso nel testo per ogni nota che debba essere inclusa nel melisma:

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] ~ f4 ~ 4
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ e __ _ _ _
  }
>>
```




## Comandi predefiniti

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

### Vedi anche

Glossario musicale: Sezione “melisma” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Allineare il testo alla melodia” in *Manuale di Apprendimento*.

Guida alla notazione: [Allineamento del testo alla melodia], pagina 269, [Durate automatiche delle sillabe], pagina 271, [Impostare il comportamento delle travature automatiche], pagina 88, [Strofe con ritmi diversi], pagina 300.

Guida al funzionamento interno: Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Le linee di estensione sotto i melismi non vengono creati automaticamente; devono essere inseriti a mano con un doppio trattino basso.

### Estensori e trattini

Nell’ultima sillaba di una parola, i melismi sono talvolta indicati con una lunga linea orizzontale che inizia dalla sillaba del melisma e termina in quella successiva. Tale linea viene chiamata linea di estensione e si inserisce con ‘ `__` ’ (notare gli spazi prima e dopo i due caratteri).

> **Nota:** I melismi appaiono nella partitura come linee di estensione, che si inseriscono con un doppio trattino basso; ma i melismi brevi si possono inserire anche saltando note singole but short melismata can also be entered by skipping individual notes, which are entered as single underscore characters; these do not make an extender line to be typeset by default.

Il trattino centrato si inserisce con ‘ `--` ’ tra le sillabe di una stessa parola (notare gli spazi prima e dopo i due caratteri). Il trattino sarà centrato tra le sillabe e la sua lunghezza sarà regolata a seconda dello spazio tra le sillabe.

Nelle partiture compresse i trattini possono essere eliminati. Questo comportamento è controllato da due proprietà di `LyricHyphen`: `minimum-distance` (distanza minima tra le due sillabe) e `minimum-length` (soglia sotto la quale i trattini vengono rimossi).

### Vedi anche

Guida al funzionamento interno: Sezione “LyricExtender” in *Guida al Funzionamento Interno*, Sezione “LyricHyphen” in *Guida al Funzionamento Interno*.
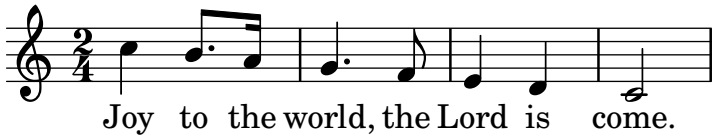
## 2.1.2 Tecniche specifiche per il testo vocale

### Lavorare con testo e variabili

Si possono creare delle variabili contenenti il testo vocale, ma questo deve essere inserito in modalità testo vocale:

```
musicOne = \relative {
  c''4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
```
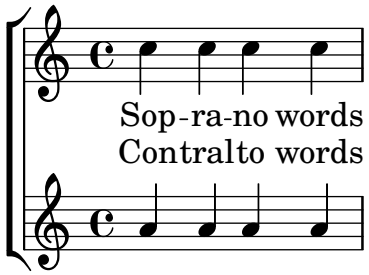
```
    \new Voice = "one" {
      \time 2/4
      \musicOne
    }
    \new Lyrics \lyricsto "one" {
      \verseOne
    }
  >>
}
```




Non è necessario aggiungere le durate se la variabile viene richiamata con `\addlyrics` o `\lyricsto`. Se la musica ha un ordine diverso e più complesso, conviene definire prima le variabili che contengono la musica e il testo, poi impostare la gerarchia di righi e testo, omettendo il testo stesso, e infine aggiungere il testo all'interno di un blocco `\context`. Ciò garantisce che le voci richiamate da `\lyricsto` siano sempre state definite prima. Per esempio:

```
sopranoMusic = \relative { c''4 c c c }
contraltoMusic = \relative { a'4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  >>
}
```




## Vedi anche

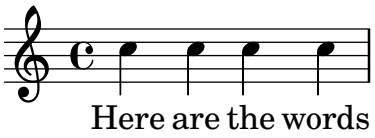
Guida alla notazione: [Posizionamento verticale del testo], pagina 281.

Guida al funzionamento interno: Sezione “LyricCombineMusic” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*.

## Posizionamento verticale del testo

A seconda del tipo di musica, il testo può trovarsi sopra o sotto il rigo oppure tra i righi. Posizionare il testo sotto il rigo associato è il modo più semplice, infatti basta definire il contesto `Lyrics` sotto il contesto `Staff`;

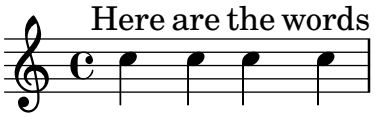
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```




Per posizionare il testo sopra il rigo ci sono due metodi. Il metodo più semplice (e preferito) consiste nell’usare la stessa sintassi precedente e indicare in modo esplicito la posizione del testo:

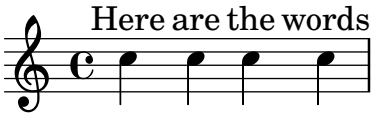
```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
```

```
}
```




Altrimenti si possono usare due passaggi. Prima si dichiara il contesto Lyrics (privo di contenuto) prima dei contesti Staff e Voice, poi il comando `\lyricsto` viene posizionato dopo la voce a cui si riferisce tramite `\context`. Ecco un esempio:

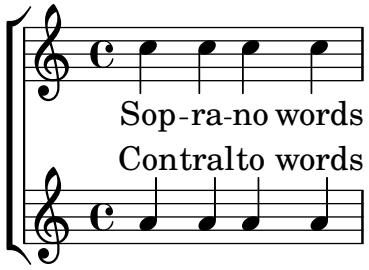
```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % il testo sopra un rigo deve avere questo override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```




Quando ci sono due voci in righi separati, si può posizionare il testo tra i righi usando uno di questi metodi. Ecco un esempio del secondo metodo:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % il testo sopra un rigo deve avere questo override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative { a'4 a a a }
      }
    }
```

```
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        Sop -- ra -- no words
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        Con -- tral -- to words
      }
    }
  >>
}
```




Si possono generare altre combinazioni di testo e righi a partire da questi esempi oppure studiando i modelli nel Manuale di apprendimento, vedi Sezione "Modelli per gruppi vocali" in *Manuale di Apprendimento*.

## Frammenti di codice selezionati

*Ottenere la spaziatura del testo della vecchia versione 2.12*

Il motore di spaziatura verticale è cambiato a partire dalla versione 2.14. Ciò può far sì che il testo vocale abbia un posizionamento diverso.

È possibile impostare delle proprietà dei contesti `Lyric` e `Staff` che facciano sì che il motore di spaziatura si comporti come nella versione 2.12.

```
global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
```
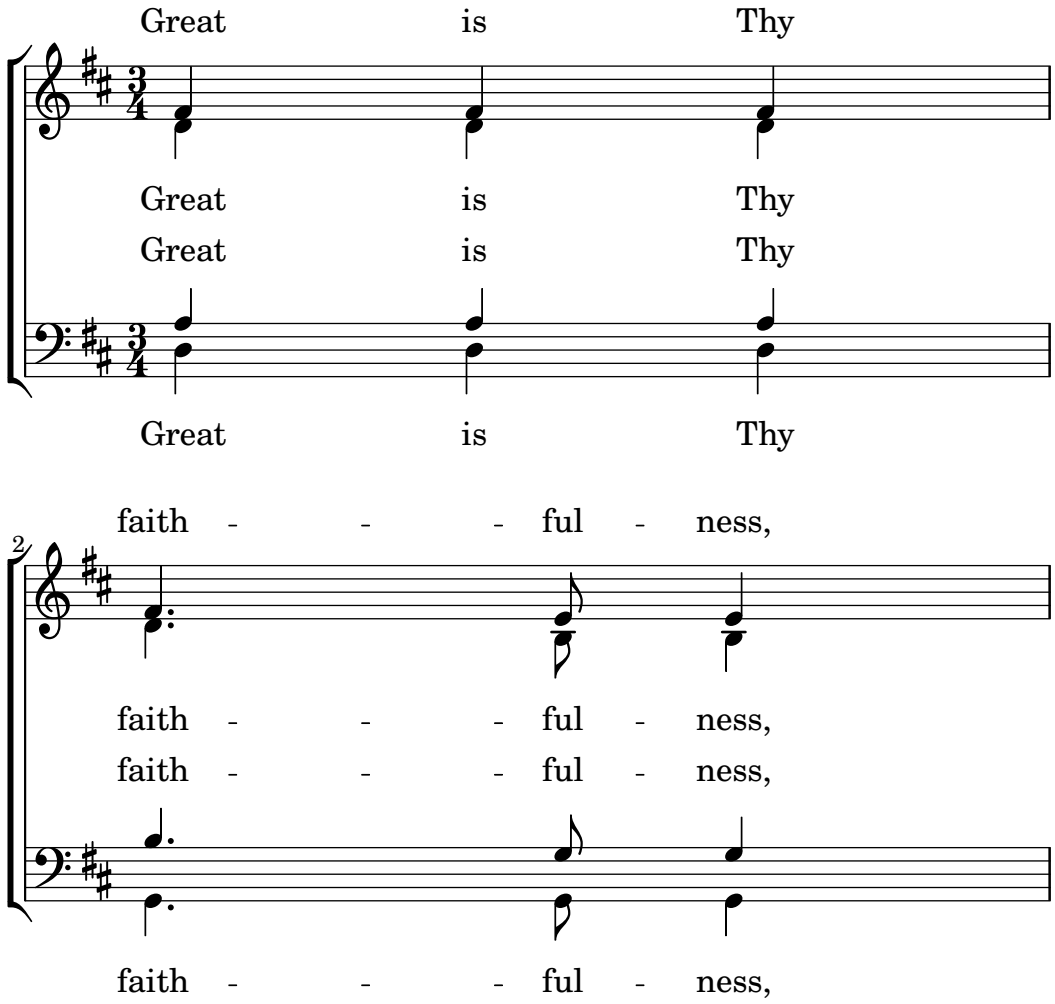
```
}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

words = \lyricmode {
  Great is Thy faith -- ful -- ness,
}

\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo  \global \bassMusic
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup.staff-affinity = ##f
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
    \context {
```

```
      \Staff
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
           (minimum-distance . 2)
           (padding . 2))
    }
  }
}
```




## Vedi anche

Manuale di apprendimento: Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.
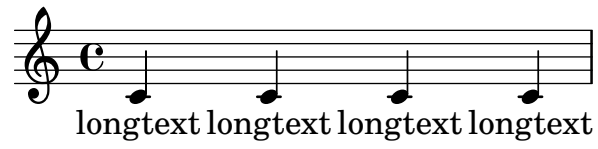
Guida alla notazione: Sezione 5.1.7 [Ordine di disposizione dei contesti], pagina 610, Sezione 5.1.2 [Creazione e citazione di un contesto], pagina 594.
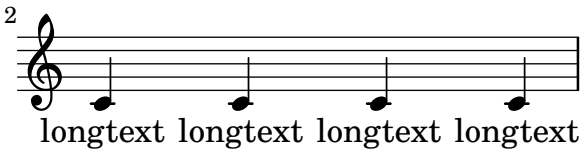
## Posizionamento orizzontale delle sillabe

Per aumentare lo spazio tra le righe del testo, si imposta la proprietà `minimum-distance` di `LyricSpace`.

```
\relative c' {
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = #1.0
```
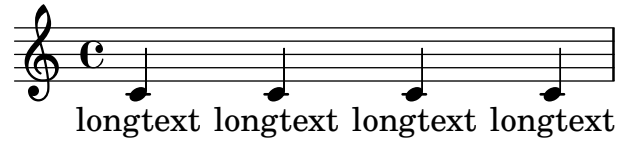
```
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```
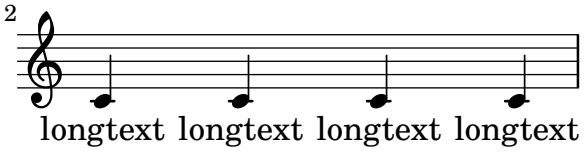






Per applicare questa modifica a tutti i testi della partitura, impostare la proprietà nel blocco `\layout`.

```
\score {
  \relative {
  c' c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace.minimum-distance = #1.0
    }
  }
}
```



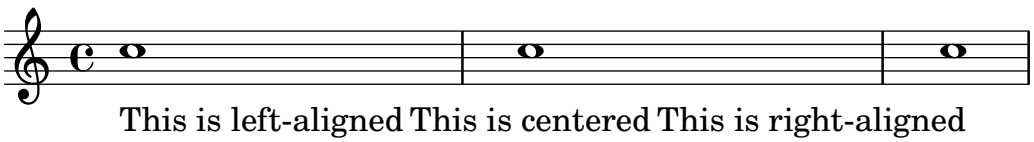



## Frammenti di codice selezionati

*Allineamento del testo vocale*

L'allineamento orizzontale del testo vocale si imposta attraverso la proprietà `self-alignment-X` dell'oggetto `LyricText`. `#-1` è sinistra, `#0` è centro e `#1` è destra; si possono usare anche `#LEFT`, `#CENTER` e `#RIGHT`.

```
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}
```




## Problemi noti e avvertimenti

Verificare che le annotazioni testuali e il testo vocale stiano dentro i margini richiede ulteriori calcoli. Se si desidera velocizzare un po' l'elaborazione, tale funzionalità può essere disabilitata:

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Per far sì che il testo eviti anche le stanghette, usare

```
\layout {
  \context {
    \Lyrics
      \consists "Bar_engraver"
      \consists "Separating_line_group_engraver"
      \hide BarLine
  }
}
```

## Testo e ripetizioni
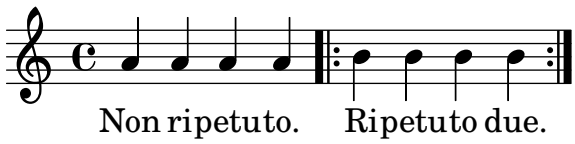
## Ripetizioni semplici

Le ripetizioni in *musica* sono trattate in un'altra sezione: Sezione 1.4 [Ripetizioni], pagina 152. Questa sezione spiega come aggiungere del testo vocale a delle parti musicali ripetute.

Il testo che si riferisce a una sezione musicale ripetuta deve avere lo stesso costrutto di ripetizione della musica, se le parole non sono modificate.
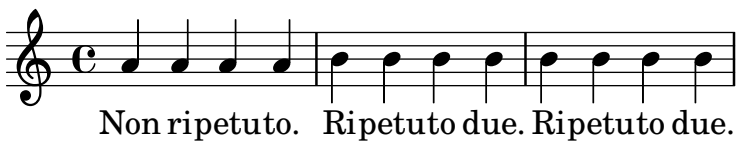
```
\score {
  <<
    \new Staff {
```

```
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Non ri -- petu -- to.
        \repeat volta 2 { Ri -- petu -- to due. }
      }
    }
  >>
}
```




Le parole verranno poi espanse correttamente se le ripetizioni sono ripetute.

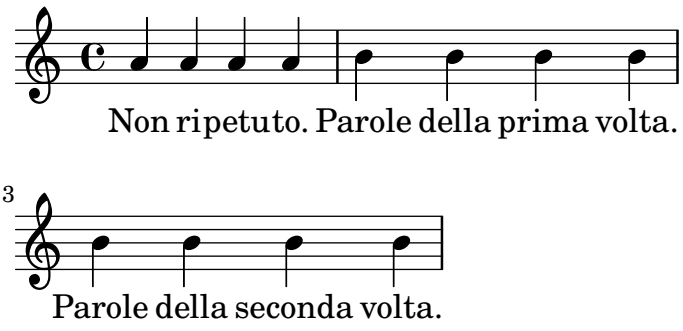
```
\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
          \relative {
            a'4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Non ri -- petu -- to.
          \repeat volta 2 { Ri -- petu -- to due. }
        }
      }
    >>
  }
}
```




Se la sezione ripetuta deve essere ripetuta di nuovo con parole diverse, è sufficiente inserire tutte le parole:
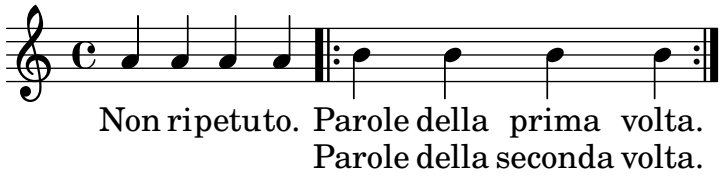
```
\score {
  <<
```

```
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Non ri -- petu -- to.
        Parole della prima volta.
        Parole della seconda volta.
      }
    }
  >>
}
```
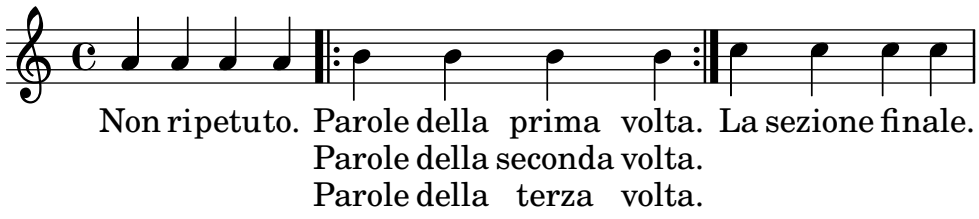



Quando le parole che si riferiscono a una ripetizione sono diverse, le parole di ogni ripetizione devono essere inserite in contesti `Lyrics` separati, annidati correttamente in sezioni parallele:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Non ri -- petu -- to.
      <<
  { Parole della prima volta. }
  \new Lyrics {
    \set associatedVoice = "melody"
    Parole della seconda volta.
  }
      >>
    }
  >>
}
```




Si possono aggiungere più strofe in modo analogo:

```
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
  \relative {
    a'4 a a a
    \repeat volta 3 { b4 b b b }
    c4 c c c
  }
      }
    }
    \new Lyrics \lyricsto "singleVoice" {
      Non ri -- petu -- to.
      <<
  { Parole della prima volta. }
  \new Lyrics {
    \set associatedVoice = "singleVoice"
    Parole della seconda volta.
  }
  \new Lyrics {
    \set associatedVoice = "singleVoice"
    Parole della terza volta.
  }
      >>
      La sezione fi -- nale.
    }
  >>
}
```

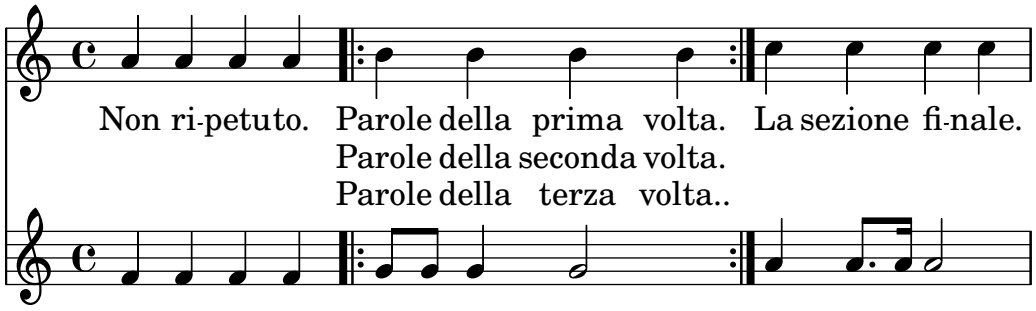



Tuttavia, se questo costrutto si trova all'interno di un contesto multirigo come `ChoirStaff`, il testo della seconda e terza strofa apparirà sotto il rigo inferiore.

Per posizionarli correttamente usare `alignBelowContext`:

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
  \relative {
    a'4 a a a
    \repeat volta 3 { b4 b b b }
    c4 c c c
```

```
  }
      }
    }
    \new Lyrics = "firstVerse" \lyricsto "melody" {
      Non ri -- petu -- to.
      <<
  { Parole della prima volta. }
  \new Lyrics = "secondVerse"
  \with { alignBelowContext = "firstVerse" } {
    \set associatedVoice = "melody"
    Parole della seconda volta.
  }
  \new Lyrics = "thirdVerse"
  \with { alignBelowContext = "secondVerse" } {
    \set associatedVoice = "melody"
    Parole della terza volta..
  }
      >>
      La sezione fi -- nale.
    }
    \new Voice = "harmony" {
      \relative {
  f'4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2
      }
    }
  >>
}
```
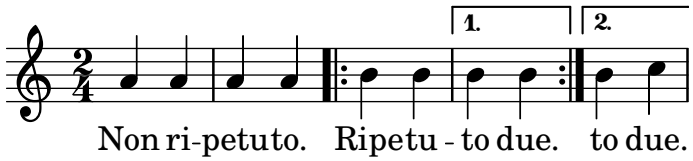



## Ripetizioni con finali alternativi

Se le parole della sezione ripetuta sono le stesse, e nessuno dei finali alternativi inizia con una pausa, si può usare la stessa identica struttura sia per il testo che per la musica. Ciò comporta il vantaggio che `unfoldRepeats` espanderà correttamente sia la musica che il testo vocale.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
  \relative {
    a'4 a a a
    \repeat volta 2 { b4 b }
    \alternative { { b b } { b c } }
  }
```

```
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Non ri -- petu -- to.
  \repeat volta 2 { Ripe -- tu -- }
  \alternative { { to due. } { to due. } }
      }
    }
  >>
}
```
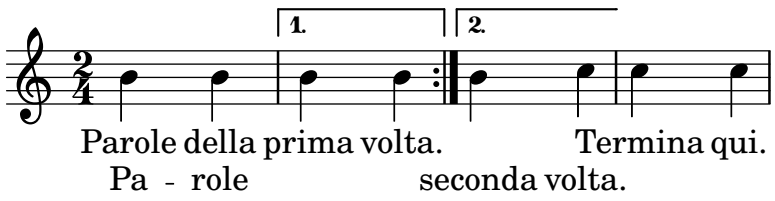



Ma quando la sezione ripetuta ha parole diverse o uno dei blocchi `\alternative` inizia con una pausa, non si può usare il costrutto della ripetizione per le parole e bisogna inserire manualmente i comandi `\skip` per le note delle sezioni alternative prive di un testo corrispondente.

Attenzione: non usare il trattino basso, `_`, per saltare le note, perché il trattino basso indica un melisma e fa sì che la sillaba precedente sia allineata a sinistra.

> **Nota:** Il comando `\skip` deve essere seguito da un numero, ma questo numero viene ignorato se nel testo vocale la durata delle sillabe deriva dalla durata delle note in una melodia associata attraverso `\addlyrics` o `\lyricsto`. Ogni `\skip` salta una singola nota di un qualsiasi valore, senza tener conto del valore del numero che segue.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
  \relative {
    \repeat volta 2 { b'4 b }
    \alternative { { b b } { b c } }
    c4 c
  }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Parole della prima volta.
  \repeat unfold 2 { \skip 1 }
  Termina qui.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Pa -- role
```
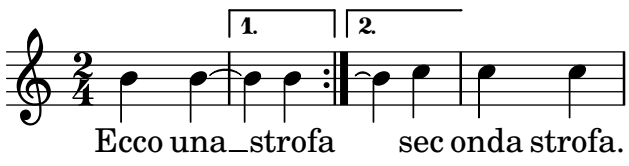
```
  \repeat unfold 2 { \skip 1 }
  seconda volta.
      }
    }
  >>
}
```




Quando una nota è legata verso uno o più finali alternativi, si usa una legatura di valore per portare quella nota nel primo finale alternativo e `\repeatTie` per portarla nel secondo e nei successivi. Questa struttura comporta dei problemi di difficile allineamento quando è presente il testo; si può ottenere un risultato più accettabile aumentando la lunghezza delle sezioni alternative in modo che le note legate siano contenute interamente al loro interno.

La legatura crea un melisma nella prima alternativa, ma non nella seconda e nelle successive, dunque per allineare il testo correttamente occorre disabilitare la creazione automatica dei melismi dopo la sezione delle volte e inserire dei salti manuali.
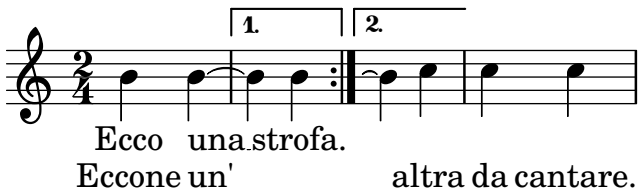
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
  \relative {
    \set melismaBusyProperties = #'()
    \repeat volta 2 { b'4 b ~}
    \alternative { { b b } { b \repeatTie c } }
    \unset melismaBusyProperties
    c4 c
  }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  \repeat volta 2 { Ecco una __ }
  \alternative {
    { \skip 1 strofa }
    { \skip 1 sec }
  }
  onda strofa.
      }
    }
  >>
}
```



Se `\unfoldRepeats` precede una sezione contenente `\repeatTie`, bisogna togliere `\repeatTie` per evitare che appaiano entrambi i tipi di legatura.

Quando la sezione ripetuta ha parola diverse, non si può mettere il testo in un blocco `\repeat` e bisogna inserire manualmente i comandi `\skip`, come si è visto prima.
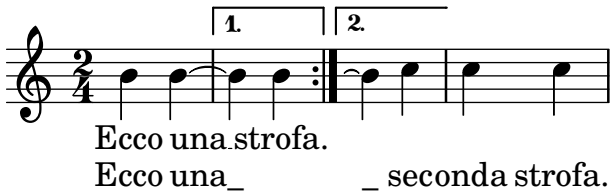
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
  \relative {
    \repeat volta 2 { b'4 b ~}
    \alternative { { b b } { b \repeatTie c } }
    c4 c
  }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Ecco una __ strofa.
  \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Eccone un'
  \repeat unfold 2 { \skip 1 }
  altra da cantare.
      }
    }
  >>
}
```




Se si desidera mostrare gli estensori e i trattini subito prima o dopo un finale alternativo, questi vanno inseriti a mano.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
  \relative {
    \repeat volta 2 { b'4 b ~}
    \alternative { { b b } { b \repeatTie c } }
    c4 c
  }
      }
```

```
    }
    \new Lyrics {
      \lyricsto "melody" {
  Ecco una __ strofa.
  \repeat unfold 2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
  Ecco "una_"
  \skip 1
  "_" sec -- onda  strofa.
      }
    }
  >>
}
```




## Vedi anche
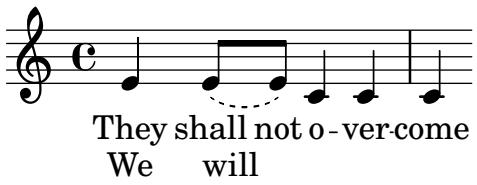
Guida alla notazione: Sezione 5.1.3 [Conservazione di un contesto], pagina 598, Sezione 1.4 [Ripetizioni], pagina 152.

## Testi alternati

Quando solo le parole e le durate di due parti differiscono mentre le altezze sono le stesse, conviene disabilitare temporaneamente la rilevazione automatica dei melismi e indicare il melisma nel testo cantato:
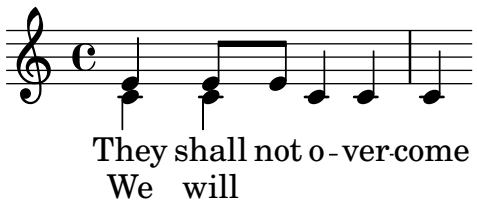
```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
  \set melismaBusyProperties = #'()
  \slurDown
  \slurDashed
  e4 e8 ( e ) c4 c |
  \unset melismaBusyProperties
  c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _
    }
  >>
```

```
}
```




Quando sia la musica che le parole differiscono, è meglio nominare i contesti della voce in cui musica e testo sono diversi e assegnare il testo a quei contesti specifici:

```
\score {
  <<
    \new Voice = "melody" {
      \relative {
  <<
    {
      \voiceOne
      e'4 e8 e
    }
    \new Voice = "splitpart" {
      \voiceTwo
      c4 c
    }
  >>
  \oneVoice
  c4 c |
  c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```



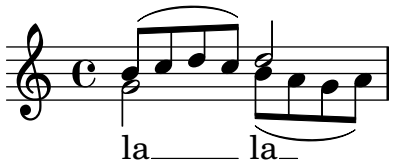

Nella musica corale è comune avere una voce divisa in varie misure. Il costrutto `<< {...} \\ {...} >>`, in cui due (o più) espressioni musicali sono separate dalla doppia barra inversa, potrebbe sembrare il modo giusto di impostare le voci divise. Tuttavia questo costrutto assegna **tutte** le espressioni al suo interno a **NUOVI contesti Voice**, dunque *nessun testo cantato* sarà impostato per esse perché il testo si collegherà al contesto della voce originale. Di norma, non è ciò che si desidera. Il costrutto adatto in questa situazione è il passaggio polifonico temporaneo, spiegato nella sezione *Passaggi polifonici temporanei* in [Polifonia su un solo rigo], pagina 175.

### Polifonia con testo in comune

Quando due voci contenenti durate diverse condividono lo stesso testo cantato, allineare il testo a una delle voci può creare dei problemi nell'altra voce. Per esempio, il secondo estensore del testo nell'esempio seguente è troppo corto, perché il testo è allineato solo alla voce superiore:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
words = \lyricmode { la __ la __ }

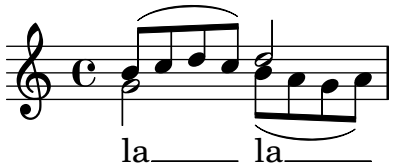
\new Staff <<
  \new Voice = "sopranoVoice" { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new Lyrics \lyricsto "sopranoVoice" \words
>>
```




Per ottenere il risultato desiderato, si allinea il testo a un nuovo contesto `NullVoice` contenente una giusta combinazione delle due voci. Le note del contesto `NullVoice` non appaiono nello spartito, ma servono soltanto ad allineare il testo nel modo corretto:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```

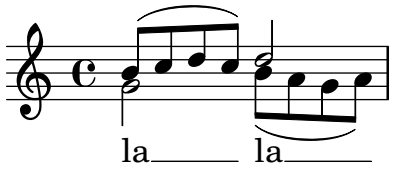



Questo metodo si può usare anche con la funzione `\partCombine`, che di per sé non accetta il testo cantato:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice \partCombine \soprano \alto
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
```

```
>>
```




### Problemi noti e avvertimenti

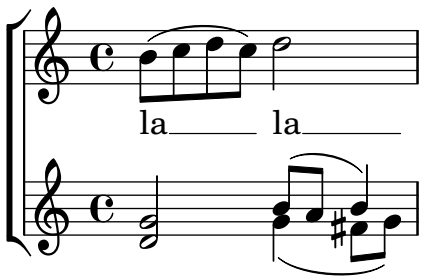
La funzione `\addlyrics` funziona solo con testi vocali collegati a contesti `Voice` e non può essere usata con `NullVoice`.

La funzione `\partCombine` è descritta in [Combinazione automatica delle parti], pagina 185.

Infine, si può usare questo metodo anche quando le voci si trovano su righi diversi e non è limitata a due sole voci:

```
soprano = \relative { b'8( c d c) d2 }
altoOne = \relative { g'2 b8( a b4) }
altoTwo = \relative { d'2 g4( fis8 g) }
aligner = \relative { b'8( c d c) d( d d d) }
words = \lyricmode { la __ la __ }

\new ChoirStaff \with {\accepts NullVoice } <<
  \new Staff <<
    \soprano
    \new NullVoice = "aligner" \aligner
  >>
  \new Lyrics \lyricsto "aligner" \words
  \new Staff \partCombine \altoOne \altoTwo
>>
```

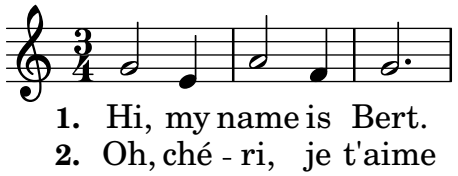



## 2.1.3 Strofe

### Aggiungere i numeri di strofa

I numeri di strofa si aggiungono impostando `stanza`:

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = "1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = "2. "
  Oh, ché -- ri, je t'aime
}
```



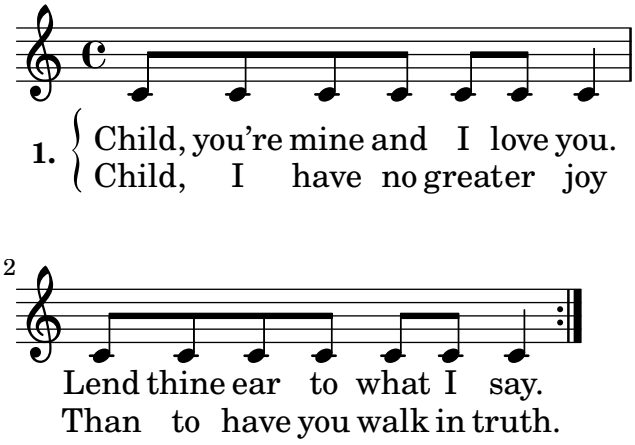

Questi numeri appaiono prima dell'inizio della prima sillaba. Due linee di una strofa possono anche essere raggruppate insieme, per esempio in caso di una ripetizione con testo diverso:

```
leftbrace = \markup {
  \override #'(font-encoding . fetaBraces)
  \lookup "brace80"
}

stanzaOneOne = \lyricmode {
  \set stanza = \markup {
    \column { \vspace #.33 "1. "}
    \leftbrace
  }
  Child, you're mine and I love you.
  Lend thine ear to what I say.
}

stanzaOneThree = \lyricmode {
  Child, I have no great -- er joy
  Than to have you walk in truth.
}

\new Voice {
  \repeat volta 2 {
    c'8 c' c' c' c' c' c'4
    c'8 c' c' c' c' c' c'4
  }
}
\addlyrics { \stanzaOneOne }
\addlyrics { \stanzaOneThree }
```



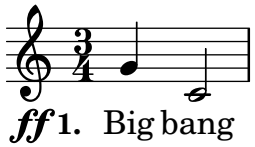

## Aggiungere le dinamiche alle strofe

Le strofe che hanno un volume diverso possono essere indicate con un segno di dinamica all'inizio di ogni strofa. In LilyPond, tutto ciò che si trova di fronte a una strofa va nell'oggetto

`StanzaNumber`; lo stesso vale per i segni di dinamica. Per ragioni tecniche, bisogna impostare la strofa fuori da `\lyricmode`:

```
text = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
}

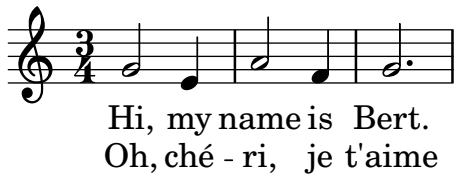
<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \text
>>
```




## Aggiungere i nomi dei cantanti alle strofe

Si possono aggiungere anche i nomi dei cantanti. Appariranno all'inizio del rigo, proprio come per i nomi degli strumenti. Si creano impostando `vocalName`. Una versione abbreviata si inserisce con `shortVocalName`.

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = "Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = "Ernie "
  Oh, ché -- ri, je t'aime
}
```




## Strofe con ritmi diversi
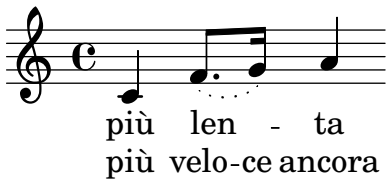
Spesso, strofe diverse di una canzone sono collegate a una melodia in modi leggermente diversi. Tali variazioni possono essere colte con `\lyricsto`.

## Ignorare i melismi

Può capitare ad esempio che il testo abbia un melisma in una strofa, ma varie sillabe in un'altra. Una possibile soluzione consiste nel far sì che la voce più veloce ignori il melisma, impostando `ignoreMelismata` nel contesto Lyrics.

```
<<
```

```
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    più len -- ta
  }
  \new Lyrics \lyricsto "lahlah" {
    più
    \set ignoreMelismata = ##t
    velo -- ce
    \unset ignoreMelismata
    ancora
  }
>>
```




## Problemi noti e avvertimenti
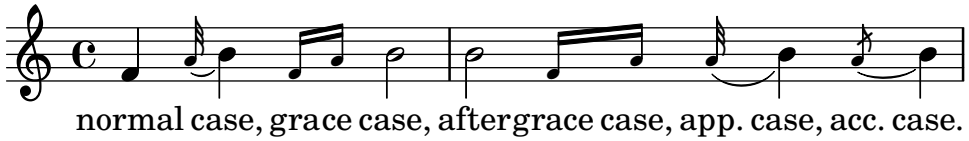
Diversamente dalla maggior parte dei comandi `\set`, `\set ignoreMelismata` non funziona se preceduto da `\once`. Bisogna usare `\set` e `\unset` per contrassegnare il testo in cui il melisma deve essere ignorato.

## Aggiungere le sillabe agli abbellimenti

Per impostazione predefinita, gli abbellimenti (ovvero le note inserite con `\grace`) non sono assegnati alle sillabe quando si usa `\lyricsto`, ma tale comportamento può essere modificato:

```
<<
  \new Voice = melody \relative {
    f'4 \appoggiatura a32 b4
    \grace { f16 a16 } b2
    \afterGrace b2 { f16[ a16] }
    \appoggiatura a32 b4
    \acciaccatura a8 b4
  }
  \new Lyrics
  \lyricsto melody {
    normal
    \set includeGraceNotes = ##t
    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
    acc. case.
```
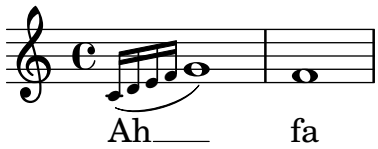
```
  }
>>
```




## Problemi noti e avvertimenti

Come per `associatedVoice`, `includeGraceNotes` deve essere impostato al più tardi una sillaba prima di quella da mettere sotto un abbellimento. Per il caso di un abbellimento proprio all'inizio di un brano, meglio usare un blocco `\with` o `\context`:

```
<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

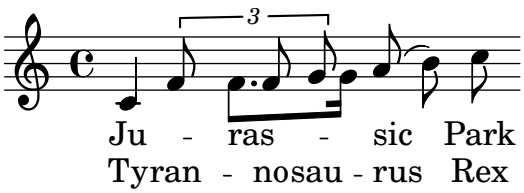



## Passare a una melodia alternativa

Sono possibili variazioni più complesse nell'impostare testo e musica. La melodia su cui è impostato il testo può essere modificata all'interno del contesto del testo impostando la proprietà `associatedVoice`:

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    <<
      \new Voice = "alternative" {
  \voiceOne
  \tuplet 3/2 {
    % mostra chiaramente le associazioni.
    \override NoteColumn.force-hshift = #-3
    f8 f g
  }
      }
      {
  \voiceTwo
  f8.[ g16]
  \oneVoice
      } >>
```

```
  a8( b) c
}
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
\new Lyrics \lyricsto "lahlah" {
  % Complicato: bisogna impostare associatedVoice
  % una sillaba prima di quella cui si applica!
  \set associatedVoice = "alternative" % si applica a "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = "lahlah" % si applica a "rus"
  sau -- rus Rex
} >>
```




Il testo per la prima strofa viene impostato sulla melodia 'lahlah' nel solito modo, ma la seconda strofa è impostata inizialmente sul contesto `lahlah` e passa poi alla melodia `alternative` per le sillabe da 'ran' a 'sau':

```
\set associatedVoice = "alternative" % si applica a "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % si applica a "rus"
sau -- rus Rex
```

In questo esempio `alternative` è il nome del contesto `Voice` contenente la terzina.

Attenzione al posizionamento del comando `\set associatedVoice`: appare una sillaba troppo presto, ma ciò è corretto.

> **Nota:** Il comando `\set associatedVoice` deve essere inserito una sillaba *prima* di quella in cui deve verificarsi il passaggio alla nuova voce. In altre parole, il passaggio alla voce associata accade una sillaba dopo quella che ci si aspetterebbe. Ciò è dovuto a ragioni tecniche e non è un difetto di LilyPond.

## Stampare le strofe alla fine

Talvolta si allinea una sola strofa alla musica e le strofe rimanenti appaiono in forma di versi alla fine del brano. Per ottenere ciò si aggiungono le strofe ulteriori in un blocco `\markup` esterno al blocco della partitura. Nell'esempio seguente si possono notare due modi di forzare le interruzioni di linea in un blocco `\markup`.

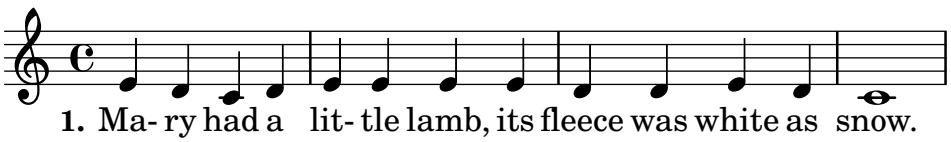
```
melody = \relative {
e' d c d | e e e e |
d d e d | c1 |
}
```

```
text = \lyricmode {
\set stanza = "1." Ma- ry had a lit- tle lamb,
its fleece was white as snow.
}

\score {
  <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
}
\markup {
  \column{
    \line { Verse 2. }
    \line { All the children laughed and played }
    \line { To see a lamb at school. }
  }
}
\markup {
  \wordwrap-string "
  Verse 3.

  Mary took it home again,

  It was against the rule."
}
```




Verse 2.
All the children laughed and played
To see a lamb at school.

Verse 3.
Mary took it home again,
It was against the rule.

## Stampare le strofe alla fine in molteplici colonne

Quando un brano ha molte strofe, queste sono spesso stampate in molteplici colonne lungo la pagina. Un numero di strofa rientrato spesso introduce ciascuna strofa. L'esempio seguente mostra come riprodurre questo output in LilyPond.
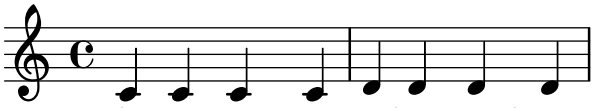
```
melody = \relative {
  c'4 c c c | d d d d
}
```

```
text = \lyricmode {
  \set stanza = "1."
  This is verse one.
  It has two lines.
}

\score {
  <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
}

\markup {
  \fill-line {
    \hspace #0.1 % sposta la colonna in avanti rispetto al margine sinistro;
     % può essere tolto se lo spazio nella pagina è stretto
     \column {
      \line { \bold "2."
  \column {
    "This is verse two."
    "It has two lines."
  }
      }
      \combine \null \vspace #0.1 % aggiunge spazio verticale tra le strofe
      \line { \bold "3."
  \column {
    "This is verse three."
    "It has two lines."
  }
      }
    }
    \hspace #0.1 % aggiunge spazio orizzontale tra le colonne;
    \column {
      \line { \bold "4."
  \column {
    "This is verse four."
    "It has two lines."
  }
      }
      \combine \null \vspace #0.1 % aggiunge spazio verticale tra le strofe
      \line { \bold "5."
  \column {
    "This is verse five."
    "It has two lines."
  }
      }
    }
  \hspace #0.1 % dà ulteriore spazio sul margine destro;
  % può essere tolto se lo spazio nella pagina è stretto
  }
```

```
}
```



**1.** This is verse one. It has two lines.

**2.** This is verse two.
It has two lines.

**3.** This is verse three.
It has two lines.

**4.** This is verse four.
It has two lines.

**5.** This is verse five.
It has two lines.

### Vedi anche

Guida al funzionamento interno: Sezione “LyricText” in *Guida al Funzionamento Interno*, Sezione “StanzaNumber” in *Guida al Funzionamento Interno*.

## 2.1.4 Canzoni

### Riferimenti per canzoni

Le canzoni si scrivono solitamente su tre righi: la melodia per il cantante nel rigo superiore e due righi per l’accompagnamento di pianoforte in basso. Il testo della prima strofa appare immediatamente sotto il rigo superiore. Se ci sono solo poche altre strofe, queste possono essere poste subito sotto la prima; ma se ci sono più strofe di quante ne possano essere contenute lì la seconda strofa e le successive vengono stampate dopo la musica come testo separato.

Tutti gli elementi della notazione necessari per scrivere canzoni sono descritti dettagliatamente in altre parti della documentazione:

- Per costruire la struttura del rigo: Sezione 1.6.1 [Aspetto del rigo], pagina 193.
- Per scrivere la musica per pianoforte: Sezione 2.2 [Tastiera e altri strumenti multirigo], pagina 331.
- Per scrivere il testo da associare a una linea melodica: Sezione 2.1.1 [Notazione comune per la musica vocale], pagina 267.
- Per posizionare il testo: [Posizionamento verticale del testo], pagina 281.
- Per inserire le strofe: Sezione 2.1.3 [Strofe], pagina 298.
- Le canzoni hanno spesso i nomi degli accordi che appaiono sopra i righi. Questo argomento è trattato in Sezione 2.7.2 [Visualizzazione accordi], pagina 424.
- Per stampare i diagrammi degli accordi per l’accompagnamento per chitarra o altri strumenti a tasti: “Fret diagram markups” in ⟨undefined⟩ [Common notation for fretted strings], pagina ⟨undefined⟩.

### Vedi anche

Manuale di apprendimento: Sezione “Canzoni” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.1.1 [Notazione comune per la musica vocale], pagina 267, Sezione 2.7.2 [Visualizzazione accordi], pagina 424, Sezione 1.6.1 [Aspetto del rigo], pagina 193, Sezione 2.2 [Tastiera e altri strumenti multirigo], pagina 331, [Posizionamento verticale del testo], pagina 281, Sezione 2.1.3 [Strofe], pagina 298.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

### Canzonieri

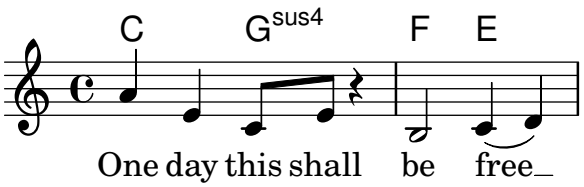
I canzonieri (in inglese *lead sheet*) si ottengono combinando le parti vocali con la 'modalità per accordi'; la sintassi è spiegata in Sezione 2.7 [Notazione per accordi], pagina 419.

### Frammenti di codice selezionati

*Canzoniere semplice*

Mettendo insieme nomi degli accordi, melodia e testo si ottiene un canzoniere (in inglese "lead sheet"):

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```




### Vedi anche

Guida alla notazione: Sezione 2.7 [Notazione per accordi], pagina 419.

## 2.1.5 Musica corale

Questa sezione tratta le questioni che hanno a che fare più direttamente con la musica corale, compresi inni, cori polifonici, oratori, etc.

### Riferimenti per musica corale

La musica corale viene solitamente rappresentata con due, tre o quattro righi in un gruppo `ChoirStaff`. L'accompagnamento, se richiesto, è posto sotto in un gruppo `PianoStaff`, che viene solitamente rimpicciolito per le prove di opere corali *a cappella*. Le note di ogni parte vocale sono inserite in un contesto `Voice` e a ogni rigo viene assegnato una sola parta vocale (ovvero un contesto `Voice`) o una coppia di parti vocali (due contesti `Voice`).

Le parole vengono poste nei contesti `Lyrics`, o sotto ciascun rigo musicale corrispondente oppure una strofa sopra e una sotto il rigo musicale se questo contiene musica per due parti.

Vari argomenti comuni nella musica corale sono trattati dettagliatamente in altre sezioni della documentazione:

- Un'introduzione alla creazione di una partitura vocale SATB si trova nel manuale di apprendimento: Sezione "Partitura vocale a quattro parti SATB" in *Manuale di Apprendimento*. È disponibile anche un modello integrato che semplifica la scrittura di musica vocale SATB: Sezione "Modelli integrati" in *Manuale di Apprendimento*.
- Vari modelli adatti per vari stili di musica corale si trovano anche nel manuale di apprendimento: Sezione "Modelli per gruppi vocali" in *Manuale di Apprendimento*.
- Le informazioni relative a `ChoirStaff` e `PianoStaff` si trovano in [Raggruppare i righi], pagina 195.

- Le teste di nota a forma variabile, come quelle usate nello stile Sacred Harp e notazione simile, sono descritte in ⟨undefined⟩ [Shape note heads], pagina ⟨undefined⟩.
- Quando due parti vocali si trovano su uno stesso rigo, i gambi, le legature, etc. della parte più alta sono dirette in su e quelle della parte più bassa sono rivolte in giù. Per farlo si usa `\voiceOne` e `\voiceTwo`, come è spiegato in [Polifonia su un solo rigo], pagina 175.
- Quando una parte vocale si divide temporaneamente, si devono usare i *passaggi polifonici temporanei* (vedi [Polifonia su un solo rigo], pagina 175).

## Comandi predefiniti

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

## Vedi anche

Manuale di apprendimento: Sezione “Partitura vocale a quattro parti SATB” in *Manuale di Apprendimento*, Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.7 [Ordine di disposizione dei contesti], pagina 610, [Raggruppare i righi], pagina 195, ⟨undefined⟩ [Shape note heads], pagina ⟨undefined⟩, [Polifonia su un solo rigo], pagina 175.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

## Struttura di una partitura corale

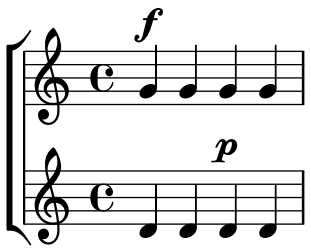
La musica corale contenente quattro righi, con o senza accompagnamento del pianoforte, viene solitamente disposta in due sistemi per pagina. A seconda della dimensione della pagina, ottenere ciò può richiedere modifiche a varie impostazioni predefinite. Occorre considerare le seguenti impostazioni:

- La dimensione globale del rigo può essere modificata per cambiare la dimensione degli elementi della partitura. Vedi Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.
- Le distanze tra i sistemi, i righi e il testo cantato possono essere tutte regolate in modo indipendente. Vedi Sezione 4.4 [Spaziatura verticale], pagina 561.
- Le dimensioni delle variabili di disposizione verticale possono essere mostrate per facilitare la regolazione della spaziatura verticale. Questa e altre possibilità per far entrare la musica in meno pagine sono descritte in Sezione 4.6 [Riduzione del numero di pagine di una partitura], pagina 588.
- Se il numero dei sistemi per pagina cambia da uno a due, è uso comune indicare ciò con un separatore di sistema tra i due sistemi. Vedi [Separare i sistemi], pagina 200.
- Per maggiori informazioni sulle proprietà di formattazione della pagina, leggere Sezione 4.1 [Formattazione della pagina], pagina 539.

Le indicazioni dinamiche sono poste, per impostazione predefinita, sotto il rigo, ma nella musica corale sono poste solitamente sopra il rigo per evitare il testo cantato. Il comando predefinito `\dynamicUp` permette di ottenere ciò per le indicazioni dinamiche in un singolo contesto `Voice`. Se ci sono molti contesti `Voice`, questo comando dovrebbe essere posto in ciascuno di essi. Ma esiste la possibilità di usare una sola volta la sua forma estesa, che sposta tutte le indicazioni dinamiche dell’intera partitura sopra i loro rispettivi righi, come mostrato in questo esempio:

```
\score {
  \new ChoirStaff <<
    \new Staff {
```

```
      \new Voice {
  \relative { g'4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
  \relative { d'4 d d\p d }
      }
    }
  >>
  \layout {
    \context {
      \Score
      \override DynamicText.direction = #UP
      \override DynamicLineSpanner.direction = #UP
    }
  }
}
```




## Comandi predefiniti

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

## Vedi anche

Guida alla notazione: Sezione 4.6.2 [Modificare la spaziatura], pagina 590, Sezione 4.6.1 [Visualizzare la spaziatura], pagina 589, Sezione 4.6 [Riduzione del numero di pagine di una partitura], pagina 588, Sezione 4.1 [Formattazione della pagina], pagina 539, Sezione 4.2 [Formattazione della partitura], pagina 550, [Separare i sistemi], pagina 200, Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552, Sezione 4.3 [Interruzioni], pagina 554, Sezione 4.4 [Spaziatura verticale], pagina 561.

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

## Frammenti di codice selezionati

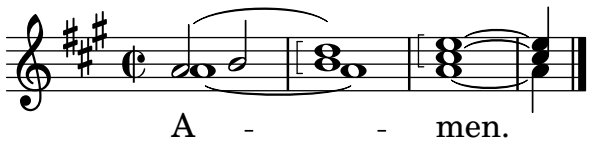
*Usare arpeggioBracket per rendere i divisi più visibili*

Si può usare `arpeggioBracket` per indicare la divisione delle voci quando non ci sono gambi che forniscano questa informazione. Questo caso è frequente nella musica corale.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
```

```
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
          <b d>1\arpeggio)
          <cs e>\arpeggio ~
          <cs e>4
        }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|."
      }
    >>
  }
  \layout { ragged-right = ##t }
}
```




### Vedi anche

Guida alla notazione: Sezione 1.3.3 [Indicazioni espressive lineari], pagina 142.

## 2.1.6 Opera e musical

La musica, il testo cantato e i dialoghi delle opere e dei musical sono di solito impostati in una o più delle seguenti forme:

- Una *partitura del direttore* contenente l'intera partitura orchestrale e le parti vocali, insieme a un libretto con le battute d'entrata se ci sono dei passaggi parlati.
- Le *parti orchestrali* contenenti la musica degli strumenti individuali dell'orchestra o della band.
- Una *partitura vocale* contenente tutte le parti vocali con accompagnamento del pianoforte. L'accompagnamento di solito è una riduzione orchestrale e in questo caso il nome dello strumento orchestrale originale viene spesso indicato. Le partiture vocali talvolta hanno anche le didascalie e il libretto con le battute d'entrata.
- Un *libro vocale* contenente soltanto le parti vocali (nessun accompagnamento), talvolta insieme al libretto.
- Un *libretto* contenente i passaggi estesi dei dialoghi parlati solitamente presenti nei musical, insieme alle parole delle parti cantate. Le didascalie sono solitamente incluse. È possibile usare LilyPond per creare i libretti, ma dato che non contengono musica è preferibile usare altri metodi.

Le sezioni della documentazione di LilyPond che trattano gli argomenti necessari per creare partiture negli stili più comuni nell'opera e nei musical sono indicati nei riferimenti seguenti. A

questi seguono sezioni che trattano le tecniche specifiche che sono peculiari per le partiture di opera e musical.

## Riferimenti per opera e musical

- La partitura di un direttore d'orchestra contiene molti righi e testi vocali raggruppati. I modi per raggruppare i righi sono mostrati in [Raggruppare i righi], pagina 195. Per annidare gruppi di righi leggere [Gruppi di righi annidati], pagina 198.
- La stampa di righi vuoti nelle partiture musicali e vocali dei direttori è solitamente soppressa. Per creare tale partitura leggere [Nascondere i righi], pagina 208.
- La scrittura di parti orchestrali è trattata in Sezione 1.6.3 [Scrittura delle parti], pagina 210. Altre sezioni del capitolo Notazione specialistica potrebbero essere rilevanti, a seconda dell'orchestrazione usata. Molti strumenti sono strumenti traspositori, vedi [Trasporto strumentale], pagina 27.
- Se il numero di sistemi per pagina cambia da pagina a pagina, è di uso comune separare i sistemi con un segno separatore di sistemi. Vedi [Separare i sistemi], pagina 200.
- Per maggiori informazioni sulle proprietà di formattazione della pagina leggere Sezione 4.1 [Formattazione della pagina], pagina 539.
- Si possono inserire suggerimenti di dialogo, didascalie e note a pié di pagina, vedi Sezione 3.2.4 [Creazione di note a piè di pagina], pagina 499, e Sezione 1.8 [Testo], pagina 238. Didascalie estese possono essere aggiunte anche con una sezione di markup indipendenti tra i due blocchi `\score`, vedi [Testo separato], pagina 245.

## Vedi anche

Glossario musicale: Sezione "Partitura senza righi vuoti" in *Glossario Musicale*, Sezione "Rigo temporaneo" in *Glossario Musicale*, Sezione "Strumento traspositore" in *Glossario Musicale*.

Guida alla notazione: Sezione 3.2.4 [Creazione di note a piè di pagina], pagina 499, [Raggruppare i righi], pagina 195, [Nascondere i righi], pagina 208, [Trasporto strumentale], pagina 27, [Gruppi di righi annidati], pagina 198, Sezione 4.1 [Formattazione della pagina], pagina 539, [Separare i sistemi], pagina 200, [Trasposizione], pagina 11, Sezione 1.6.3 [Scrittura delle parti], pagina 210, Sezione 1.8.1 [Inserimento del testo], pagina 239.
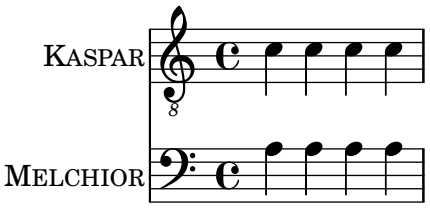
Frammenti: Sezione "Vocal music" in *Frammenti di codice*.

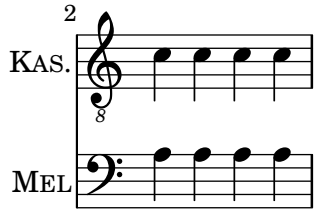
## Nomi dei personaggi

I nomi dei personaggi sono solitamente mostrati a sinistra del rigo quando il rigo è dedicato a quel personaggio soltanto:

```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative {
  \clef "G_8"
  c'4 c c c
  \break
  c4 c c c
      }
    }
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Melchior
```
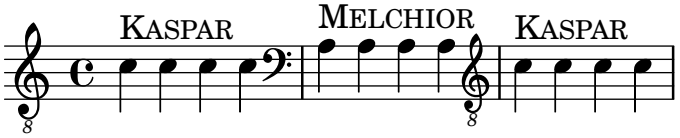
```
      \set Staff.shortVocalName = \markup \smallCaps Mel
      \clef "bass"
      \relative {
  a4 a a a
  a4 a a a
      }
    }
  >>
}
```




Quando due o più personaggi condividono lo stesso rigo, il nome del personaggio è solitamente collocato sopra il rigo all'inizio di ogni sezione appartenente a quel personaggio. È possibile fare ciò con i markup. Spesso si usa un tipo di carattere preciso a questo scopo.

```
\relative c' {
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
  \clef "bass"
  a4^\markup \fontsize #1 \smallCaps Melchior
  a a a
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
}
```
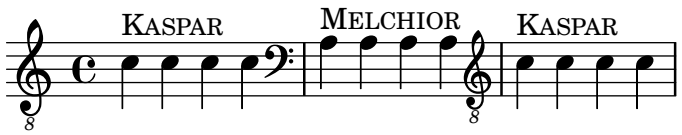



Altrimenti, se ci sono molti cambi di personaggi, è più semplice impostare una variabile per salvare le definizioni di ogni personaggio, in modo che il cambio di personaggio possa essere indicato in modo facile e conciso.

```
kaspar = {
  \clef "G_8"
  \set Staff.shortVocalName = "Kas."
  \set Staff.midiInstrument = "voice oohs"
  <>^\markup \smallCaps "Kaspar"
```

```
}

melchior = {
  \clef "bass"
  \set Staff.shortVocalName = "Mel."
  \set Staff.midiInstrument = "choir aahs"
  <>^\markup \smallCaps "Melchior"
}

\relative c' {
  \kaspar
  c4 c c c
  \melchior
  a4 a a a
  \kaspar
  c4 c c c
}
```




## Vedi anche

Manuale di apprendimento: Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.8 [Testo], pagina 238, Sezione A.11 [Comandi per markup], pagina 707.

## Suggerimenti musicali

I suggerimenti musicali possono essere inseriti nelle pariture vocali, nei libri vocali e nelle parti orchestrali per indicare quale musica in un’altra parte precede immediatamente un’entrata. I suggerimenti sono spesso inseriti anche nella riduzione per pianoforte nelle partiture vocali per indicare cosa sta suonando ogni strumento dell’orchestra. Ciò aiuta il direttore quando non è disponibile la partitura completa.

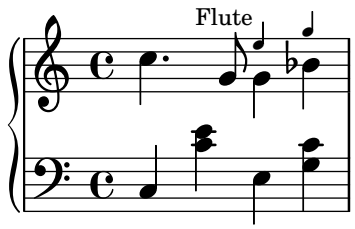
Il meccanismo di base per inserire i suggerimenti è spiegato dettagliatamente in [Citare altre voci], pagina 213, e [Formattazione delle notine], pagina 217. Ma quando si devono inserire molti suggerimenti, per esempio per aiutare il direttore in una partitura vocale, il nome dello strumento deve essere posizionato attentamente proprio prima dell’inizio delle citazioni in corpo più piccolo (“cue notes”). L’esempio seguente mostra come si fa.

```
flute = \relative {
  s4 s4 e'' g
}
\addQuote "flute" { \flute }

pianoRH = \relative {
  c''4. g8
  % posiziona il nome dello strumento citato proprio prima delle citazioni in corpo pi
  % e sopra il rigo
  <>^\markup { \right-align { \tiny "Flute" } }
  \cueDuring "flute" #UP { g4 bes4 }
```

```
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
      \pianoRH
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
  >>
}
```



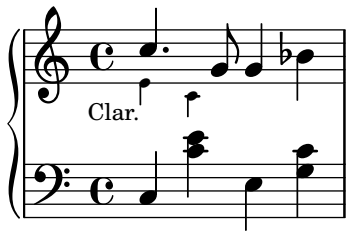

Se viene citato uno strumento traspositore, la parte strumentale deve specificare la sua armatura di chiave in modo che la conversione delle sue notine sia fatta automaticamente. Il prossimo esempio mostra questa trasposizioone per un clarinetto in Si bemolle. Le note in questo esempio si trovano in basso nel rigo, quindi viene specificato `DOWN` in `\cueDuring` (in modo che i gambi vadano giù) e il nome dello strumento è posizionato sotto il rigo.

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % posiziona il nome dello strumento citato sotto il rigo
  <>_\markup { \right-align { \tiny "Clar." } }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {
  \new Voice {
    \pianoRH
  }
      }
      \new Staff {
```

```
  \clef "bass"
  \pianoLH
      }
    >>
  >>
}
```




Da questi due esempi è evidente che inserire molte citazioni in corpo piccolo in una partitura vocale sarebbe noioso, e le note della parte per pianoforte sarebbero confuse. Tuttavia, come mostra il frammento seguente, è possibile definire una funzione musicale per ridurre l'input e rendere più chiare le note per pianoforte.

## Frammenti di codice selezionati

*Aggiungere citazioni orchestrali a una partitura vocale*

L'esempio seguente mostra un approccio per simplificare l'aggiunta di citazioni orchestrali a una riduzione per pianoforte di una partitura vocale. La funzione musicale `\cueWhile` prende quattro argomenti: la musica da cui prendere la citazione, come è definita da `\addQuote`, il nome da inserire prima delle notine, poi o `#UP` o `#DOWN` per specificare o `\voiceOne` col nome sopra il rigo o `\voiceTwo` col nome sotto il rigo, e infine la musica per pianoforte che deve apparire in parallelo alle notine. Il nome dello strumento citato è posto a sinistra delle notine. Molti passaggi possono essere citati, ma non possono sovrapporsi l'un l'altro nel tempo.

```
cueWhile =
#(define-music-function
   (instrument name dir music)
   (string? string? ly:dir? ly:music?)
   #{
     \cueDuring $instrument #dir {
       \once \override TextScript.self-alignment-X = #RIGHT
       \once \override TextScript.direction = $dir
       <>-\markup { \tiny #name }
       $music
     }
   #})

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }
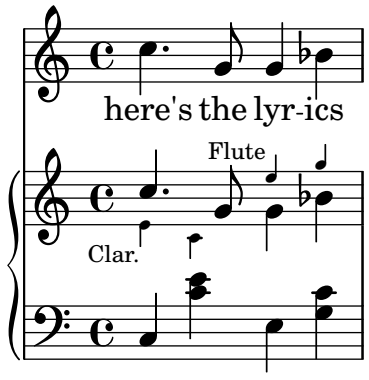
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
```

```
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```




## Vedi anche

Glossario musicale: Sezione “Notine o Citazioni in corpo più piccolo” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.5.1 [Allineamento degli oggetti], pagina 646, Sezione 5.4.2 [Direzione e posizionamento], pagina 630, [Formattazione delle notine], pagina 217, [Citare altre voci], pagina 213, Sezione 5.6 [Uso delle funzioni musicali], pagina 659.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “CueVoice” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

`\cueDuring` inserisce automaticamente un contesto `CueVoice` e tutte le notine sono poste in quel contesto. Ciò significa che non è possibile avere due sequenze sovrapposte di notine con questa tecnica. Si possono inserire sequenze sovrapposte solo dichiarando esplicitamente contesti `CueVoice` distinti e usando `\quoteDuring` per estrarre e inserire le notine.

## Musica parlata
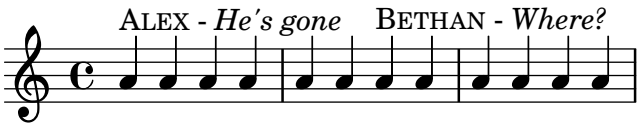
Effetti come il ‘parlato’ o ‘Sprechgesang’ chiedono all’esecutore di parlare senza intonare note ma andando comunque a ritmo; tali effetti si indicano con teste di nota barrate, come è illustrato in [Teste di nota speciali], pagina 39.

## Dialogo sopra la musica

Il dialogo parallelo alla musica appare solitamente sopra i righi in corsivo, con l’inizio di ogni frase collegato a un momento musicale ben preciso.

In caso di brevi intromissioni può bastare un semplice `\markup`.

```
\relative {
  a'4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
  a4 a a^\markup { \smallCaps { Bethan - } \italic Where? } a
  a4 a a a
}
```




In caso di frasi più lunghe può essere necessario espandere la musica per poter far entrare le parole. Non c’è modo di fare ciò del tutto automaticamente in LilyPond e sarà necessario anche qualche intervento manuale per formattare la pagina.

In caso di frasi lunghe o di passaggi con molti dialoghi serrati, l’uso di un contesto Lyrics darà risultati migliori. Il contesto Lyrics non deve essere associato a una voce musicale; occorre invece assegnare a ogni parte del dialogo una durata esplicita. Se c’è un vuoto nel dialogo, la parola finale deve essere separata dal resto e la durata divisa tra le due così che la musica sottostante abbia spazio sufficiente.

Se il dialogo si estende per più di un rigo sarà necessario inserire manualmente dei `\break` e aggiustare il posizionamento del dialogo per evitare di entrare nel margine destro. La parola finale dell’ultima misura di un rigo deve anche essere separata, come mostrato prima.

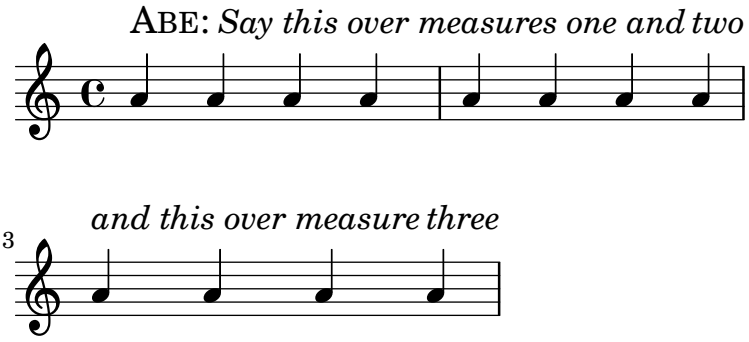
Ecco un esempio che illustra come fare.

```
music = \relative {
  \repeat unfold 3 { a'4 a a a }
}

dialogue = \lyricmode {
  \markup {
```

```
    \fontsize #1 \upright \smallCaps Abe:
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText.font-shape = #'italic
      \override LyricText.self-alignment-X = #LEFT
    }
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```




## Vedi anche

Guida alla notazione: [Durate manuali delle sillabe], pagina 274, Sezione 1.8 [Testo], pagina 238.

Internal Reference: Sezione “LyricText” in *Guida al Funzionamento Interno*.

### 2.1.7 Canti salmi e inni

La musica e le parole per canti, sali e inni di solito segue un formato ben definito in ciascuna chiesa. Sebbene i formati possano differire da chiesa a chiesa, i problemi tipografici che si possono incontrare sono generalmente simili e sono trattati in questa sezione.

## Riferimenti per canti e salmi

La composizione tipografica dei canti gregoriani in vari stili di notazione antica è descritta in Sezione 2.9 [Notazione antica], pagina 440.

## Vedi anche

Guida alla notazione: Sezione 2.9 [Notazione antica], pagina 440.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

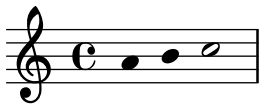
## Impostare un canto

Le impostazioni per il canto moderno usano la notazione moderna insieme ad alcuni elementi della notazione antica. Alcuni di questi elementi e metodi sono mostrati in questa sezione.

I canti usano spesso note di un quarto senza gambi per indicare l'altezza, mentre le durate vengono dal ritmo parlato delle parole.
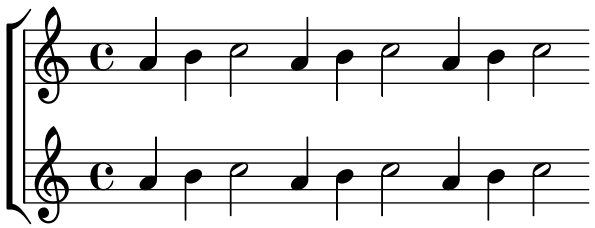
```
stemOff = { \hide Staff.Stem }

\relative c' {
  \stemOff
  a'4 b c2 |
}
```
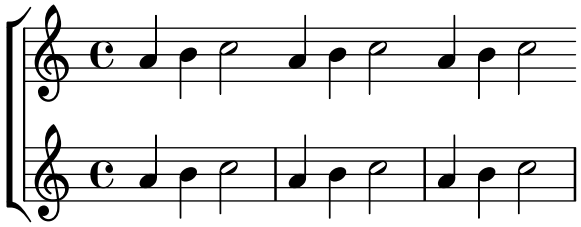


Nei canti le stanghette sono spesso omesse oppure si usano delle stanghette più brevi o punteggiate per indicare le pause nella musica. Per omettere tutte le stanghette da tutti i righi si disattiva l'incisore delle stanghette:

```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative {
  a'4 b c2 |
  a4 b c2 |
  a4 b c2 |
      }
    }
    \new Staff {
      \relative {
  a'4 b c2 |
  a4 b c2 |
  a4 b c2 |
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```
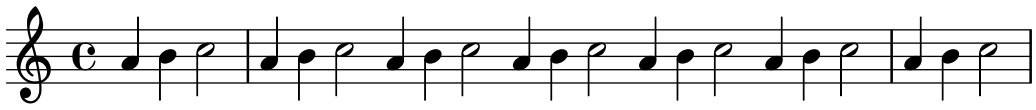


Le stanghette possono anche essere tolte solo in certi righi:

```
\score {
  \new ChoirStaff <<
    \new Staff
    \with { \remove "Bar_engraver" } {
      \relative {
  a'4 b c2 |
  a4 b c2 |
  a4 b c2 |
      }
    }
    \new Staff {
      \relative {
  a'4 b c2 |
  a4 b c2 |
  a4 b c2 |
      }
    }
  >>
}
```



Per togliere le stanghette da una sezione musicale soltanto basta trattarla come una cadenza. Se la sezione è lunga potrebbe essere necessario inserire delle stanghette fittizie con `\bar ""` per mostrare dove si deve interrompere la linea.
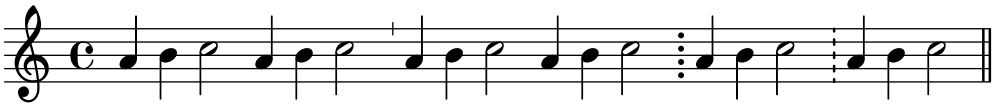
```
\relative a' {
  a4 b c2 |
  \cadenzaOn
  a4 b c2
  a4 b c2
  \bar ""
  a4 b c2
  a4 b c2
  \cadenzaOff
  a4 b c2 |
  a4 b c2 |
}
```



Nei canti le pause si indicano con stanghette modificate.

```
\relative a' {
  a4
```

```
  \cadenzaOn
  b c2
  a4 b c2
  \bar "'"
  a4 b c2
  a4 b c2
  \bar ";"
  a4 b c2
  \bar "!"
  a4 b c2
  \bar "||"
}
```
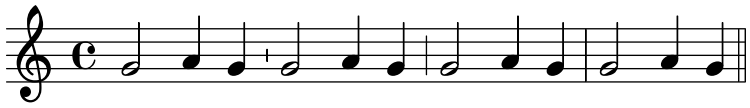


Altrimenti, talvolta si usa la notazione usata nel canto gregoriano per le pause anche se il resto della notazione è moderna. Nell'esempio seguente si usa un segno `\breathe` modificato:

```
divisioMinima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-minima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
divisioMaior = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maior
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
divisioMaxima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maxima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \breathe
}
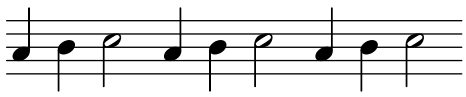
\score {
  \relative {
    g'2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
    g2 a4 g
    \finalis
  }
  \layout {
```

```
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}
```



Nei canti viene solitamente omessa l'indicazione di tempo e spesso anche la chiave.

```
\score {
  \new Staff {
    \relative {
      a'4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
    }
  }
}
```



I canti per salmi della tradizione anglicana sono solitamente o *singoli*, con 7 battute musicali, oppure *doppi*, con due gruppi di 7 battute. Ogni gruppo di 7 battute è diviso a metà, che corrispondono alle metà di ciascun verso, di solito separato da una doppia stanghetta. Si usano solo semibrevi e minime. La prima battuta di ogni metà contiene sempre un solo accordo di semibrevi, che viene chiamato la "nota recitativa". I canti sono centrati sulla pagina.

```
SopranoMusic = \relative {
  g'1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative {
  e'1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

TenorMusic = \relative {
  c'1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
```

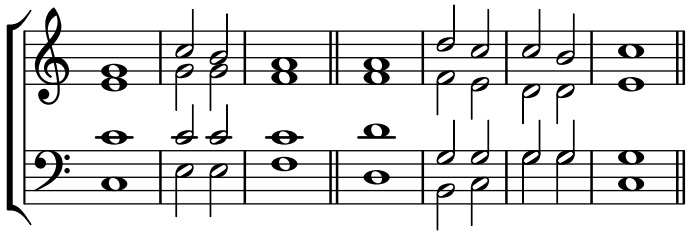
```
}

BassMusic = \relative {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

% Si usa un blocco markup per centrare il canto sulla pagina
\markup {
  \fill-line {
    \score { % centered
      <<
  \new ChoirStaff <<
    \new Staff <<
      \global
      \clef "treble"
      \new Voice = "Soprano" <<
        \voiceOne
        \SopranoMusic
      >>
      \new Voice = "Alto" <<
        \voiceTwo
        \AltoMusic
      >>
    >>
    \new Staff <<
      \clef "bass"
      \global
      \new Voice = "Tenor" <<
        \voiceOne
        \TenorMusic
      >>
      \new Voice = "Bass" <<
        \voiceTwo
        \BassMusic
      >>
    >>
  >>
      >>
      \layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration = #(ly:make-moment
    1/2)
  }
  \context {
    \Staff
    \remove "Time_signature_engraver"
```

```
  }
      }
    } % End score
  }
} % End markup
```



Altri approcci per impostare un canto simile sono illustrati nel primo dei seguenti frammenti.

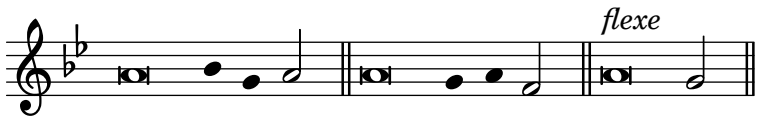
## Frammenti di codice selezionati

*Notazione per canti e salmi*

Questa forma di notazione è utilizzata per i salmi, dove i versi non sono sempre della stessa lunghezza.

```
stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \bar "||"
  }
}
```




I cantici e altri testi liturgici possono essere impostati in modo più libero e possono usare elementi della notazione della musica antica. Le parole sono spesso mostrate sotto e allineate con le note. In questo caso le note sono spaziate in base alle sillabe invece che alle durate delle note.

*Modello per notazione antica – trascrizione moderna di musica gregoriana*
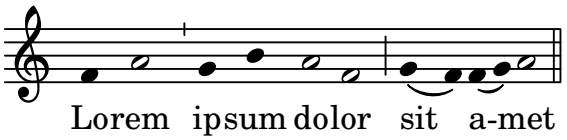
Questo esempio mostra come realizzare una trascrizione moderna di musica gregoriana. La musica gregoriana non presenta la suddivisione in misure né gambi; impiega soltanto le teste della minima e della semiminima, e dei segni appositi che indicano pause di diversa lunghezza.

```
\include "gregorian.ly"
```

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```




## Vedi anche

Manuale di apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*, Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.9 [Notazione antica], pagina 440, [Stanghette], pagina 101, Sezione 5.1.4 [Modifica dei componenti aggiuntivi di un contesto], pagina 600, Sezione 2.9.4 [Scrivere il canto gregoriano], pagina 452, [Musica in tempo libero], pagina 76, Sezione 5.4.7 [Visibilità degli oggetti], pagina 638.

## Salmi

Le parole di un salmo anglicano sono solitamente stampate in versi separati centrati sotto il canto.

I canti singoli (7 battute) si ripetono per ogni verso. I canti doppi (con 14 battute) si ripetono per ogni coppia di versi. Sono inseriti dei segni nelle parole per mostrare come si combinano col canto. Ogni verso è diviso a metà e tale divisione è indicata solitamente dai due punti, che corrispondono alla doppia stanghetta in musica. Le parole che precedono i due punti sono cantate insieme alle prime tre battute della musica; quelle successive insieme alle restanti quattro battute.

Stanghette singole (o in alcuni libri di salmi una virgola inversa o segno simile) sono inserite tra le parole per indicare dove cadono le stanghette nella musica. In modalità markup una stanghetta singola può essere inserita usando il simbolo di controllo battuta `|`.

```
\markup {
  \fill-line {
    \column {
      \left-align {
  \line { O come let us sing | unto the | Lord : let }
  \line { us heartily rejoice in the | strength of | our }
  \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Altri simboli potrebbero richiedere i glifi dei tipi di carattere `fetaMusic`. Maggiori informazioni in Sezione 1.8.3 [Tipi di carattere], pagina 260.

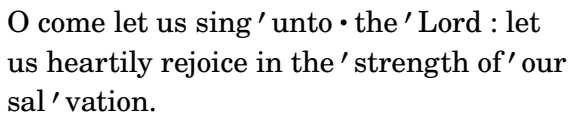
```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
  \line { O come let us sing \tick unto the \tick Lord : let }
  \line {
    us heartily rejoice in the \tick strength of \tick our
  }
  \line { sal \tick vation. }
      }
    }
  }
}
```

O come let us sing ' unto the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Se c'è una semibreve in una battuta, tutte le parole che si riferiscono a quella battuta sono recitate su quella nota singola col ritmo del parlato. Dove ci sono due note in una battuta ci

saranno solo una o due sillabe corrispondenti. Se ci sono più di due sillabe, si inserisce solitamente un punto per indicare dove si trova il cambio di nota.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
  \line {
    O come let us sing \tick unto \dot the \tick Lord : let
  }
  \line {
    us heartily rejoice in the \tick strength of \tick our
  }
  \line { sal \tick vation. }
      }
    }
  }
}
```




In alcuni libri di salmi si usa un asterisco, al posto di una virgola, per indicare una pausa in una sezione recitata, mentre le sillabe accentate o leggermente allungate sono indicate in grassetto.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
  \line { Today if ye will hear his voice * }
  \line {
    \concat { \bold hard en }
    | not your | hearts : as in the pro-
  }
  \line { vocation * and as in the \bold day of tempt- | }
  \line { -ation | in the | wilderness. }
      }
    }
  }
```

```
}
```

Today if ye will hear his voice *
**hard**en | not your | hearts : as in the pro-
vocation * and as in the **day** of tempt- |
-ation | in the | wilderness.

In altri libri di salmi si usa un simbolo di accento sopra la sillaba per indicare l'accento.

```
tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
  \line {
    O come let us \concat {
      si \combine \tick ng
    }
    | unto the | Lord : let
  }
  \line {
    us heartily \concat {
      rejo \combine \tick ice
    }
    in the | strength of | our
  }
  \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing′ | unto the | Lord : let
us heartily rejoice′ in the | strength of | our
sal- | -vation.

L'uso di `\markup` per centrare il testo e disporre le linee in colonne è descritto in Sezione 1.8.2 [Formattazione del testo], pagina 246.

La maggior parte di questi elementi sono mostrati in uno dei due versi del modello Sezione "Salmi" in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione "Salmi" in *Manuale di Apprendimento*, Sezione "Modelli per gruppi vocali" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.8.3 [Tipi di carattere], pagina 260, Sezione 1.8.2 [Formattazione del testo], pagina 246.

## Misure parziali nei motivi degli inni

Le melodie degli inni spesso iniziano e terminano ogni rigo musicale con misure parziali, così che ciascun rigo musicale corrisponda esattamente a un rigo di testo. Per fare ciò è necessario un comando `\partial` all'inizio della musica e dei comandi `\bar "|"` o `\bar "||"` alla fine di ogni linea.

*Modello per inno*

Il codice seguente presenta un modo di impostare un inno in cui ogni verso inizia e finisce con una misura parziale. Mostra anche come aggiungere delle strofe come testo separato sotto la musica.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic =  \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

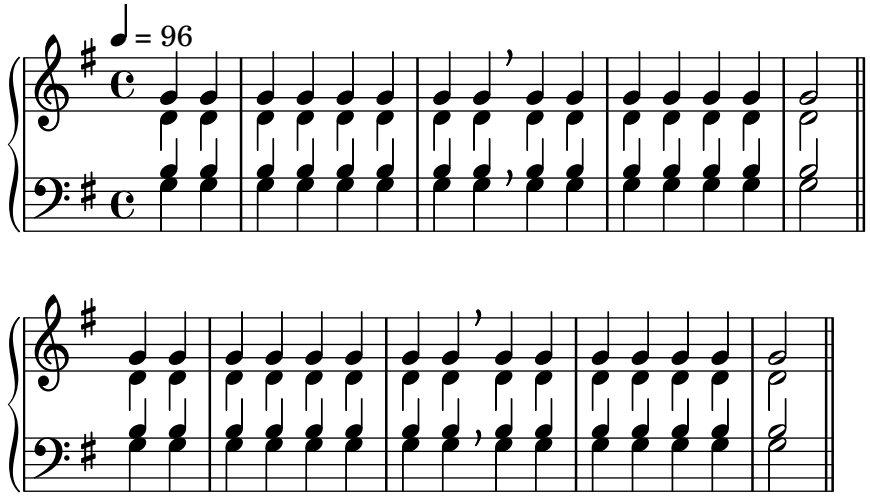
global = {
 \key g \major
}

\score {  % Start score
  <<
    \new PianoStaff <<  % Start pianostaff
      \new Staff <<  % Start Staff = RH
        \global
        \clef "treble"
        \new Voice = "Soprano" <<  % Start Voice = "Soprano"
          \Timeline
          \voiceOne
          \SopranoMusic
```

```
        >>  % End Voice = "Soprano"
        \new Voice = "Alto" <<  % Start Voice = "Alto"
          \Timeline
          \voiceTwo
          \AltoMusic
        >>  % End Voice = "Alto"
      >>  % End Staff = RH
      \new Staff <<  % Start Staff = LH
        \global
        \clef "bass"
        \new Voice = "Tenor" <<  % Start Voice = "Tenor"
          \Timeline
          \voiceOne
          \TenorMusic
        >>  % End Voice = "Tenor"
        \new Voice = "Bass" <<  % Start Voice = "Bass"
          \Timeline
          \voiceTwo
          \BassMusic
        >>  % End Voice = "Bass"
      >>  % End Staff = LH
    >>  % End pianostaff
  >>
}  % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
    ""
  }
}

\paper {  % Start paper block
  indent = 0     % don't indent first system
  line-width = 130   % shorten line length to suit music
}  % End paper block
```




This is line one of the first verse
This is line two of the same
And here's line three of the first verse
And the last line of the same

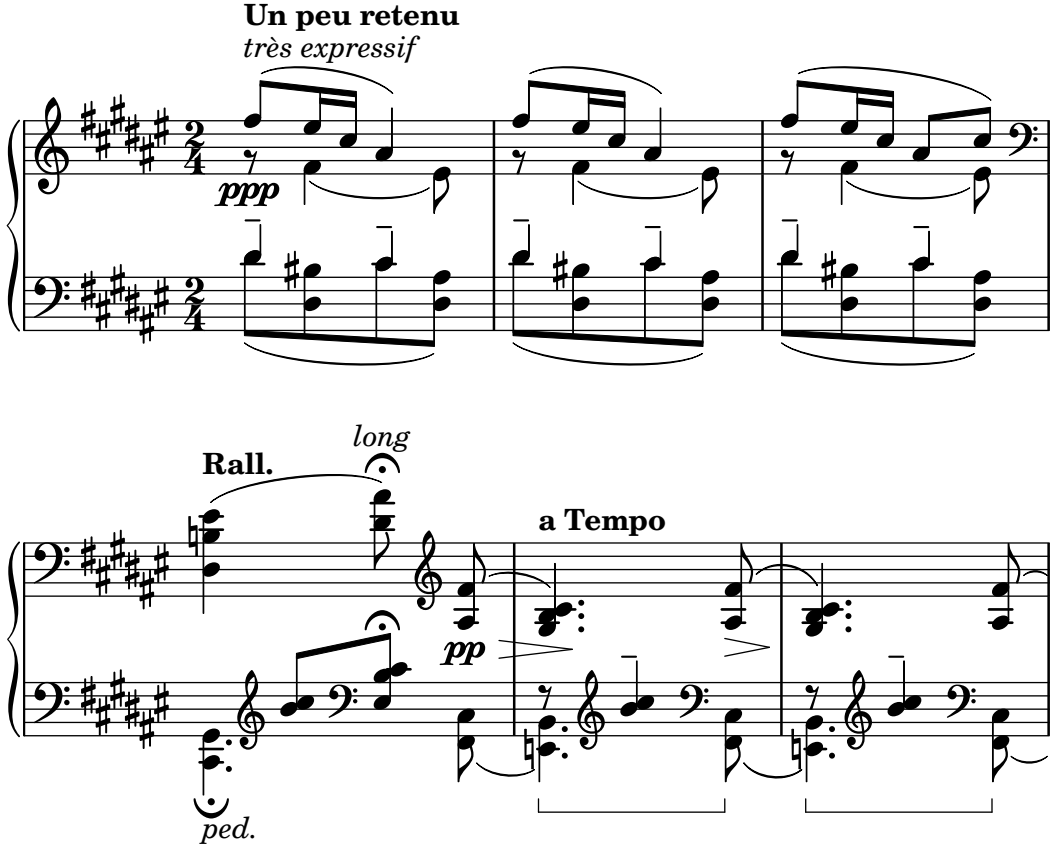
### 2.1.8 Musica vocale antica

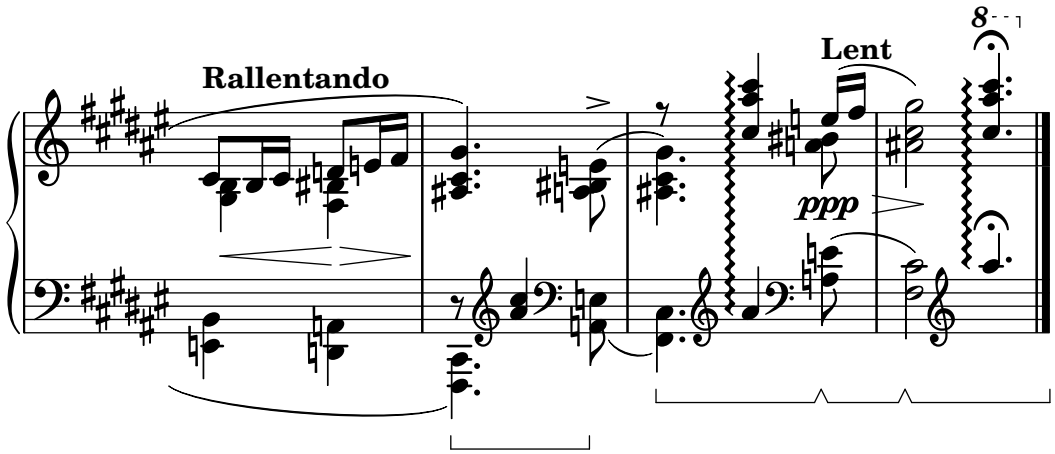
È supportata la musica vocale antica, come spiegato in Sezione 2.9 [Notazione antica], pagina 440.

### Vedi anche

Guida alla notazione: Sezione 2.9 [Notazione antica], pagina 440.

## 2.2 Tastiera e altri strumenti multirigo






Questa sezione tratta vari aspetti della notazione musicale che sono specifici degli strumenti a tastiera e di altri strumenti che vengono scritti su molti righi, come arpe e vibrafoni. Ai fini di questa sezione, questo gruppo di strumenti multirigo viene chiamato "tastiere" per brevità, anche se alcuni di questi strumenti non hanno una tastiera.

### 2.2.1 Notazione comune per tastiere

Questa sezione tratta le questioni di notazione che riguardano la maggior parte degli strumenti a tastiera.

#### Riferimenti per tastiere

Gli strumenti a tastiera vengono rappresentati solitamente con righi per pianoforte, ovvero due o più righi normali raggruppati da una parentesi graffa. La stessa notazione viene usata anche per altri strumenti dotati di tasti. La musica per organo viene scritta normalmente con due righi all'interno di un gruppo `PianoStaff` più un terzo rigo normale per i pedali.

I righi nella musica per tastiera sono molto indipendenti, ma talvolta le voci possono attraversare i due righi. Questa sezione tratta tecniche di notazione specifiche della musica per tastiera.

Varie questioni comuni della musica per tastiera sono discusse altrove:

- La musica per tastiera contiene solitamente varie voci, il cui numero può cambiare regolarmente, come è descritto in [Risoluzione delle collisioni], pagina 179.
- La musica per tastiera può essere scritta in parallelo, come è descritto in [Scrivere la musica in parallelo], pagina 191.
- Le dinamiche possono essere poste in un contesto `Dynamics`, tra i due contesti `Staff`, per allineare le indicazioni dinamiche su una linea orizzontale centrata tra i righi; vedi [Dinamiche], pagina 126.
- Le diteggiature sono indicate con [Indicazioni di diteggiatura], pagina 226.
- Le indicazioni per i pedali dell'organo sono inserite come articolazioni, vedi Sezione A.14 [Elenco delle articolazioni], pagina 764.
- Le linee della griglia verticali possono essere mostrate con le [Linee della griglia], pagina 234.
- La musica per tastiera contiene spesso legature di valore *Laissez vibrer* così come legature di valore su arpeggi e tremoli, descritti in [Legature di valore], pagina 55.
- Il posizionamento di arpeggi attraverso molteplici voci e righi è trattato in [Arpeggio], pagina 147.
- I segni di tremolo sono descritti in [Ripetizioni con tremolo], pagina 167.
- Varie modifiche manuali che possono rendersi necessarie nella musica per tastiera sono illustrati in Sezione "Esempio musicale" in *Manuale di Apprendimento*.
- Si possono usare le note nascoste per produrre delle legature di valore che attraversano le voci, come mostrato in Sezione "Altri usi delle modifiche con \tweak" in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione “Esempio musicale” in *Manuale di Apprendimento*, Sezione “Altri usi delle modifiche con \tweak” in *Manuale di Apprendimento*.

Guida alla notazione: [Raggruppare i righi], pagina 195, [Nomi degli strumenti], pagina 210, [Risoluzione delle collisioni], pagina 179, [Scrivere la musica in parallelo], pagina 191, [Indicazioni di diteggiatura], pagina 226, Sezione A.14 [Elenco delle articolazioni], pagina 764, [Linee della griglia], pagina 234, [Legature di valore], pagina 55, [Arpeggio], pagina 147, [Ripetizioni con tremolo], pagina 167.

Guida al funzionamento interno: Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

Frammenti: Sezione “Keyboards” in *Frammenti di codice*.

## Modifica manuale del rigo

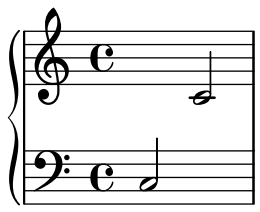
Le voci possono essere passate a un altro rigo manualmente, usando il comando

```
\change Staff = nomerigo
```

La stringa *nomerigo* è il nome del rigo. Fa passare la voce corrente dal suo rigo corrente al rigo chiamato *nomerigo*. Valori tipici di *nomerigo* sono `"su"` e `"giù"` o `"MD"` (mano destra) e `"MS"` (mano sinistra).

Il rigo a cui viene passata la voce deve esistere nel momento dello scambio. Se necessario, i righi devono essere “conservati”, vedi Sezione 5.1.3 [Conservazione di un contesto], pagina 598, o istanziati esplicitamente, per esempio usando l’accordo vuoto, `<>`, vedi [Note in un accordo], pagina 170.
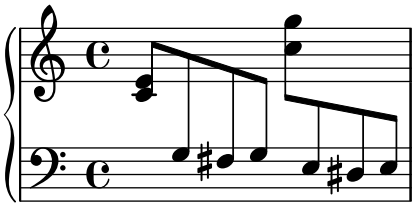
```
\new PianoStaff <<
  \new Staff = "up" {
    % forza la creazione di tutti i contesti in questo momento
    <>
    \change Staff = "down" c2
    \change Staff = "up" c'2
  }
  \new Staff = "down" {
    \clef bass
    % conserva il rigo
    s1
  }
>>
```



Le note trasversali al rigo hanno la travatura automatica:

```
\new PianoStaff <<
  \new Staff = "su" {
    <e' c'>8
    \change Staff = "giù"
    g8 fis g
    \change Staff = "su"
    <g'' c''>8
```

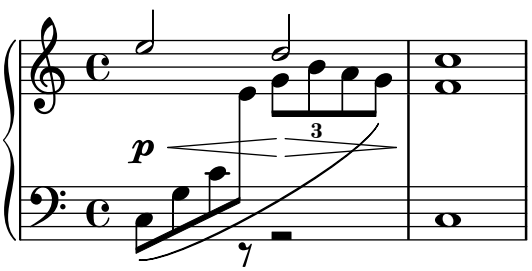
```
    \change Staff = "giù"
    e8 dis e
    \change Staff = "su"
  }
  \new Staff = "giù" {
    \clef bass
    % conserva il contesto
    s1
  }
>>
```



Se la disposizione delle travature deve essere modificata, modificare prima le direzioni dei gambi. Le posizioni delle travature vengono quindi misurate dal centro del rigo che è più vicino alla travatura. Un semplice esempio di modifica manuale della travatura è descritto in Sezione "Correggere elementi della notazione sovrapposti" in *Manuale di Apprendimento*.

Alcuni elementi della notazione potranno sovrapporsi quando le voci sono trasversali ai righi:

```
\new PianoStaff <<
  \new Staff = "su" {
    \voiceOne
    % Fai spazio alla diteggiatura nella voce trasversale ai righi
    \once\override DynamicLineSpanner.staff-padding = #4
    e''2\p\< d''\>
    c''1\!
  }
  \new Staff = "giù" <<
    {
      \clef bass
      s4. e,8\rest g,2\rest
      c1
    } \\ {
      c8\( g c'
      \change Staff = "su"
      e' g' b'-3 a' g'\)
      f'1
    }
  >>
>>
```



Il gambo e la legatura di portamento si sovrappongono alla linea intermedia della dinamica, perché la risoluzione automatica delle collisioni è sospesa per travature, legature di portamento e altri estensori che collegano note di righi diversi, così come per gambi e articolazioni se il loro posizionamento è influenzato da un estensore trasversale ai righi. Le collisioni risultanti devono essere risolte manualmente, laddove necessario, usando i metodi descritti in Sezione "Correggere elementi della notazione sovrapposti" in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione "Correggere elementi della notazione sovrapposti" in *Manuale di Apprendimento*.

Guida alla notazione: [Gambi], pagina 232, [Travature automatiche], pagina 85, Sezione 5.1.3 [Conservazione di un contesto], pagina 598.

Frammenti: Sezione "Keyboards" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Beam" in *Guida al Funzionamento Interno*, Sezione "ContextChange" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non è possibile evitare la collisione delle travature per le travature automatiche che terminano appena prima di un cambio di rigo. In questo caso occorre usare le travature manuali.

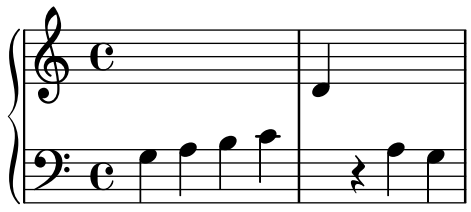
## Modifica automatica del rigo

È possibile scambiare automaticamente le voci tra rigo superiore e inferiore. La sintassi è la seguente:

```
\autoChange ...musica...
```

Ciò creerà due righi dentro il gruppo di righi corrente (solitamente un `PianoStaff`), chiamati `"up"` (su) e `"down"` (giù). Il rigo più basso sarà in chiave di basso. La funzione di scambio automatico entra in azione in base all'altezza (il Do centrale è il punto di svolta), e guarda in avanti saltando le pause per poter scambiare in anticipo.

```
\new PianoStaff {
  \autoChange {
    g4 a b c'
    d'4 r a g
  }
}
```
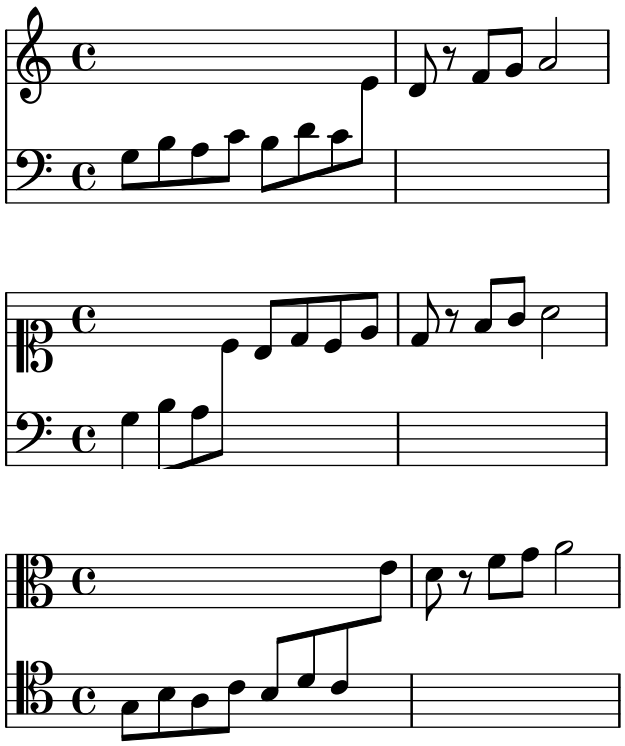


Si possono specificare altre altezze per il punto di svolta. Se i righi non sono istanziati esplicitamente, si possono usare altre chiavi.

```
music = {
  g8 b a c' b8 d' c'8 e'
  d'8 r f' g' a'2
}

\autoChange d' \music
```

```
\autoChange b \with { \clef soprano } \music
\autoChange d' \with { \clef alto } \with { \clef tenor } \music
```



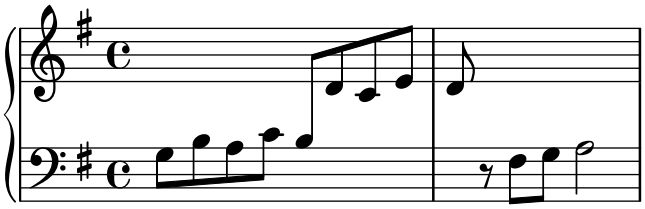
Una sezione `\relative` che si trova fuori da `\autoChange` non ha effetto sulle altezze della musica. Dunque, se necessario, mettere `\relative` dentro `\autoChange`.

Se è richiesto ulteriore controllo sui righi individuali, possono essere creati manualmente coi nomi `"up"` e `"down"`. Il comando `\autoChange` scambierà allora la sua voce tra i righi esistenti.

> **Nota:** Se i righi sono creati manualmente, *devono* chiamarsi `"up"` e `"down"`.

Per esempio, i righi devono essere creati manualmente per posizionare un'armatura di chiave nel rigo inferiore:

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melOne" {
      \key g \major
      \autoChange \relative {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



## Vedi anche

Guida alla notazione: [Modifica manuale del rigo], pagina 333.

Frammenti: Sezione “Keyboards” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “AutoChangeMusic” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

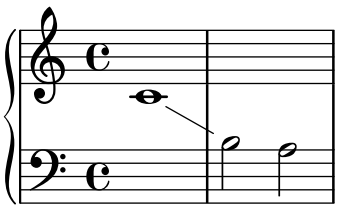
Gli scambi di rigo potrebbero finire in posizioni non ottimali. Per ottenere un risultato di alta qualità, i cambi di rigo devono essere specificati manualmente.

Gli accordi non vengono divisi tra i righi; vengono assegnati a un rigo in base alla prima nota nominata nel costrutto dell’accordo.

## Linee del cambio rigo

Quando una voce passa a un altro rigo, è possibile stampare automaticamente una linea connettiva:

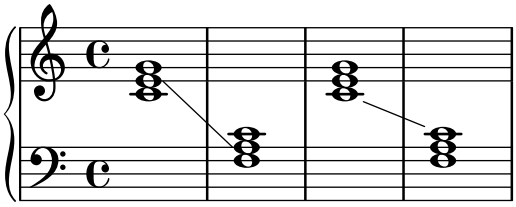
```
\new PianoStaff <<
  \new Staff = "uno" {
    \showStaffSwitch
    c'1
    \change Staff = "due"
    b2 a
  }
  \new Staff = "due" {
    \clef bass
    s1*2
  }
>>
```



Una linea di cambio rigo tra accordi collega le “ultime note” degli accordi come sono scritte nel file sorgente; ciò può essere utile per regolare velocemente le posizioni verticali di inizio e fine della linea.

```
\new PianoStaff <<
  \new Staff = "uno" {
    <c' e' g'>1
    \showStaffSwitch
    \change Staff = "due"
    <a c' f>1
    \hideStaffSwitch
    \change Staff = "uno"
```

```
    <e' g' c'>1
    \showStaffSwitch
    \change Staff = "due"
    <f a c'>1
  }
  \new Staff = "due" {
    \clef bass
    s1*4
  }
>>
```



## Comandi predefiniti

`\showStaffSwitch`, `\hideStaffSwitch`.

## Vedi anche

Frammenti: Sezione “Keyboards” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Note_head_line_engraver” in *Guida al Funzionamento Interno*, Sezione “VoiceFollower” in *Guida al Funzionamento Interno*.

## Frammenti di codice selezionati

*Cross staff stems*
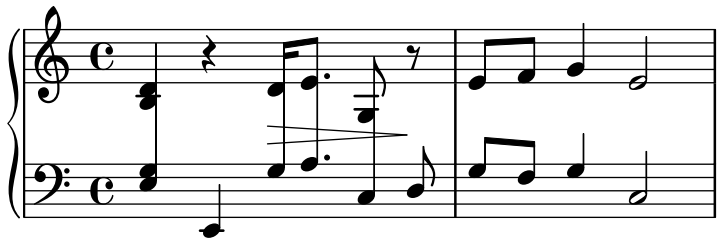
This snippet shows the use of the `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem length need not be specified, as the variable distance between noteheads and staves is calculated automatically.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
    \new Staff {
      <b d'>4 r d'16\> e'8. g8 r\!
      e'8 f' g'4 e'2
    }
    \new Staff {
      \clef bass
      \voiceOne
      \autoBeamOff
      \crossStaff { <e g>4 e, g16 a8. c8} d
```

```
      \autoBeamOn
      g8 f g4 c2
    }
  >>
}
```



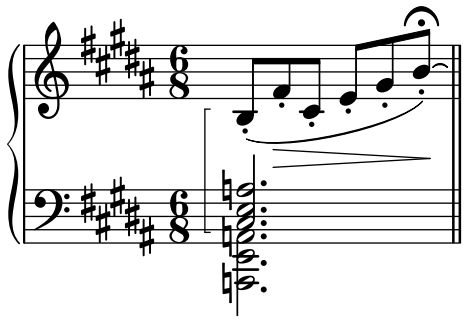
Per il momento, questo incisore non può essere specificato scrivendo il suo nome tra virgolette doppie. A causa del modo in cui è implementato, il suo nome deve essere preceduto dal simbolo di cancelletto `#`.

*Indicare accordi trasversali al rigo con la parentesi quadrata dell'arpeggio*

Una parentesi quadrata dell'arpeggio può indicare che delle note su due righi diversi devono essere suonate con la stessa mano. Per farlo, bisogna far sì che `PianoStaff` accetti gli arpeggi trasversali ai righi e gli arpeggi siano impostati nella forma della parentesi quadrata nel contesto `PianoStaff`.

(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.
        }
      >>
    }
  }
>>
```



### Vedi anche

Frammenti: Sezione “Keyboards” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Stem” in *Guida al Funzionamento Interno*.
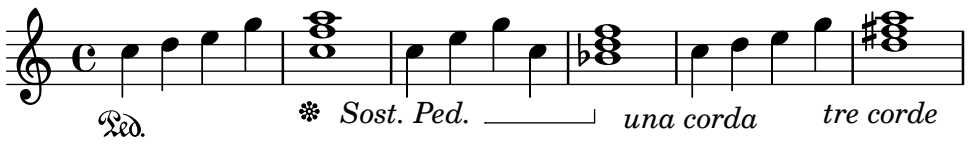
## 2.2.2 Pianoforte

Questa sezione tratta le questioni di notazione che hanno a che fare più direttamente col pianoforte.

### Pedali del pianoforte

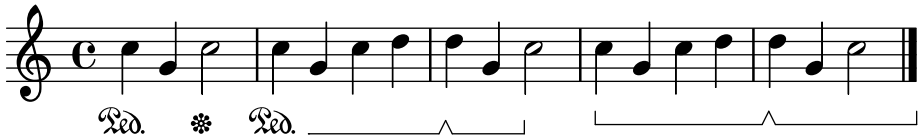
I pianoforti hanno generalmente tre pedali che alterano il modo in cui il suono viene prodotto: *risonanza*, *tonale* o *sostenuto* (*sos.*) e *una corda* (*U.C.*). I pedali di risonanza si trovano anche nei vibrafoni e nelle celeste.

```
\relative {
  c''4\sustainOn d e g
  <c, f a>1\sustainOff
  c4\sostenutoOn e g c,
  <bes d f>1\sostenutoOff
  c4\unaCorda d e g
  <d fis a>1\treCorde
}
```




Ci sono tre stili per indicare i pedali: `text` (testo), `bracket` (parentesi quadra) e `mixed` (misto). Il pedale di risonanza e il pedale una corda usano il testo come stile predefinito, mentre il pedale tonale usa lo stile misto.

```
\relative {
  c''4\sustainOn g c2\sustainOff
  \set Staff.pedalSustainStyle = #'mixed
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2\sustainOff
  \set Staff.pedalSustainStyle = #'bracket
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2
  \bar "|."
}
```



Il posizionamento dei comandi del pedale corrisponde ai movimenti fisici del pedale di risonanza durante l'esecuzione musicale. Per indicare che il pedale è attivo fino alla stanghetta finale si omette il comando finale che disattiva il pedale.

Le indicazioni del pedale possono essere poste in un contesto `Dynamics`, che le allinea su una linea orizzontale.

### Vedi anche

Guida alla notazione: [Legature di valore], pagina 55.

Frammenti: Sezione "Keyboards" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "SustainPedal" in *Guida al Funzionamento Interno*, Sezione "SustainPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "SustainEvent" in *Guida al Funzionamento Interno*, Sezione "SostenutoPedal" in *Guida al Funzionamento Interno*, Sezione "SostenutoPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "SostenutoEvent" in *Guida al Funzionamento Interno*, Sezione "UnaCordaPedal" in *Guida al Funzionamento Interno*, Sezione "UnaCordaPedalLineSpanner" in *Guida al Funzionamento Interno*, Sezione "UnaCordaEvent" in *Guida al Funzionamento Interno*, Sezione "PianoPedalBracket" in *Guida al Funzionamento Interno*, Sezione "Piano_pedal_engraver" in *Guida al Funzionamento Interno*.

## 2.2.3 Fisarmonica

Questa sezione tratta la notazione specifica della fisarmonica.

### Simboli di discanto

Le fisarmoniche sono spesso costruite con più di un insieme di ance che possono essere all'unisono, un'ottava sopra o un'ottava sotto rispetto all'altezza scritta. Ogni costruttore di fisarmoniche ha nomi diversi per i *registri* che selezionano le varie combinazioni di ance, come *oboe*, *musette* o *bandoneon*. È stato quindi ideato un sistema di simboli per semplificare le istruzioni di esecuzione.

Un elenco completo di tutti i registri per fisarmonica disponibili si trova in Sezione A.11.6 [Accordion Registers], pagina 749.

### Frammenti di codice selezionati

*Simboli di registro della fisarmonica*
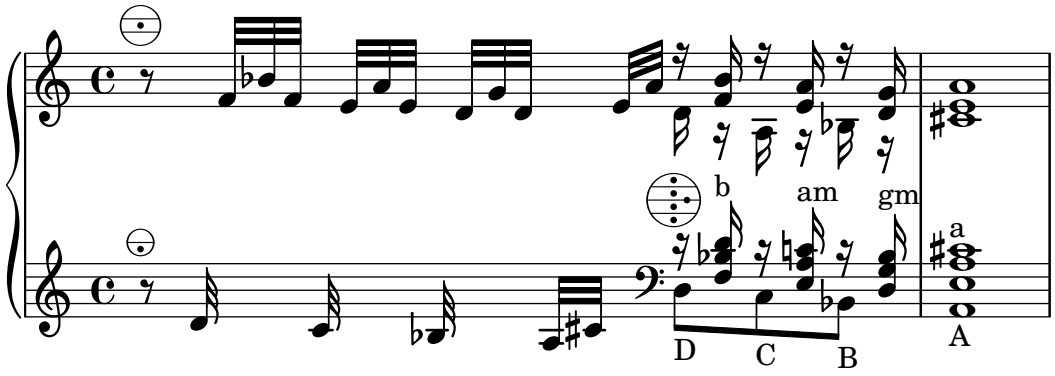
I simboli di registro della fisarmonica sono disponibili sia come `\markup` sia come eventi musicali autonomi (perché i cambi di registro capitano solitamente tra reali eventi musicali). I registri bassi non sono troppo standardizzati. I comandi disponibili si trovano nella sezione «Registri della fisarmonica» della Guida alla notazione.

```
#(use-modules (scm accreg))

\new PianoStaff
<<
  \new Staff \relative {
    \clef treble
    \discant "10"
    r8 s32 f'[ bes f] s e[ a e] s d[ g d] s16 e32[ a]
    <<
      { r16 <f bes> r <e a> r <d g> }
      \\
      { d r a r bes r }
    >> |
```

```
    <cis e a>1
  }

  \new Staff \relative {
    \clef treble
    \freeBass "1"
    r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
    \clef bass \stdBass "Master"
    <<
      { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
      <e a cis>1^"a" }
      \\
      { d8_"D" c_"C" bes_"B" | a1_"A" }
    >>
  }
>>
```




## Vedi anche

Frammenti: Sezione “Keyboards” in *Frammenti di codice*.

### 2.2.4 Arpa

Questa sezione tratta le questioni di notazione specifiche dell’arpa.

## Riferimenti per arpe

Alcune caratteristiche comuni della musica per arpa sono trattate altrove:

- Il glissando è la tecnica più caratteristica dell’arpa, vedi [Glissando], pagina 142.
- Un *bisbigliando* è scritto come un tremolo, vedi [Ripetizioni con tremolo], pagina 167.
- Gli armonici naturali sono trattati in [Armonici], pagina 345.
- Per arpeggi direzionali e non-arpeggi, vedere [Arpeggio], pagina 147.

## Vedi anche
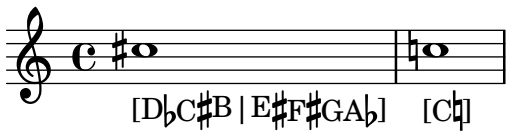
Guida alla notazione: [Ripetizioni con tremolo], pagina 167, [Glissando], pagina 142, [Arpeggio], pagina 147, [Armonici], pagina 345.

## Pedali dell’arpa

Le arpe hanno sette corde per ottava che possono risuonare all’altezza naturale, bemolle o diesis. Nelle arpe con leva, ogni corda viene regolata individualmente, ma nelle arpe a pedale ogni corda con la stessa altezza è regolata da un unico pedale. I pedali, da sinistra a destra

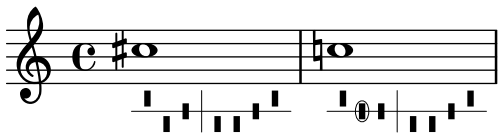
rispetto all'esecutore, sono D, C e B a sinistra e E, F, G e A a destra. La posizione dei pedali può essere indicata con dei segni testuali:

```
\textLengthOn
cis''1_\markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c''!1_\markup \concat \vcenter {
  [ C \natural ] }
```




o con dei diagrammi dei pedali:

```
\textLengthOn
cis''1_\markup { \harp-pedal "^v-|vv-^" }
c''!1_\markup { \harp-pedal "^o--|vv-^" }
```

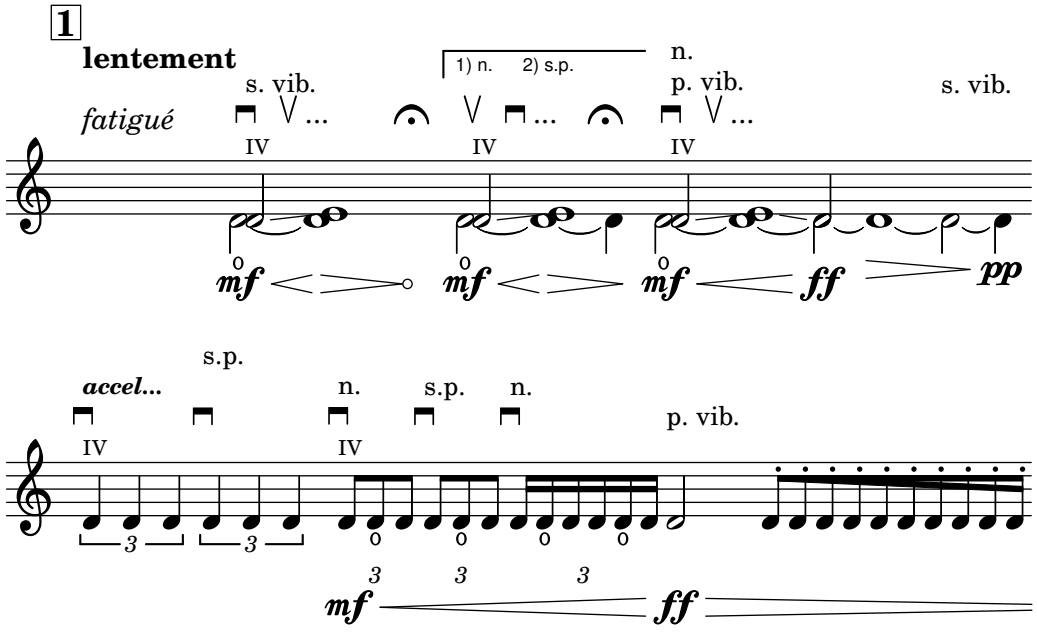


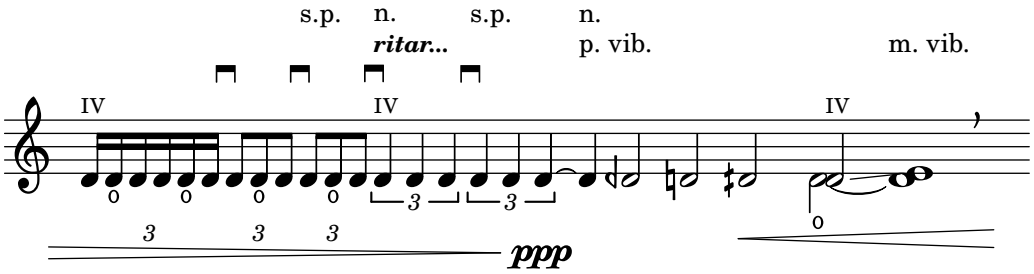
Il comando `\harp-pedal` accetta una stringa di caratteri, dove `^` è la posizione più alta del pedale (altezza bemolle), `-` è la posizione centrale del pedale (altezza naturale), `v` è la posizione più bassa del pedale (altezza diesis) e `|` è il divisore. `o` posto prima di un simbolo produrrà un cerchio intorno al simbolo di pedale.

### Vedi anche

Guida alla notazione: [Scritte], pagina 239, Sezione A.11.5 [Instrument Specific Markup], pagina 746.

## 2.3 Strumenti a corde senza tasti






Questa sezione fornisce informazioni e riferimenti utili per scrivere musica per strumenti a corde senza tasti, principalmente archi da orchestra.

### 2.3.1 Notazione comune per strumenti a corde senza tasti

La notazione specialistica per strumenti a corde senza tasti è esigua. La musica viene scritta su un solo rigo e di solito è richiesta una sola voce. Due voci potrebbero essere necessarie solo per alcuni passaggi a doppia corda o i divisi.

#### Riferimenti per archi senza tasti

La maggior parte della notazione utile per gli archi da orchestra e altri strumenti ad arco è trattata altrove:

- Le indicazioni testuali come "pizz." e "arco" sono inserite come testo semplice, vedi [Scritte], pagina 239.
- Le diteggiature, incluse le indicazioni per il pollice, sono descritte in [Indicazioni di diteggiatura], pagina 226.
- La doppia corda viene normalmente indicata scrivendo un accordo, vedi [Note in un accordo], pagina 170. Si possono inserire delle direttive su come eseguire gli accordi, vedi [Arpeggio], pagina 147.
- I modelli per i quartetti d'archi si trovano in Sezione "Modelli per quartetto d'archi" in *Manuale di Apprendimento*. Altri sono illustrati nei frammenti.

#### Vedi anche

Manuale di apprendimento: Sezione "Modelli per quartetto d'archi" in *Manuale di Apprendimento*.

Guida alla notazione: [Scritte], pagina 239, [Indicazioni di diteggiatura], pagina 226, [Note in un accordo], pagina 170, [Arpeggio], pagina 147.
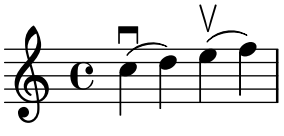
Frammenti: Sezione "Unfretted strings" in *Frammenti di codice*.

#### Indicazioni di arcata

Le indicazioni di arcata vengono create come articolazioni, descritte in [Articolazioni e abbellimenti], pagina 124.

I comandi di arcata, `\upbow` (su) e `\downbow` (giù), vengono usati con le legature di portamento nel seguente modo:
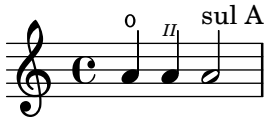
```
\relative { c''4(\downbow d) e(\upbow f) }
```



Si possono usare i numeri romani come numeri di corda (al posto dei numeri arabi racchiusi in un cerchio), come è spiegato in [Indicazioni di numero di corda], pagina 348.

Altrimenti, le indicazioni di corda possono essere mostrate anche tramite comandi markup; le articolazioni possono indicare anche corde vuote.

```
a'4 \open
\romanStringNumbers
a'\2
a'2^\markup { \small "sul A" }
```




## Comandi predefiniti

`\downbow`, `\upbow`, `\open`, `\romanStringNumbers`.

## Vedi anche
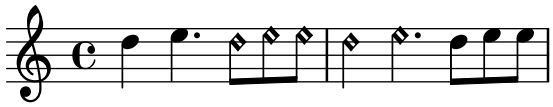
Guida alla notazione: [Articolazioni e abbellimenti], pagina 124, [Indicazioni di numero di corda], pagina 348, [Legature di portamento], pagina 135.

## Armonici

*Armonici naturali*
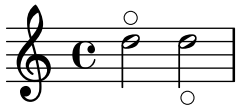
Gli armonici naturali possono essere rappresentati in vari modi. Una testa di nota romboidale indica generalmente che bisogna toccare la corda dove si prenderebbe la nota se questa non fosse romboidale.

```
\relative d'' {
  d4 e4.
  \harmonicsOn
  d8 e e
  d4 e4.
  \harmonicsOff
  d8 e e
}
```



Altrimenti si può mostrare una testa di nota normale all'altezza da riprodurre insieme a un piccolo cerchio che indica che deve essere suonata come armonico:

```
d''2^\flageolet d''_\flageolet
```
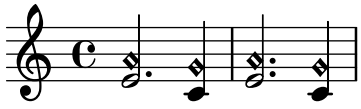


Si può creare anche un cerchio più piccolo, vedi l'elenco di frammenti in [Riferimenti per archi senza tasti], pagina 344.

*Armonici artificiali*

Gli armonici artificiali sono rappresentati da due note, una con una testa normale per indicare la posizione in cui sfiorare la corda e un'altra con una testa romboidale vuota per indicare la posizione dell'armonico.

Gli armonici artificial indicati con `\harmonic` non mostrano i punti. Occorre impostare la proprietà di contesto `harmonicDots` per far apparire i punti.

```
\relative e' {
  <e a\harmonic>2.  <c g'\harmonic>4
  \set harmonicDots = ##t
  <e a\harmonic>2.  <c g'\harmonic>4
}
```



### Vedi anche

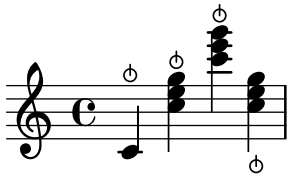
Glossario musicale: Sezione “armonici” in *Glossario Musicale*.

Guida alla notazione: [Teste di nota speciali], pagina 39, [Riferimenti per archi senza tasti], pagina 344.

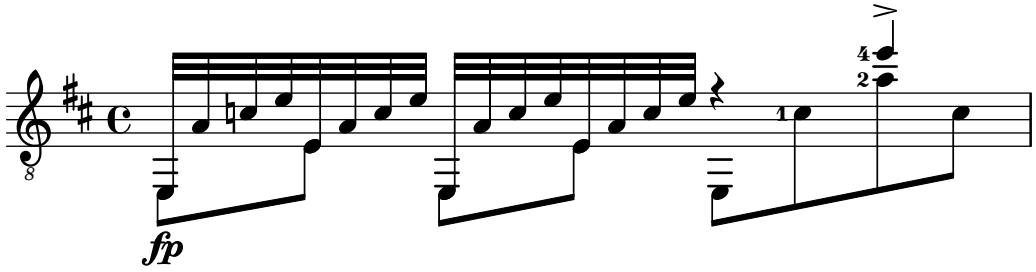
### Pizzicato alla Bartók
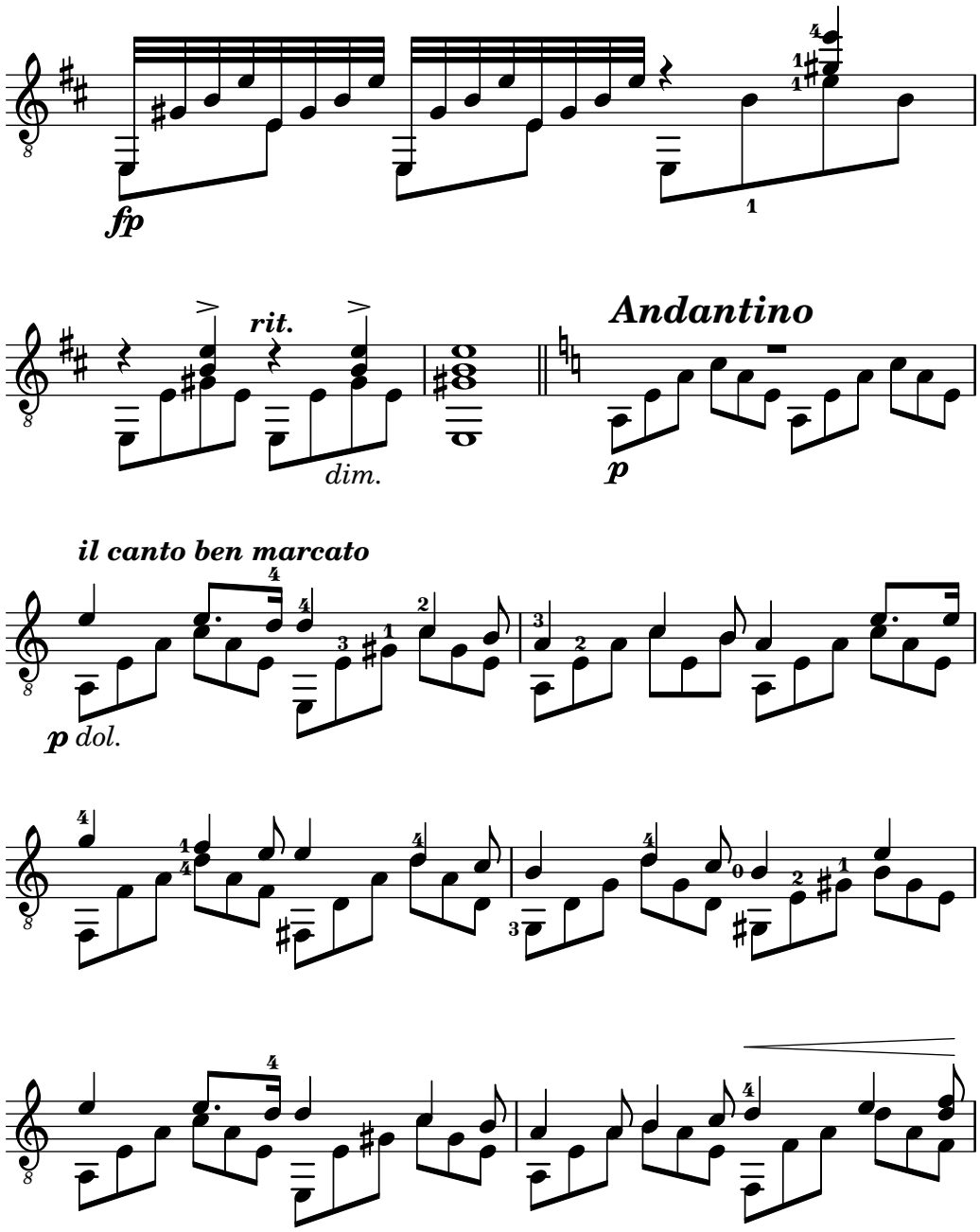
Il pizzicato alla Bartók, in inglese “snap pizzicato” (pizzicato a schiocco), è un tipo di pizzicato in cui la corda viene deliberatamente pizzicata verticalmente (invece che di lato) in modo che sbatta sulla tastiera.

```
\relative {
  c'4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



## 2.4 Strumenti a corde con tasti






Questa sezione tratta vari aspetti della notazione musicale che riguardano esclusivamente gli strumenti a corde con tasti.

### 2.4.1 Notazione comune per strumenti a corde con tasti

Questa sezione presenta la notazione comune che riguarda esclusivamente gli strumenti a corde con tasti.

### Riferimenti per strumenti a corde con tasti

La musica per strumenti a corde con tasti viene normalmente scritta su un rigo singolo, o nella notazione musicale tradizionale o in intavolatura. Talvolta i due tipi sono combinati; nella musica popolare è molto comune usare i diagrammi degli accordi sopra un rigo di notazione tradizionale. La chitarra e il banjo sono strumenti traspositori, che suonano un'ottava più bassa di quella in cui vengono scritti. Le partiture per questi strumenti devono usare la chiave `"treble_8"` (per ottenere l'output MIDI corretto usare `\transposition c`). Altri elementi relativi agli strumenti a corde con tasti sono trattati altrove:

- Le diteggiature vengono indicate come spiegato in [Indicazioni di diteggiatura], pagina 226.

- Istruzioni per le legature di valore *Laissez vibrer* così come per le legature di valore su arpeggi e tremoli si trovano in [Legature di valore], pagina 55.
- Istruzioni su come gestire voci multiple si trovano in [Risoluzione delle collisioni], pagina 179.
- Istruzioni per indicare gli armonici si trovano in [Armonici], pagina 345.

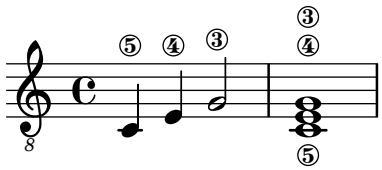
## Vedi anche

Guida alla notazione: [Indicazioni di diteggiatura], pagina 226, [Legature di valore], pagina 55, [Risoluzione delle collisioni], pagina 179, [Nomi degli strumenti], pagina 210, [Scrivere la musica in parallelo], pagina 191, [Arpeggio], pagina 147, Sezione A.14 [Elenco delle articolazioni], pagina 764, [Chiave], pagina 17, [Trasporto strumentale], pagina 27.

## Indicazioni di numero di corda

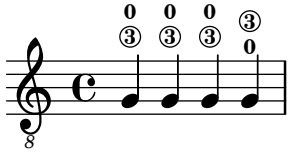
La corda sulla quale suonare una nota può essere indicata aggiungendo ***numero*** a una nota.

```
\clef "treble_8"
c4\5 e\4 g2\3
<c\5 e\4 g\3>1
```



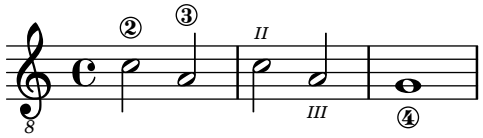
Se si usano insieme diteggiature e indicazioni di corda, il loro posizionamento è regolato dall'ordine in cui i due elementi appaiono nel codice *solo* se si trovano all'interno di un accordo esplicito: se applicate a un intero accordo o a singole note *fuori* da un accordo, le diteggiature vengono posizionate con un meccanismo diverso.

```
\clef "treble_8"
g4\3-0
g-0\3
<g\3-0>
<g-0\3>
```



I numeri di corda possono anche essere rappresentati da numeri romani, come è abituale con gli strumenti a corde senza tasti, e posizionati sotto il rigo invece che sopra.

```
\clef "treble_8"
c'2\2
a\3
\romanStringNumbers
c'\2
\set stringNumberOrientations = #'(down)
a\3
\arabicStringNumbers
g1\4
```

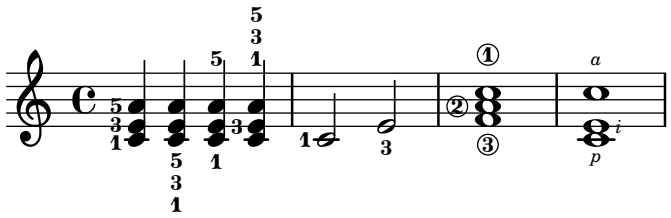


## Frammenti di codice selezionati

*Controllare il posizionamento delle diteggiature di un accordo*

Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi <> anche per le note singole. Si può impostare in modo simile l'orientamento dei numeri di corda e delle diteggiature della mano destra.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger #1 e\rightHandFinger #2 c'\rightHandFinger #4 >
}
```
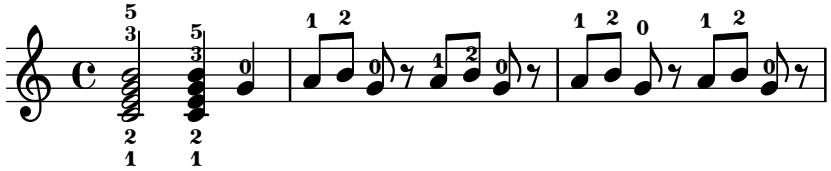


*Far sì che la diteggiatura appaia dentro il rigo*

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo; questo comportamento tuttavia può essere disabilitato. Occorre fare attenzione alle situazioni in cui le diteggiature e i gambi sono rivolti nella stessa direzione: normalmente le diteggiature evitano soltanto i gambi con travature. Questa impostazione predefinita può essere cambiata in modo da evitare tutti i gambi oppure nessuno. L'esempio seguente mostra queste due opzioni, così come tornare al comportamento predefinito.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 g'-0
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
```

```
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
}
```



## Comandi predefiniti

`\arabicStringNumbers`, `\romanStringNumbers`.

## Vedi anche

Guida alla notazione: [Indicazioni di diteggiatura], pagina 226.

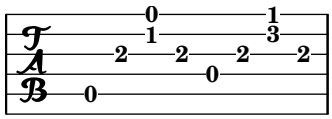
Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StringNumber” in *Guida al Funzionamento Interno*, Sezione “Fingering” in *Guida al Funzionamento Interno*.

## Intavolature predefinite

La musica per gli strumenti a corde pizzicate si scrive con un sistema di notazione basato sulle dita/tocco ovvero l’intavolatura (o tablatura). Diversamente dalla notazione tradizionale, le altezze non sono rappresentate da teste di nota, ma da numeri (o da simboli simili a lettere nell’intavolatura antica). Le linee del rigo nell’intavolatura indicano la corda su cui suonare la nota, mentre un numero posto su una linea del rigo indica il tasto su cui premere la corda corrispondente. Le note da suonare simultaneamente sono allineate verticalmente. Per impostazione predefinita, la corda 1 è la corda più alta e corrisponde alla linea più alta del rigo dell’intavolatura (`TabStaff`). L’accordatura predefinita delle corde di `TabStaff` è l’accordatura standard per chitarra (a sei corde). Le note appaiono in intavolatura, usando i contesti `TabStaff` e `TabVoice`. Viene aggiunta automaticamente una chiave calligrafica per l’intavolatura.

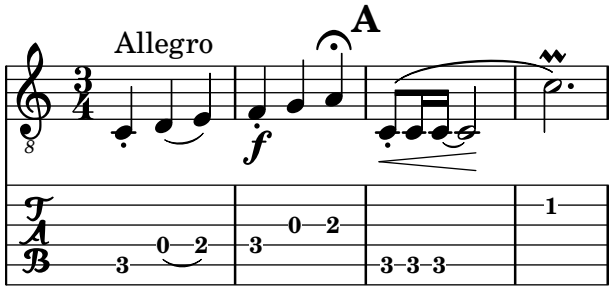
```
\new TabStaff \relative {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```



L’aspetto predefinito per le intavolature non prevede alcun simbolo di durata né altri simboli musicali, come per esempio i segni di espressione.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ 2\!
  c'2.\prall\)
}
```

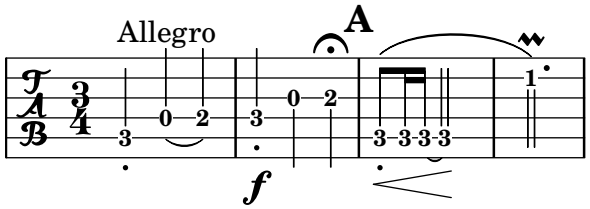
```
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff { \symbols }
  >>
}
```




Se si desidera che tutti i simboli usati nella notazione tradizionale appaiano anche nell'intavolatura, occorre applicare il comando `\tabFullNotation` in un contesto `TabStaff`. Tenere presente che nell'intavolatura le note da due quarti hanno un doppio gambo per distinguerle dalle note di un quarto.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c~ 2\!
  c'2.\prall\)
}

\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```

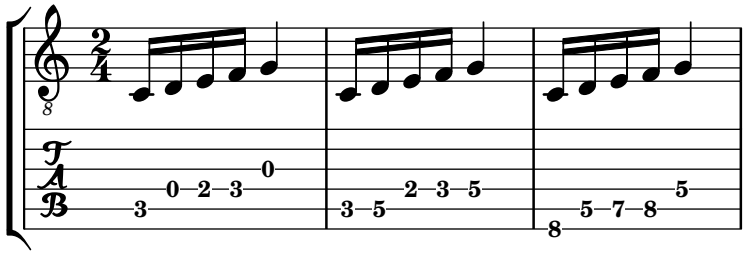



Di norma le altezze vengono assegnate alla posizione più bassa della tastiera (prima posizione). Viene data priorità alle corde a vuoto. Per far sì che una certa altezza venga suonata su una corda specifica, aggiungere un'indicazione di numero di corda al nome dell'altezza. Per nascondere le indicazioni di numero di corda nella notazione tradizionale, sovrascrivere lo stampo (stencil) corrispondente.

Spesso è più comodo definire la posizione sulla tastiera modificando la proprietà `minimumFret`, il cui valore predefinito è 0.

Anche quando `minimumFret` è impostato esplicitamente, vengono usate le corde a vuoto ogni volta che è possibile. È possibile modificare questo comportamento impostando `restrainOpenStrings` su `#t`.

```
\layout { \omit Voice.StringNumber }
\new StaffGroup <<
   \new Staff \relative {
     \clef "treble_8"
     \time 2/4
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     c,16 d e f g4
   }
   \new TabStaff \relative {
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     \set TabStaff.minimumFret = #5
     \set TabStaff.restrainOpenStrings = ##t
     c,16 d e f g4
   }
>>
```



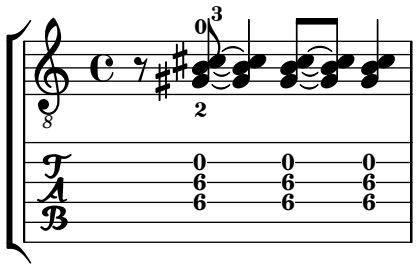
Gli accordi possono essere ripetuti col simbolo di ripetizione accordo `q`, che però rimuove i numeri di corda e le diteggiature. Se usato nelle intavolature, può quindi creare problemi. Converrà quindi eseguire il comando

```
\chordRepeats #'(string-number-event fingering-event)
```

esplicitamente sulle espressioni musicali dell'intavolatura che usano la [Ripetizione di un accordo], pagina 171. Questo comando è così comune che è disponibile come `\tabChordRepeats`.

```
guitar = \relative {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ 8 q4
}

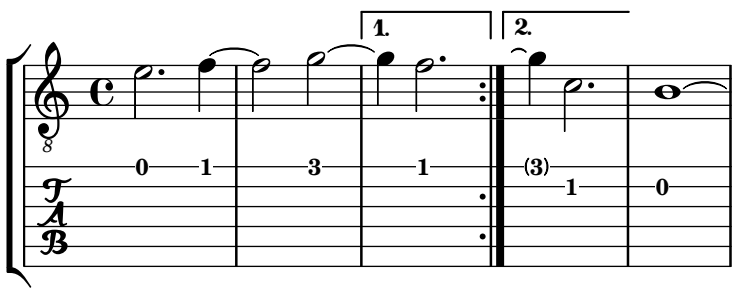
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
```

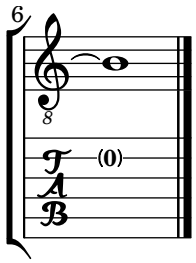


Le legature di valore spezzate da un a capo appaiono tra parentesi. Lo stesso vale per la seconda alternativa di una ripetizione.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~
  }
  \alternative {
     { g4 f2. }
     { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```

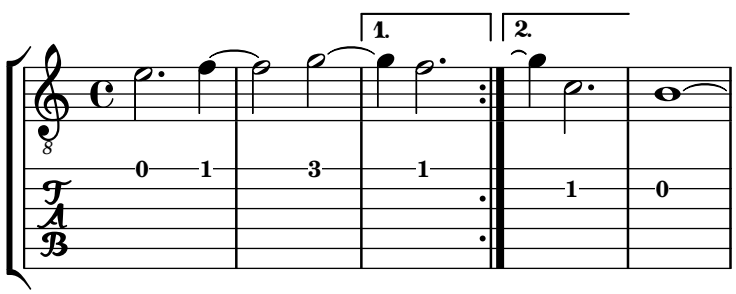


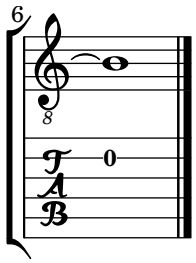


Il comando `\hideSplitTiedTabNotes` annulla il comportamento di racchiudere i numeri di tasto tra parentesi:

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

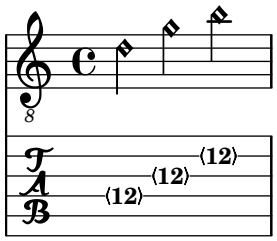
\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
  indent = #0
  ragged-right = ##t
  }
}
```





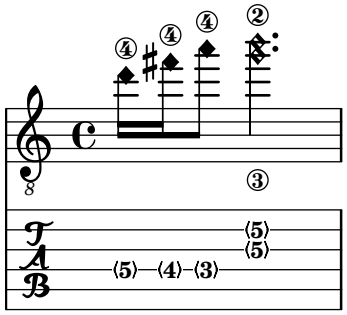

Gli armonici possono essere inseriti in un'intavolatura nelle altezze in cui suonano:

```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```



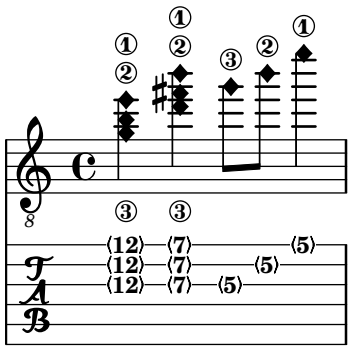

Il comando `\harmonic` deve essere sempre attaccato a note singole (possibilmente dentro a un costrutto per accordo) invece che ad accordi interi. Ha senso usarlo solo per gli armonici sul dodicesimo tasto. Tutti gli altri armonici devono essere calcolati da LilyPond. Ciò si ottiene indicando il tasto in cui il dito della mano che preme i tasti debba toccare una corda.

```
fretHarmonics = {
  \harmonicByFret #5 d16\4
  \harmonicByFret #4 d16\4
  \harmonicByFret #3 d8\4
  \harmonicByFret #5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \fretHarmonics
    }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Gli armonici possono essere calcolati anche definendo il rapporto delle lunghezze della corda sopra e sotto la diteggiatura dell'armonico.

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \ratioHarmonics
    }
    \new TabStaff { \ratioHarmonics }
  >>
}
```

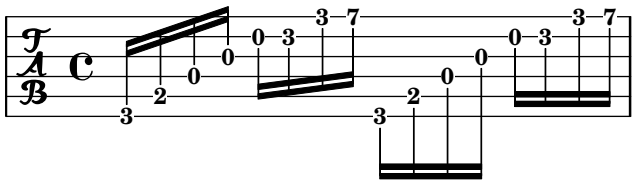


## Frammenti di codice selezionati

*Comportamento di gambi e travature in intavolatura*

La direzione dei gambi nell'intavolatura è regolata nello stesso modo della notazione tradizionale. Le travature possono essere rese orizzontali, come illustrato in questo esempio.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
    g,,16 b d g b d g b
  }
}
```



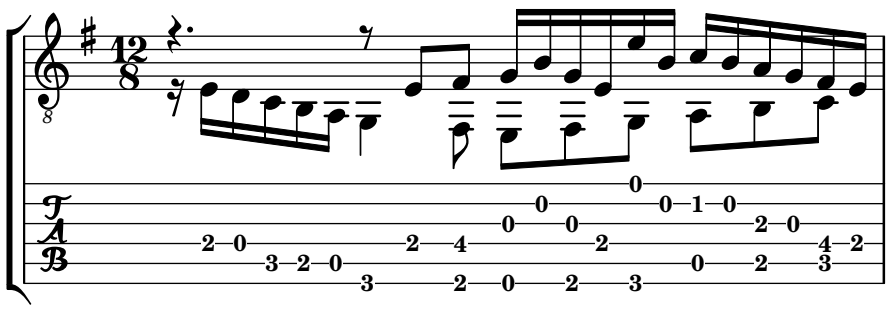
*Polifonia in intavolatura*

La polifonia in `TabStaff` funziona proprio come nel rigo normale.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  <<
    \new StaffGroup = "tab with traditional" <<
      \new Staff = "guitar traditional" <<
        \clef "treble_8"
        \new Voice = "upper" \upper
        \new Voice = "lower" \lower
      >>
      \new TabStaff = "guitar tab" <<
        \new TabVoice = "upper" \upper
        \new TabVoice = "lower" \lower
      >>
    >>
  >>
}
```



*Armonici su corde a vuoto in intavolatura*

Questo frammento mostra come scrivere armonici su corde a vuoto.

```
openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = #3
  \override TextSpanner.dash-fraction = #0.3
  \override TextSpanner.dash-period = #1
```

```
%first harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "1st harm. "
\harmonicByFret #12 e,2\6\startTextSpan
\harmonicByRatio #1/2 e,\6\stopTextSpan

%second harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "2nd harm. "
\harmonicByFret #7 e,\6\startTextSpan
\harmonicByRatio #1/3 e,\6
\harmonicByFret #19 e,\6
\harmonicByRatio #2/3 e,\6\stopTextSpan
%\harmonicByFret #19 < e,\6 a,\5 d\4 >
%\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >

%third harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "3rd harm. "
\harmonicByFret #5 e,\6\startTextSpan
\harmonicByRatio #1/4 e,\6
\harmonicByFret #24 e,\6
\harmonicByRatio #3/4 e,\6\stopTextSpan
\break

%fourth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "4th harm. "
\harmonicByFret #4 e,\6\startTextSpan
\harmonicByRatio #1/5 e,\6
\harmonicByFret #9 e,\6
\harmonicByRatio #2/5 e,\6
\harmonicByFret #16 e,\6
\harmonicByRatio #3/5 e,\6\stopTextSpan

%fifth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "5th harm. "
\harmonicByFret #3 e,\6\startTextSpan
\harmonicByRatio #1/6 e,\6\stopTextSpan
\break

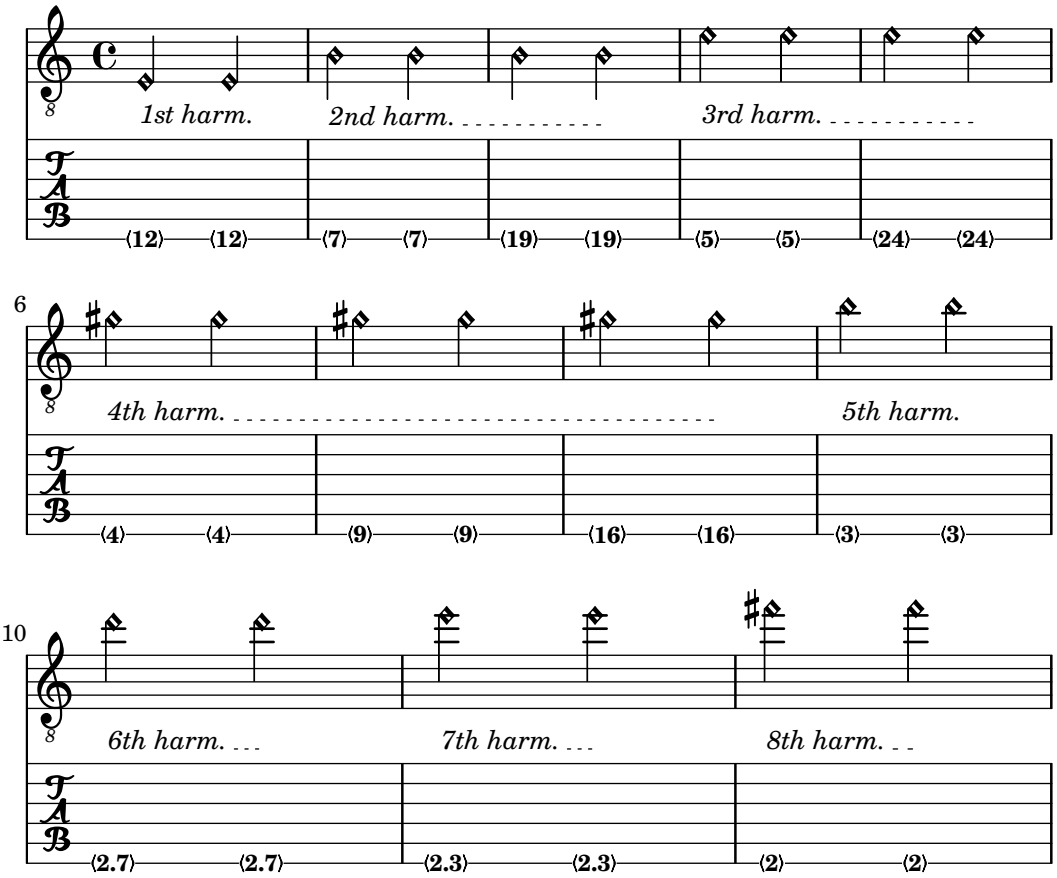
%sixth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "6th harm. "
\harmonicByFret #2.7 e,\6\startTextSpan
\harmonicByRatio #1/7 e,\6\stopTextSpan

%seventh harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "7th harm. "
```

```
  \harmonicByFret #2.3 e,\6\startTextSpan
  \harmonicByRatio #1/8 e,\6\stopTextSpan

  %eighth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "8th harm. "
  \harmonicByFret #2 e,\6\startTextSpan
  \harmonicByRatio #1/9 e,\6\stopTextSpan
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}
```



*Armonici su corde premute in intavolatura*

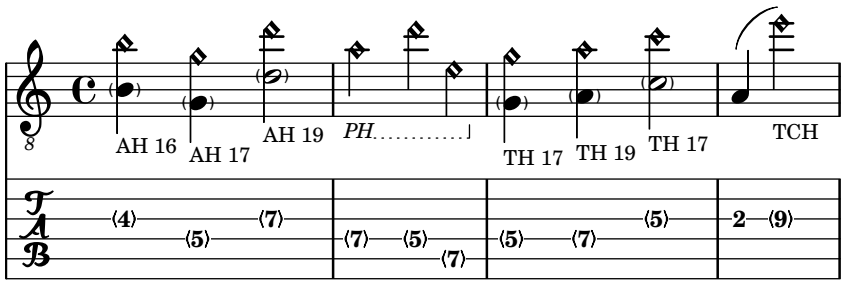
Questo frammento mostra come scrivere su intavolatura armonici su corde premute.

```
pinchedHarmonics = {
   \textSpannerDown
   \override TextSpanner.bound-details.left.text =
      \markup {\halign #-0.5 \teeny "PH" }
      \override TextSpanner.style =
         #'dashed-line
   \override TextSpanner.dash-period = #0.6
   \override TextSpanner.bound-details.right.attach-dir = #1
   \override TextSpanner.bound-details.right.text =
      \markup { \draw-line #'(0 . 1) }
   \override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup { \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup { \teeny "AH 17" }
  <\parenthesize d' d''\harmonic>2_\markup { \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d''\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup { \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup { \teeny "TH 19" }
  <\parenthesize c'\3 c''\harmonic>2_\markup { \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e''\harmonic>2. )_\markup { \teeny "TCH" }
}

frettedStrings = {
  %artificial harmonics (AH)
  \harmonicByFret #4 g4\3
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 g2\3
  %pinched harmonics (PH)
  \harmonicByFret #7 d2\4
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 a4\5
  %tapped harmonics (TH)
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 d4\4
  \harmonicByFret #5 g2\3
  %touch harmonics (TCH)
  a4 \harmonicByFret #9 g2.\3
}
```

```
\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}
```



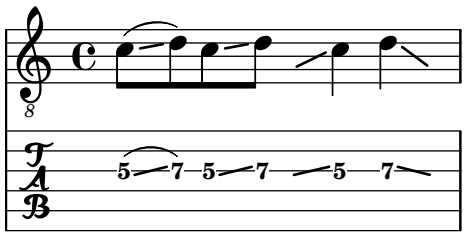

*Slides in tablature*

Slides can be typeset in both `Staff` and `TabStaff` contexts:

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
  \stemDown \hideNotes
  g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = #4
      \override Glissando.springs-and-rods =
                          #ly:spanner::set-spacing-rods
```

```
      \override Glissando.thickness = #2
      \omit StringNumber
      % or:
      %\override StringNumber.stencil = ##f
    }
  }
}
```



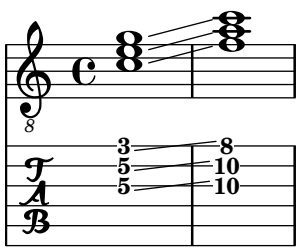
*Glissando di accordi in intavolatura*

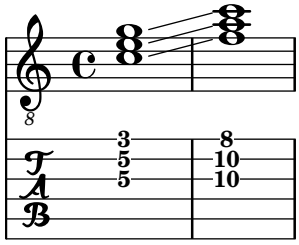
I glissati (o slide) di accordi vengono indicati sia nel rigo (contesto `Staff`) sia nell'intavolatura (contesto `TabStaff`). I numeri di corda sono necessari per `TabStaff`, perché i calcoli automatici della corda sono diversi per gli accordi e per le note singole.

```
myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \myMusic
  >>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \with { \override Glissando.style = #'none } {
      \myMusic
    }
  >>
}
```

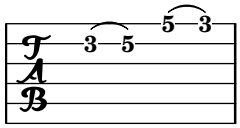




*Martellato e strappato*

Il martellato (hammer on) e lo strappato (pull off) si possono ottenere con le legature di portamento.

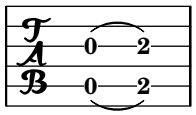
```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



*Martellato e strappato usando le voci*

L'arco del martellato o dello strappato è rivolto in alto nella prima e terza voce, mentre è rivolto in basso nella seconda e quarta voce.

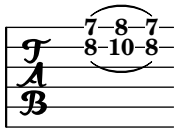
```
\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}
```



*Martellato e strappato usando accordi*

Quando il martellato o lo strappato si applicano a delle note in un accordo, viene disegnato un solo arco. Ma è possibile avere un "doppio arco" impostando la proprietà `doubleSlurs` su `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```



## Vedi anche

Guida alla notazione: [Ripetizione di un accordo], pagina 171, [Glissando], pagina 142, [Armonici], pagina 345, [Gambi], pagina 232, [Ripetizioni ricopiate], pagina 162.

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

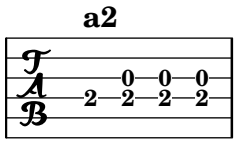
Guida al funzionamento interno: Sezione “TabNoteHead” in *Guida al Funzionamento Interno*, Sezione “TabStaff” in *Guida al Funzionamento Interno*, Sezione “TabVoice” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Gli accordi non sono gestiti in un modo specifico, dunque il selettore automatico delle corde potrebbe selezionare la stessa corda per due note di un accordo.

Per poter gestire `\partCombine`, un rigo `TabStaff` deve usare voci create appositamente:

```
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 }
  >>
>>
```




Gli effetti speciali per chitarra sono limitati ad armonici e slide (glissati).

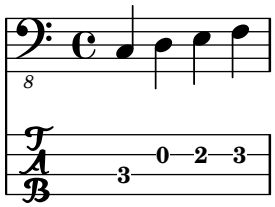
## Intavolature personalizzate

L’intavolatura di LilyPond calcola automaticamente il tasto per una nota in base alla corda a cui è assegnata. Per far ciò, deve essere specificata l’accordatura usando la proprietà `stringTunings`.

LilyPond ha accordature predefinite per banjo, mandolino, chitarra, basso, ukulele, violino, viola, violoncello e contrabbasso. LilyPond imposta automaticamente la corretta trasposizione per le accordature predefinite. L’esempio seguente è per basso, che suona un’ottava più bassa di come è scritto.

```
<<
  \new Voice \with {
    \omit StringNumber
  } {
    \clef "bass_8"
    \relative {
      c,4 d e f
    }
  }
  \new TabStaff \with {
    stringTunings = #bass-tuning
  } {
    \relative {
```

```
      c,4 d e f
    }
  }
>>
```



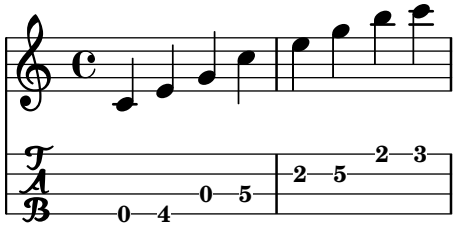
L'accordatura predefinita è `guitar-tuning`, ovvero l'accordatura standard EADGBE. Altre accordature predefinite sono `guitar-open-g-tuning`, `mandolin-tuning` e `banjo-open-g-tuning`. Tutte le accordature predefinite si trovano in `ly/string-tunings-init.ly`.

Si può definire qualsiasi accordatura tramite la funzione `\stringTuning`, che imposta la proprietà `stringTunings` per il contesto corrente.

Prende come argomento un accordo che definisce le altezze di ciascuna corda dell'accordatura. Le note dell'accordo sono in modalità di ottava assoluta, vedi [Ottava assoluta], pagina 1. La corda col numero più alto (generalmente la corda più grave) deve apparire per prima nell'accordo. Per esempio, possiamo definire un'accordatura per uno strumento a quattro corde con le altezze `a''`, `d''`, `g'` e `c'`:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set Staff.stringTunings = \stringTuning <c' g' d'' a''>
    \mynotes
  }
>>
```



La proprietà `stringTunings` viene utilizzata anche da `FretBoards` per calcolare i diagrammi automatici dei tasti.

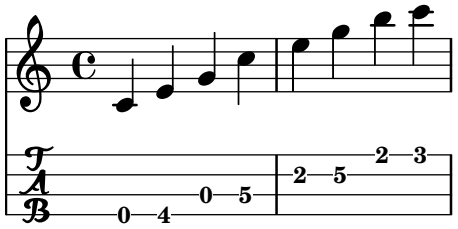
Le accordature vengono usate per calcolare i diagrammi dei tasti predefiniti (vedi [Diagrammi dei tasti predefiniti], pagina 378).

L'esempio precedente può essere scritto anche così:

```
custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    \set TabStaff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



Internamente un'accordatura è una lista Scheme di altezze, una per ciascuna corda, ordinata da un numero di corda che va da 1 a N, dove la corda 1 è in cima al rigo dell'intavolatura e la corda N è in fondo. Normalmente ciò significa che l'ordine va dall'altezza più alta a quella più bassa, con l'eccezione di alcuni strumenti (come l'ukulele) che non hanno corde ordinate per altezza.

Un'altezza in una lista per accordatura, nella terminologia di LilyPond, è un oggetto altezza, creato dalla funzione Scheme `ly:make-pitch` (vedi Sezione A.22 [Funzioni Scheme], pagina 821).

`\stringTuning` crea tale oggetto a partire da un accordo.

LilyPond calcola automaticamente il numero di linee del rigo dell'intavolatura (`TabStaff`) e il numero di corde in una tastiera (`FretBoard`) calcolata automaticamente in base al numero di elementi in `stringTunings`.

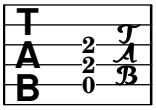
Per far sì che tutti i contesti TabStaff usino la stessa accordatura personalizzata, si può usare

```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

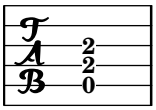
Si può usare anche una chiave moderna per l'intavolatura.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
```

```
  <a, e a>1
}
```



```
2
```



La chiave moderna per l'intavolatura supporta tablature da 4 a 7 corde.

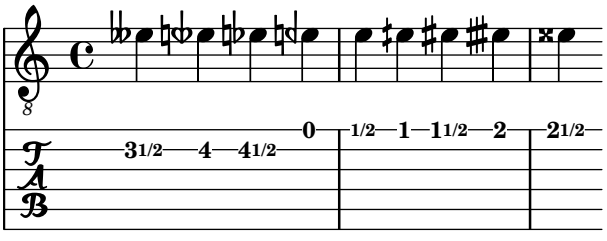
`TabStaff` supporta i microtoni come i quarti di tono, che possono essere suonati piegando le corde (bending). Occorre impostare `supportNonIntegerFret = ##t` nel contesto `Score`. I microtoni tuttavia non sono supportati nel contesto `FretBoards`.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

custom-tuning = \stringTuning <e, a, d ges beh eeh'>

mus = \relative {
  eeses'4
  eeseh
  ees
  eeh
  e
  eih
  eis
  eisih
  eisis
}

<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \with { stringTunings = \custom-tuning } \mus
>>
```



## Vedi anche

Guida alla notazione: [Ottava assoluta], pagina 1, [Diagrammi dei tasti predefiniti], pagina 378, Sezione A.22 [Funzioni Scheme], pagina 821.

File installati: `ly/string-tunings-init.ly`, `scm/tablature.scm`.

Frammenti: Sezione "Fretted strings" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Tab_note_heads_engraver" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

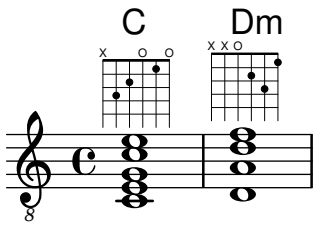
I calcoli automatici dell'intavolatura non funzionano correttamente nella maggior parte dei casi per strumenti (come l'ukulele) in cui le altezze delle corde non variano in modo uniforme col numero di corda.

## Diagrammi dei tasti

I diagrammi dei tasti possono essere collegati alla musica come testo (markup) attaccato alla nota desiderata. Il testo contiene l'informazione sul diagramma dei tasti desiderato. Esistono tre diverse interfacce per inserire un diagramma dei tasti (fret-diagram): standard, conciso (terse) e ridondante (verbose). Le tre interfacce producono testi graficamente equivalenti, ma hanno diverse quantità di informazioni nella stringa di testo. I dettagli sulla sintassi delle diverse stringhe di markup usate per definire i diagrammi dei tasti si trovano in Sezione A.11.5 [Instrument Specific Markup], pagina 746.

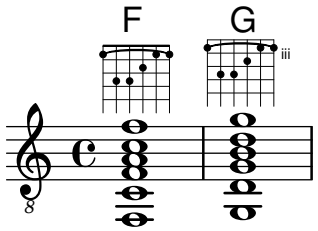
Il comando per il diagramma dei tasti standard, `\fret-diagram`, richiede il numero di corda e il numero di tasto per ogni punto da posizionare sulla corda. Per esempio `5-3` indica che la quinta corda è premuta sul terzo tasto. Si possono indicare anche corde a vuoto o non suonate (mute) attaccando rispettivamente `-o` e `-x` al numero di corda.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



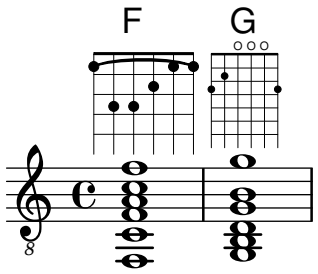
Con fret-diagram si possono aggiungere indicazioni di barré al diagramma inserendo all'inizio della stringa, per esempio, `c:4-1-3`, che indica che il barré va dalla quarta alla prima corda e si trova sul terzo tasto.

```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram "c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
    }
  }
>>
```



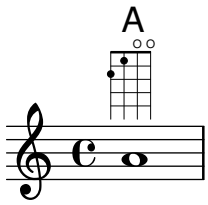

La dimensione del diagramma dei tasti si può variare col modificatore `s:`. Un numero superiore a 1 aumenterà la dimensione predefinita, mentre uno inferiore a 1 la ridurrà. Il numero dei tasti del diagramma può essere cambiato col modificatore `h:` seguito dal numero di tasti desiderato.

```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram "h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
    }
  }
>>
```



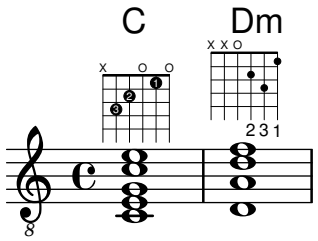

Il numero di corde in un diagramma creato con `\fret-diagram` può essere cambiato per adattarsi a diversi strumenti come banjo e ukulele. Il modificatore è `w:` seguito dal numero di corde desiderato.

```
<<
  \new ChordNames {
    \chordmode {
      a1
    }
  }
  \new Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram "w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



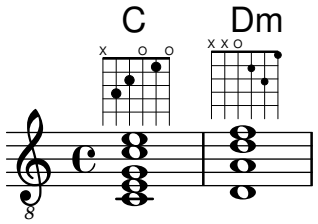

Si possono aggiungere indicazioni di diteggiatura appendendo un terzo numero, dopo quelli della corda e del tasto. La posizione delle diteggiature è controllata dal modificatore `f:`, che può prendere due valori: 1, per mostrare i numeri di diteggiatura sul tasto; 2 per mostrarli alla base della rispettiva corda (posizione predefinita).

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
    }
  }
>>
```




Il comando `fret-diagram` consente anche di modificare il raggio del punto (modificatore `d:`) e la sua posizione rispetto al tasto (modificatore `p:`).

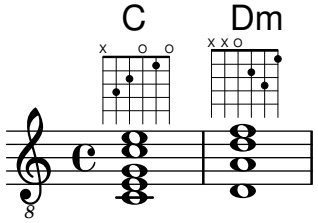
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```




Il comando markup `\fret-diagram-terse` ha un formato più conciso perché omette i numeri di corda, che sono calcolati implicitamente in base al punto e virgola che separa ciascuna corda. Il primo punto e virgola corrisponde al numero di corda più alto e l'ultimo punto e virgola corrisponde alla prima corda. Si possono indicare anche corde a vuoto, mute e con tasto premuto, come visto prima.

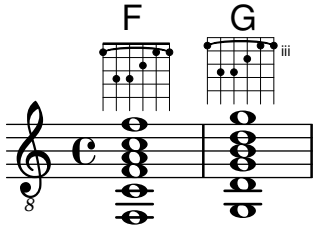
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3;2;o;1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2;3;1;"
    }
  }
```

```
>>
```




Nella stringa di `\fret-diagram-terse` si possono inserire indicazioni di barré:

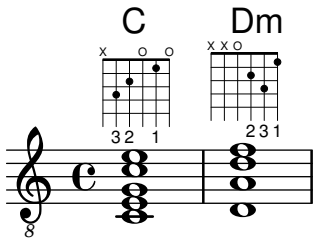
```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram-terse "1-(;3;3;2;1;1-);"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram-terse "3-(;5;5;4;3;3-);"
    }
  }
>>
```




`\fret-diagram-terse` accetta anche indicazioni di diteggiatura, che si troveranno in seconda posizione, dopo il numero del tasto.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3-3;2-2;o;1-1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2-2;3-3;1-1;"
    }
```

```
  }
>>
```

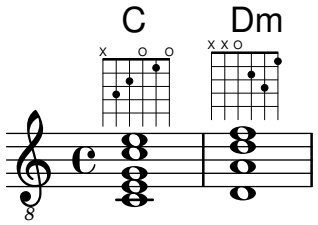



Quando si usa `\fret-diagram-terse`, altre proprietà del diagramma dei tasti possono essere regolate usando `\override`.

Solo un'indicazione per corda può essere inclusa in un markup fret-diagram-terse. Per avere indicazioni multiple per corda, usare una diagramma dei tasti o un markup fret-diagram-verbose.

Il comando `\fret-diagram-verbose` prende come argomento una lista Scheme. Ciascun elemento della lista indica un elemento da posizionare sul diagramma.

```
<<
    \new ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



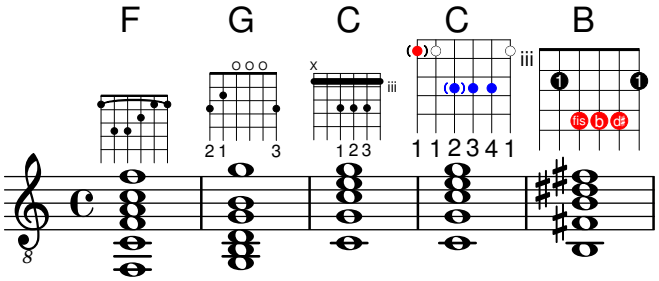

Nella lista di fret-diagram-verbose si possono inserire indicazioni di diteggiatura e barré. Solo in questa interfaccia è possibile inserire un'indicazione di capotasto per il diagramma. L'indicazione di capotasto è una barra spessa che copre tutte le corde. Il tasto col capotasto sarà il tasto più basso del diagramma.

I punti con le diteggiature possono essere colorati o racchiusi da parentesi; il colore delle parentesi può essere modificato in modo indipendente.

Si può anche inserire del testo di tipo markup dentro i punti.

```
<<
    \new ChordNames {
      \chordmode {
        f1 g c c b
      }
    }
  \new Staff {
    \clef "treble_8"
    \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
        (open 4)
        (open 3)
        (open 2)
        (place-fret 1 3 3)
      )
    }
    <c g c' e' g'>1^\markup {
      \fret-diagram-verbose #'(
        (capo 3)
        (mute 6)
        (place-fret 4 5 1)
        (place-fret 3 5 2)
        (place-fret 2 5 3)
      )
    }
    \override Voice.TextScript.size = 1.4
```

```
    <c g c' e' g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 1 red parenthesized default-paren-color)
        (place-fret 5 3 1 inverted)
        (place-fret 4 5 2 blue parenthesized)
        (place-fret 3 5 3 blue)
        (place-fret 2 5 4 blue)
        (place-fret 1 3 1 inverted)
      )
    }
    \override Voice.TextScript.size = 1.5
    <b, fis b dis' fis'>1^\markup {
      \override #'(fret-diagram-details . ((finger-code . in-dot)))
      \fret-diagram-verbose #`(
        (place-fret 5 2 1)
        (place-fret 4 4 "fis" red)
        (place-fret 3 4 "b" red)
        (place-fret
          2 4
          ,#{ \markup
                \concat {
                  \vcenter "d"
                  \fontsize #-5
                  \musicglyph "accidentals.sharp"} #}
          red)
        (place-fret 1 2 1)
      )
    }
  }
>>
```




Anche in questa interfaccia è possibile modificare tutte le altre proprietà del diagramma dei tasti tramite `\override`.

L'aspetto grafico di un diagramma dei tasti può essere personalizzato secondo le proprie preferenze attraverso le proprietà dell'interfaccia `fret-diagram-interface`. Tutti i dettagli si trovano in Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*. Per un diagramma dei tasti di tipo markup, le proprietà dell'interfaccia appartengono a `Voice.TextScript`.

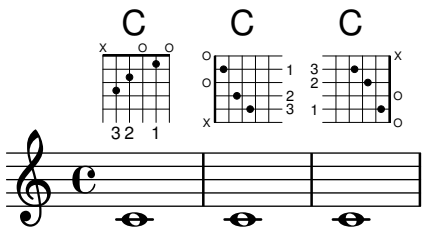
## Frammenti di codice selezionati

*Cambiare l'orientamento della tastiera*

I diagrammi dei tasti possono essere orientati in tre modi.

```
\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'
  }
>>
```




*Personalizzare il diagramma dei tasti di tipo markup*

Le proprietà del diagramma dei tasti si possono impostare tramite `'fret-diagram-details`. Per diagrammi di tipo markup, gli override possono essere applicati all'oggetto `Voice.TextScript` o direttamente al 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;" }

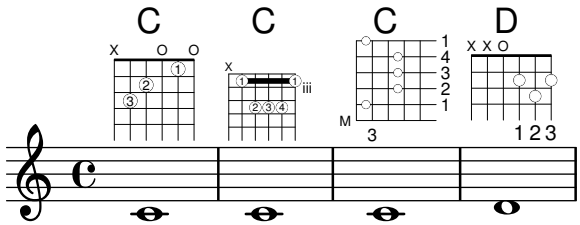
    %% C major for guitar, barred on third fret
```

```
   % 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;"
        }
      }
    }
>>
```




## Vedi anche

Guida alla notazione: Sezione A.11.5 [Instrument Specific Markup], pagina 746.

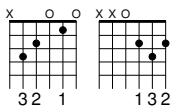
Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

## Diagrammi dei tasti predefiniti

I diagrammi dei tasti possono essere visualizzati attraverso il contesto `FretBoards`. Per impostazione predefinita, il contesto `FretBoards` mostrerà i diagrammi dei tasti salvati in una tabella dati:

```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
  }
}
```




I diagrammi dei tasti predefiniti sono contenuti nel file `predefined-guitar-fretboards.ly`. Si basano sulle altezze di un accordo e sul valore di `stringTunings` in uso. `predefined-guitar-fretboards.ly` contiene i diagrammi dei tasti predefiniti solo per l’accordatura standard per chitarra (`guitar-tuning`). Si possono aggiungere diagrammi predefiniti per altri strumenti o accordature seguendo l’esempio in `predefined-guitar-fretboards.ly`.

I diagrammi dei tasti per ukulele sono contenuti nel file `predefined-ukulele-fretboards.ly`.

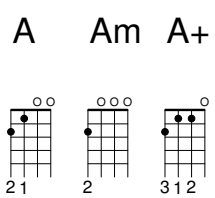
```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

\new ChordNames {
```

```
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}
```



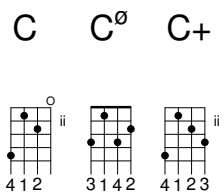

I diagrammi dei tasti per mandolino sono contenuti nel file `predefined-mandolin-fretboards.ly`.

```
\include "predefined-mandolin-fretboards.ly"

myChords = \chordmode { c1 c:m7.5- c:aug }

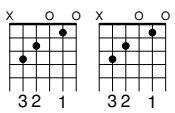
\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```




Le altezze degli accordi possono essere inserite sia come musica simultanea sia usando la modalità per accordi (vedi [Panoramica sulla modalità accordo], pagina 419).

```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```

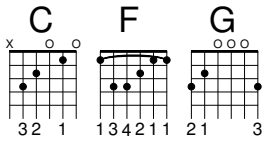



Di solito i nomi degli accordi e i diagrammi dei tasti appaiono insieme. Per farlo si pone un contesto `ChordNames` in parallelo con un contesto `FretBoards` assegnando a entrambi la stessa musica.

```
\include "predefined-guitar-fretboards.ly"
```

```
mychords = \chordmode {
  c1 f g
}

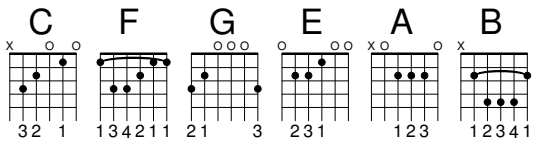
<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```




I diagrammi dei tasti predefiniti possono essere trasposti, purché un diagramma per l'accordo trasposto sia salvato nella tabella del diagramma.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 f g
}

mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \new ChordNames {
    \mychordlist
  }
  \new FretBoards {
    \mychordlist
  }
>>
```



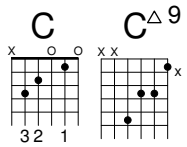

La tabella dei diagrammi dei tasti predefiniti per chitarra contiene otto accordi (maggiore, minore, aumentato, diminuito, settima dominante, settima maggiore, settima minore, nona dominante) per ognuna delle 17 tonalità.

La tabella dei diagrammi dei tasti per ukulele contiene questi accordi più ulteriori tre accordi (sesta maggiore, seconda sospesa e quarta sospesa). Un elenco completo dei diagrammi dei tasti predefiniti si trova in Sezione A.4 [Diagrammi degli accordi predefiniti], pagina 668. Se nella tabella non esiste una voce per un accordo, l'incisore FretBoards calcolerà un diagramma usando la funzionalità automatica descritta in [Diagrammi dei tasti automatici], pagina 388.

```
\include "predefined-guitar-fretboards.ly"
```

```
mychords = \chordmode{
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



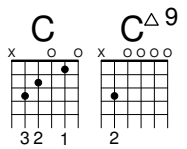

Si possono aggiungere diagrammi dei tasti alla tabella. Basta specificare la tabella del diagramma, l'accordo, l'accordatura e una definizione del diagramma. Di solito la tabella di riferimento sarà *default-fret-table*. La definizione di diagramma può essere una stringa di fret-diagram-terse o un elenco di fret-diagram-verbose.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        "x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



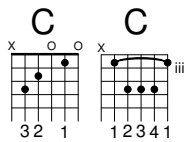

Diagrammi diversi per accordi con lo stesso nome possono essere salvati usando un'altezza con unn'ottava diversa. L'ottava diversa deve essere almeno due ottave sopra o sotto l'ottava predefinita, perché le ottave sopra e sotto sono usate per trasporre le tastiere.

```
\include "predefined-guitar-fretboards.ly"
```

```
\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```




Oltre ai diagrammi dei tasti, LilyPond contiene un elenco interno di forme di accordo. Queste forme sono diagrammi dei tasti che possono essere spostati lungo il manico in diverse posizioni per fornire diversi accordi. Si possono aggiungere nuove forme di accordo all'elenco interno e poi usarle per definire diagrammi dei tasti predefiniti. Dato che possono essere spostate in varie posizioni sul manico, le forme di un accordo normalmente non contengono corde a vuoto. Come per i diagrammi dei tasti, le forme possono essere inserite come stringhe di fret-diagram-terse o come elenchi di fret-diagram-verbose.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

\addChordShape #'powerf #guitar-tuning "1-1;3-3;3-4;x;x;x;"

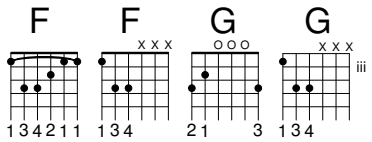
% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
                        \chordmode { f'' }
                        #guitar-tuning
                        #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
                        \chordmode { g'' }
                        #guitar-tuning
                        #(offset-fret 2 (chord-shape 'powerf guitar-tuning))

mychords = \chordmode {
  f1 f'' g g''
}

<<
```

```
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```




L'aspetto grafico di un diagramma dei tasti può essere personalizzato in base alle proprie preferenze attraverso le proprietà dell'interfaccia `fret-diagram-interface`. I dettagli si trovano in Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*. Per un diagramma dei tasti predefinito, le proprietà dell'interfaccia appartengono a `FretBoards.FretBoard`.

## Frammenti di codice selezionati

*Personalizzare la tastiera del diagramma dei tasti*

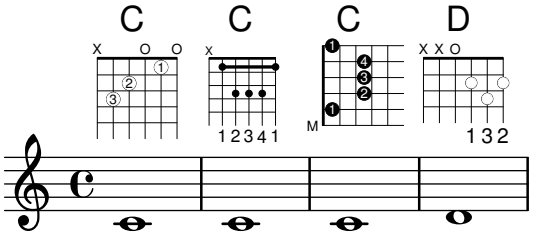
Le proprietà del diagramma dei tasti possono essere impostate tramite `'fret-diagram-details`. Per i diagrammi dell'oggetto FretBoard, gli override vengono applicati all'oggetto `FretBoards.FretBoard`. Come `Voice`, `FretBoards` è un contesto di basso livello, dunque può essere omesso negli override delle proprietà.

```
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        #"x;1-1-(;3-2;3-3;3-4;1-1-);"

% shorthand
oo = #(define-music-function
       (grob-path value)
       (list? scheme?)
       #{ \once \override $grob-path = #value #})

<<
  \new ChordNames {
    \chordmode { c1 | c | c | d }
  }
  \new FretBoards {
    % Set global properties of fret diagram
    \override FretBoards.FretBoard.size = #'1.2
    \override FretBoard.fret-diagram-details.finger-code = #'in-dot
    \override FretBoard.fret-diagram-details.dot-color = #'white
    \chordmode {
      c
      \oo FretBoard.size #'1.0
      \oo FretBoard.fret-diagram-details.barre-type #'straight
      \oo FretBoard.fret-diagram-details.dot-color #'black
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
      c'
      \oo FretBoard.fret-diagram-details.barre-type #'none
```

```
      \oo FretBoard.fret-diagram-details.number-type #'arabic
      \oo FretBoard.fret-diagram-details.orientation #'landscape
      \oo FretBoard.fret-diagram-details.mute-string #"M"
      \oo FretBoard.fret-diagram-details.label-dir #LEFT
      \oo FretBoard.fret-diagram-details.dot-color #'black
      c'
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
      \oo FretBoard.fret-diagram-details.dot-radius #0.35
      \oo FretBoard.fret-diagram-details.dot-position #0.5
      \oo FretBoard.fret-diagram-details.fret-count #3
      d
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>
```




*Definire diagrammi dei tasti predefiniti per altri strumenti*

Si possono aggiungere diagrammi dei tasti predefiniti per nuovi strumenti oltre a quelli standard per chitarra. Questo frammento mostra come farlo definendo una nuova accordatura e alcuni diagrammi predefiniti per il *quatro* venezuelano.

Mostra anche come includere le diteggiature negli accordi usati come punti di riferimento per la consultazione degli accordi e come mostrarle nel diagramma dei tasti e in `TabStaff`, ma non nella musica.

Questi diagrammi non sono trasponibili perché contengono informazioni sulle corde. È prevista una correzione in futuro.

```
% add FretBoards for the Cuatro
%   Note: This section could be put into a separate file
%      predefined-cuatro-fretboards.ly
%      and \included into each of your compositions

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
```

```
                          #cuatroTuning
                          #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                          #cuatroTuning
                          #"o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                          #cuatroTuning
                          #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                          #cuatroTuning
                          #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                          #cuatroTuning
                          #"2-2;o;1-1;o;"

% end of potential include file /predefined-cuatro-fretboards.ly


#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

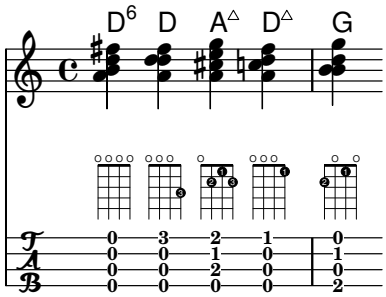
    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = 4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }
```

```
    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        #(ly:make-moment 1 16)
    }
  }
  \midi { }
}
```



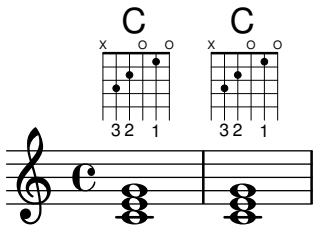

*Cambi di accordo nei diagrammi dei tasti*

Si può impostare il contesto FretBoards in modo che mostri il diagramma solo quando l'accordo cambia o all'inizio di una nuova linea.

```
\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```



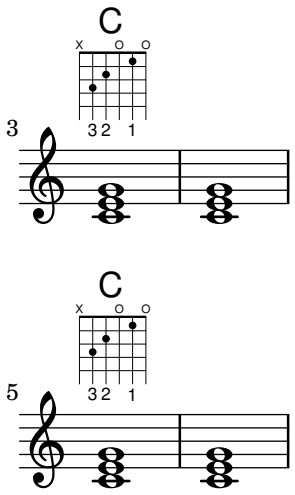



*Tabelle alternative per i diagrammi dei tasti*

Si possono creare tabelle alternative per i diagrammi dei tasti. Queste possono essere usate per avere diagrammi alternativi per uno stesso accordo.

Per poter usare una tabella alternativa, deve prima essere creata. Quindi si aggiungono i diagrammi alla tabella.

La tabella può essere vuota oppure ricopiata da una tabella esistente.

La tabella da usare nel mostrare i diagrammi predefiniti viene selezionata dalla proprietà `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"

% Make a blank new fretboard table
#(define custom-fretboard-table-one
   (make-fretboard-table))

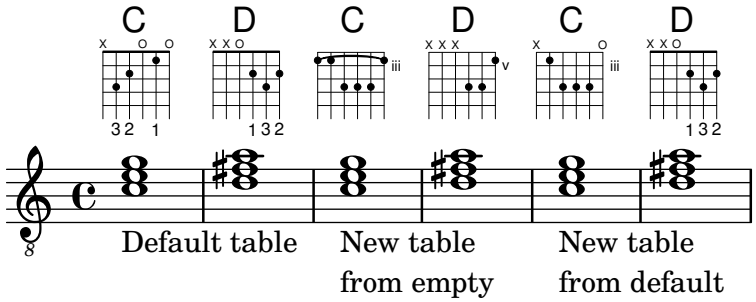
% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two
   (make-fretboard-table default-fret-table))

% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode {c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode {c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
```

```
        c1 | d1 |
        \set predefinedDiagramTable = #custom-fretboard-table-one
        c1 | d1 |
        \set predefinedDiagramTable = #custom-fretboard-table-two
        c1 | d1 |
      }
    }
    \new Staff {
      \clef "treble_8"
      <<
        \chordmode {
          c1 | d1 |
          c1 | d1 |
          c1 | d1 |
        }
        {
          s1_\markup "Default table" | s1 |
          s1_\markup \column {"New table" "from empty"} | s1 |
          s1_\markup \column {"New table" "from default"} | s1 |
        }
      >>
    }
>>
```




## Vedi anche

Guida alla notazione: [Intavolature personalizzate], pagina 364, [Diagrammi dei tasti automatici], pagina 388, [Panoramica sulla modalità accordo], pagina 419, Sezione A.4 [Diagrammi degli accordi predefiniti], pagina 668.

File installati: `ly/predefined-guitar-fretboards.ly`, `ly/predefined-guitar-ninth-fretboards.ly`, `ly/predefined-ukulele-fretboards.ly`, `ly/predefined-mandolin-fretboards.ly`.

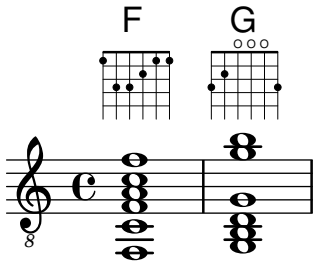
Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

## Diagrammi dei tasti automatici

I diagrammi dei tasti possono essere creati automaticamente dalle note inserite usando il contesto `FretBoards`. Se non è disponibile un diagramma predefinito per le note inserite nell’accordatura (`stringTunings`) attiva, questo contesto calcola le corde e i tasti che possono essere usati per suonare le note.

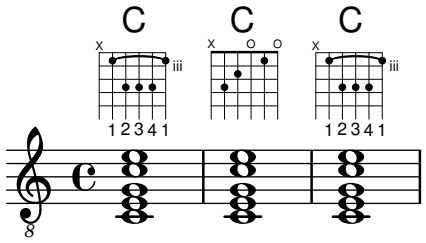
```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b' g'>1
  }
>>
```




Dato che nessun diagramma predefinito viene caricato senza impostarlo esplicitamente, il calcolo automatico dei diagrammi è il comportamento predefinito. Quando i diagrammi predefiniti vengono caricati, il calcolo automatico può essere abilitato e disabilitato con dei comandi predefiniti:

```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        "x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \new ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \new FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
```

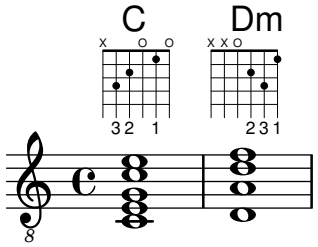
```
    <c e g c' e'>1
  }
>>
```




Talvolta il calcolatore dei diagrammi non sarà in grado di trovare un diagramma accettabile. Di solito il rimedio consiste nell'assegnare manualmente una nota a una corda. In molti casi, solo una nota deve essere posta esplicitamente su una corda; tutte le altre note verranno posizionate correttamente dal contesto `FretBoards`.

Si possono aggiungere le diteggiture ai diagrammi dei tasti dell'oggetto FretBoard.

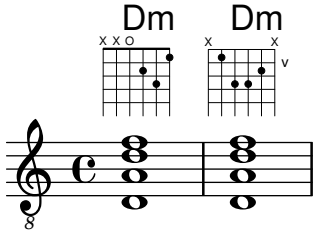
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <d a d' f'>1
  }
>>
```




Il tasto minimo da usare per calcolare le corde e i tasti per il contesto FretBoard si può impostare con la proprietà `minimumFret`.

```
<<
  \new ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \new FretBoards {
    <d a d' f'>1
```

```
    \set FretBoards.minimumFret = #5
    <d a d' f'>1
  }
  \new Staff {
    \clef "treble_8"
    <d a d' f'>1
    <d a d' f'>1
  }
>>
```




Le corde e i tasti per il contesto `FretBoards` dipendono dalla proprietà `stringTunings`, che funziona proprio come nel contesto TabStaff. Si rimanda a [Intavolature personalizzate], pagina 364, per informazioni sulla proprietà `stringTunings`.

L'aspetto grafico di un diagramma dei tasti può essere personalizzato secondo le proprie preferenze tramite le proprietà dell'interfaccia `fret-diagram-interface`. I dettagli si trovano in Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*. Per un diagramma dei tasti del contesto `FretBoards`, le proprietà dell'interfaccia appartengono a `FretBoards.FretBoard`.

## Comandi predefiniti

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

## Vedi anche

Guida alla notazione: [Intavolature personalizzate], pagina 364.

Frammenti: Sezione "Fretted strings" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "fret-diagram-interface" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

I calcoli automatici dei tasti non funzionano correttamente per gli strumenti con accordature non monotoniche.
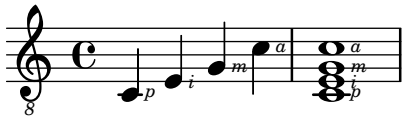
## Diteggiature della mano destra

Le diteggiature della mano destra *p-i-m-a* si inseriscono col comando `\rightHandFinger` seguito da un numero.

> **Nota:** Se si inserisce il numero in notazione Scheme, ricordarsi di aggiungere uno spazio prima del `>` di chiusura o simili.

```
\clef "treble_8"
c4\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c'\rightHandFinger #4
```

```
<c\rightHandFinger #1 e\rightHandFinger #2
 g\rightHandFinger #3 c'\rightHandFinger #4 >1
```



Per comodità, si può abbreviare `\rightHandFinger` in un comando più breve, per esempio `RH`,

```
RH=#rightHandFinger
```

## Frammenti di codice selezionati

*Posizionamento delle diteggiature della mano destra*

È possibile avere un maggior controllo sul posizionamento delle diteggiature della mano destra impostando una specifica proprietà, come illustrato nell'esempio seguente.

```
#(define RH rightHandFinger)

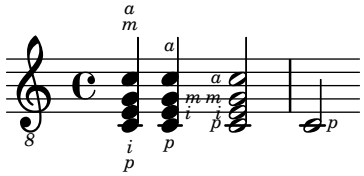
\relative c {
  \clef "treble_8"

  \set strokeFingerOrientations = #'(up down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >4

  \set strokeFingerOrientations = #'(left)
  <c\RH #1 e\RH #2 g\RH #3 c\RH #4 >2

  \set strokeFingerOrientations = #'(right)
  c\RH #1
}
```

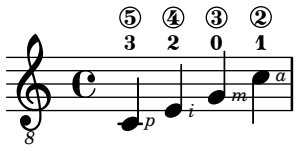


*Diteggiature, indicazioni di corda e diteggiature della mano destra*

Questo esempio combina la diteggiatura per la mano sinistra, le indicazioni di stringa e la diteggiatura della mano destra.

```
#(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
}
```




## Vedi anche

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StrokeFinger” in *Guida al Funzionamento Interno*.
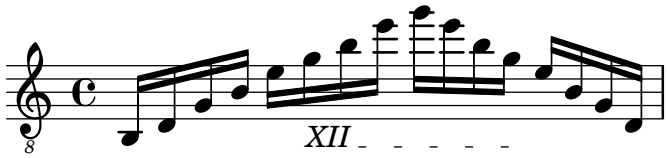
### 2.4.2 Chitarra

La maggior parte delle questioni relative alla musica per chitarra sono trattate nella sezione generale sugli strumenti a corde con tasti, ma ce ne sono altre che è meglio spiegare qui. Talvolta gli utenti desiderano creare dei canzonieri, ovvero dei documenti che hanno solo il testo e le indicazioni di accordo sopra di esso. Dal momento che LilyPond è un compositore tipografico musicale, non è consigliato per documenti che non contengono notazione musicale. Migliori alternative sono un programma di videoscrittura, un editor di testo oppure, per utenti esperti, un compositore tipografico come GuitarTeX.

## Indicazione di posizione e barré

Questo esempio illustra come mostrare la posizione sulla chitarra e le indicazioni di barré.

```
\relative {
  \clef "treble_8"
  b,16 d g b e
  \textSpannerDown
  \override TextSpanner.bound-details.left.text = "XII "
  g16\startTextSpan
  b16 e g e b g\stopTextSpan
  e16 b g d
}
```




## Vedi anche
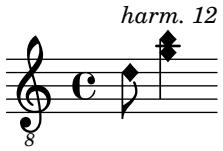
Guida alla notazione: [Estensori del testo], pagina 240.

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*, Sezione “Expressive marks” in *Frammenti di codice*.

## Indicazione di armonici e note smorzate

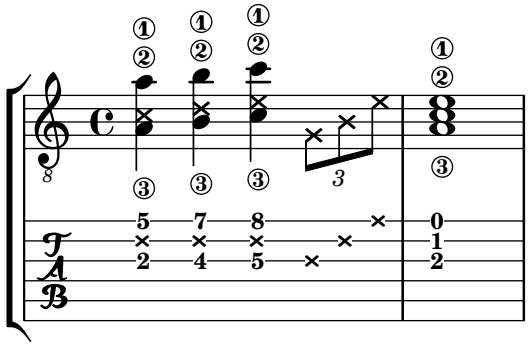
Si possono usare teste di nota speciali per indicare note smorzate o armonici. Gli armonici sono solitamente accompagnati da testo esplicativo.

```
\relative {
  \clef "treble_8"
  \override NoteHead.style = #'harmonic-mixed
  d'8^\markup { \italic \fontsize #-2 "harm. 12" } <g b>4
}
```



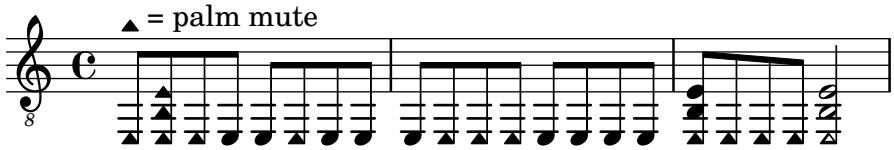
Le note smorzate (chiamate anche *dead note*) sono supportate sia sul rigo normale che sull'intavolatura:

```
music = \relative {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \tuplet 3/2 { g8 b e }
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
  \new TabStaff {
    \music
  }
>>
```



Un'altra tecnica esecutiva (usata specialmente nella chitarra elettrica) è il *palm mute*, in cui la corda viene parzialmente silenziata dal palmo della mano che la colpisce (da qui il nome). LilyPond supporta la notazione delle note in stile palm mute cambiando la testa della nota in una forma triangolare.

```
\new Voice { % Attenzione: la creazione esplicita di una Voice è
             %    necessaria per far sì che palmMuteOff funzioni correttamente
             %    quando palmMuteOn si trova all'inizio del brano.
  \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
  }
}
```




## Vedi anche

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

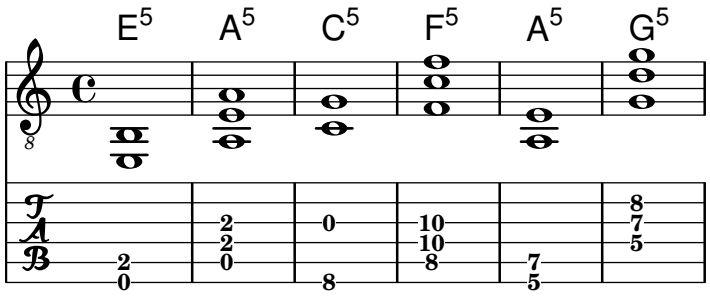
Guida alla notazione: [Teste di nota speciali], pagina 39, Sezione A.9 [Stili delle teste di nota], pagina 701.

## Indicazione di power chord

I power chord e i loro simboli possono essere inseriti in modalità accordo o come costrutti di accordo:

```
ChordsAndSymbols = {
  \chordmode {

    e,,1:5
    a,,1:5.8
    \set minimumFret = #8
    c,1:5
    f,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
    }
  >>
}
```

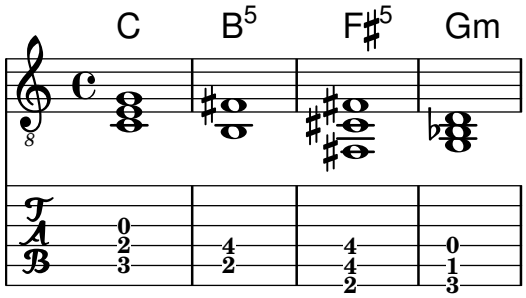



I simboli del power chord vengono automaticamente disabilitati appena si usa uno degli altri comuni modificatori degli accordi:

```
mixedChords = \chordmode {
```

```
  c,1

  b,,1:5
  fis,,1:5.8
  g,,1:m
}
\score {
  <<
    \new ChordNames {
      \mixedChords
    }
    \new Staff {
      \clef "treble_8"
      \mixedChords
    }
    \new TabStaff {
      \mixedChords
    }
  >>
}
```




## Vedi anche

Glossario musicale: Sezione “power chord” in *Glossario Musicale*.

Guida alla notazione: [Accordi estesi e alterati], pagina 422, [Stampa dei nomi degli accordi], pagina 424.

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

### 2.4.3 Banjo

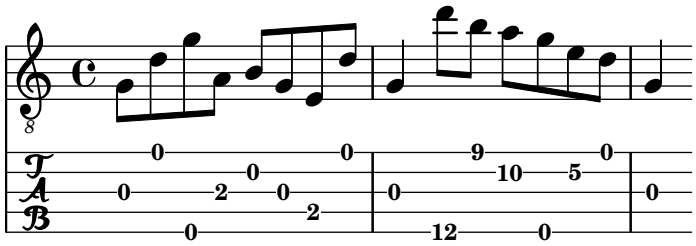
## Intavolature per banjo

LilyPond ha un supporto basilare per il banjo a cinque corde. Quando si scrivono intavolature per banjo a cinque corde, usare la funzione di formattazione dell’intavolatura per banjo per ottenere i numeri di tasto corretti per la quinta corda:

```
music = {
  g8 d' g'\5 a b g e d' |
  g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
  g4
}

<<
  \new Staff \with { \omit StringNumber }
```

```
  { \clef "treble_8"  \music }
  \new TabStaff \with {
    tablatureFormat = #fret-number-tablature-format-banjo
    stringTunings = #banjo-open-g-tuning
  }
  { \music }
>>
```



Sono disponibili varie accordature comuni per banjo a cinque corde: `banjo-c-tuning` (gCGBD), `banjo-modal-tuning` (gDGCD), `banjo-open-d-tuning` (aDF#AD) e `banjo-open-dm-tuning` (aDFAD).

Queste possono essere convertite in accordature a quattro corde usando la funzione `four-string-banjo`:

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

## Vedi anche

File installati: `ly/string-tunings-init.ly`.

Frammenti: Sezione “Fretted strings” in *Frammenti di codice*.

### 2.4.4 Liuto

## Intavolature per liuto

LilyPond supporta l’intavolatura per liuto.

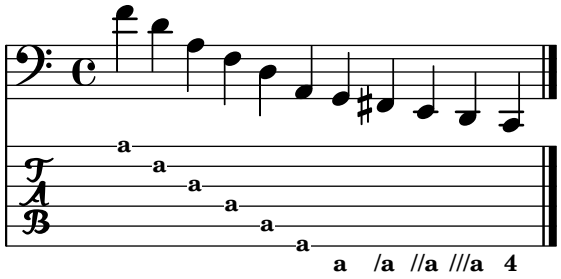
Per aggiungere altre corde basse usare `additionalBassStrings`, dove si impostano le altezze di queste corde. Appariranno sotto la linea più bassa come: a, /a, //a, ///a, 4, 5, etc.

Occorre assegnare `fret-letter-tablature-format` a `tablatureFormat` e probabilmente usare `fretLabels` per ulteriori personalizzazioni.

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  <<
    \new Staff { \clef bass \cadenzaOn  \m }
    \new TabStaff \m
  >>
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
```

```
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```




### Problemi noti e avvertimenti

L'uso di `FretBoards` insieme a `additionalBassStrings` non è supportato e produrrà risultati insoddisfacenti.

## 2.5 Percussioni

### 2.5.1 Notazione comune per le percussioni

La musica ritmica viene usata principalmente per la notazione di strumenti percussivi, ma può essere usata anche per mostrare i ritmi delle melodie.

### Riferimenti per percussioni

- Alcune percussioni possono essere scritte su un rigo ritmico; questo argomento è trattato in [Mostrare i ritmi della melodia], pagina 83, e [Istanziare nuovi righi], pagina 194.
- l'output MIDI è trattato in un'altra sezione, vedi Sezione 3.5 [Creazione dell'output MIDI], pagina 525.

### Vedi anche

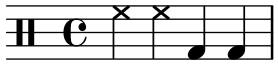
Guida alla notazione: [Mostrare i ritmi della melodia], pagina 83, [Istanziare nuovi righi], pagina 194. Sezione 3.5 [Creazione dell'output MIDI], pagina 525.

Frammenti: Sezione "Percussion" in *Frammenti di codice*.

### Notazione di base per percussioni

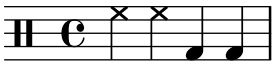
Le note percussive possono essere inserite in modalità `\drummode`, che è simile alla modalità standard di inserimento delle note. Il modo più semplice per inserire note percussive è il comando `\drums`, che crea il contesto e la modalità di inserimento corretti per le percussioni:

```
\drums {
  hihat4 hh bassdrum bd
}
```



Si tratta di una scorciatoia per:
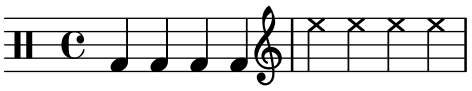
```
\new DrumStaff \drummode {
  hihat4 hh bassdrum bd
}
```



Ogni elemento di uno strumento a percussione ha un nome completo e un nome abbreviato, entrambi utilizzabili nei file di input. L'elenco completo dei nomi delle note percussive si trova in Sezione A.15 [Note percussive], pagina 766.

Si noti che l'uso della normale notazione delle altezze (come `cis4`) in un contesto `DrumStaff` produrrà un messaggio di errore. Le chiavi delle percussioni sono aggiunte automaticamente a un contesto `DrumStaff` ma possono anche essere impostate esplicitamente. È possibile usare anche altre chiavi.

```
\drums {
  \clef percussion
  bd4 4 4 4
  \clef treble
  hh4 4 4 4
}
```



Ci sono alcuni problemi relativi al supporto del MIDI per gli strumenti percussivi; maggiori dettagli in Sezione 3.5 [Creazione dell'output MIDI], pagina 525.

## Vedi anche

Guida alla notazione: Sezione 3.5 [Creazione dell'output MIDI], pagina 525, Sezione A.15 [Note percussive], pagina 766.
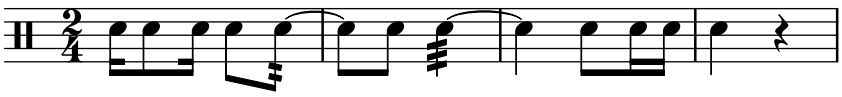
File installati: `ly/drumpitch-init.ly`.

Frammenti: Sezione "Percussion" in *Frammenti di codice*.

## Rulli di tamburo

I rulli di tamburo si indicano con tre slash sul gambo. Le note di un quarto o più lunghe vengono mostrate con tre slash, quelle da un ottavo con due slash (la travatura rappresenta la terza), mentre le note più brevi di un'ottavo hanno uno slash sul gambo per integrare le travature. Ciò si ottiene con la notazione per tremolo, come è descritto in [Ripetizioni con tremolo], pagina 167.

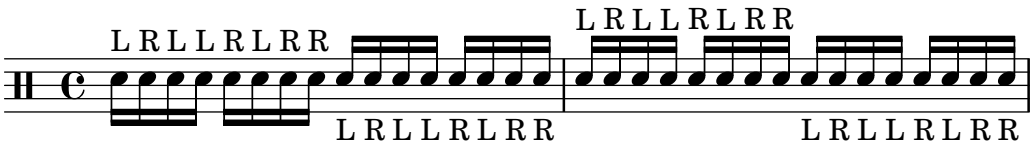
```
\drums {
  \time 2/4
  sn16 8 16 8 8:32 ~
  8 8 4:32 ~
  4 8 16 16
  4 r4
}
```



La scelta di quale bacchetta usare può essere indicata tramite il testo `"R"` (right, destra) o `"L"` (left, sinistra) da porre sopra o sotto le note, come spiegato in Sezione 5.4.2 [Direzione e posizionamento], pagina 630. Conviene modificare la proprietà `staff-padding` per ottenere un allineamento soddisfacente.

```
\drums {
  \repeat unfold 2 {
```

```
    sn16^"L" 16^"R" 16^"L" 16^"L" 16^"R" 16^"L" 16^"R" 16^"R"
    \stemUp
    sn16_"L" 16_"R" 16_"L" 16_"L" 16_"R" 16_"L" 16_"R" 16_"R"
  }
}
```




## Vedi anche

Guida alla notazione: [Ripetizioni con tremolo], pagina 167.

Frammenti: Sezione “Percussion” in *Frammenti di codice*.

## Percussioni a suono determinato

Alcuni strumenti percussivi a suono determinato (per esempio xilofono, vibrafono e timpani) si scrivono usando i righi normali, come è spiegato in altre sezioni del manuale.

## Vedi anche

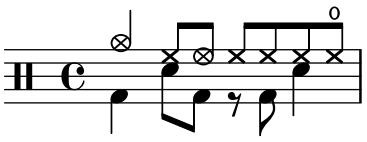
Guida alla notazione: Sezione 3.5 [Creazione dell’output MIDI], pagina 525.

Frammenti: Sezione “Percussion” in *Frammenti di codice*.

## Righi delle percussioni

Una parte percussiva per più di uno strumento di norma usa un pentagramma multilinea dove ogni posizione sul rigo si riferisce a una parte delle percussioni. Per scrivere questo tipo di musica, le note devono essere interpretate nei contesti `DrumStaff` e `DrumVoice`.

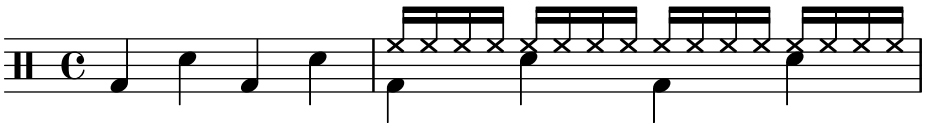
```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



L’esempio precedente mostra la notazione polifonica in forma estesa. Ma si può usare anche quella in forma breve, descritta in Sezione “Sento le Voci” in *Manuale di Apprendimento*. Per esempio,

```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
```

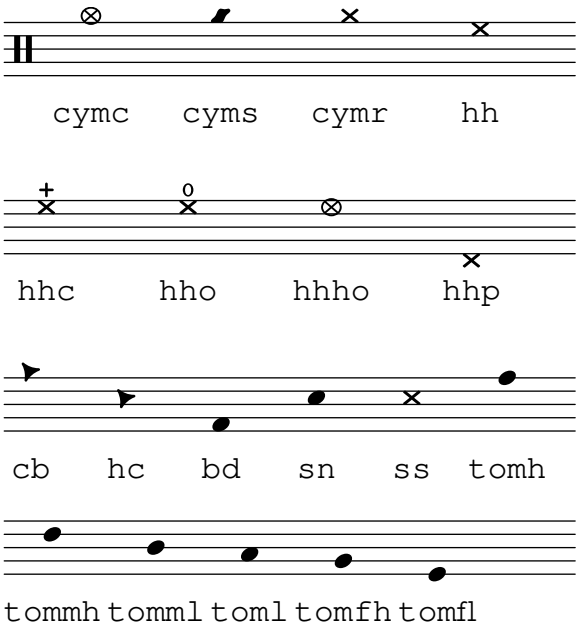
```
        \repeat unfold 16 hh16
      } \\ {
        bd4 sn4 bd4 sn4
      } >>
    }
>>
```



Esistono anche altre opzioni di formattazione, che si possono impostare tramite la proprietà `drumStyleTable` del contesto `DrumVoice`. Sono state definite le seguenti variabili:

`drums-style`

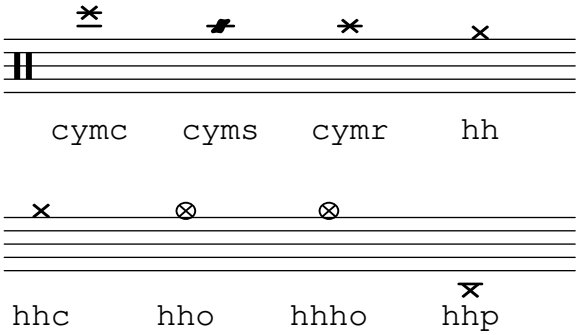
Questo è lo stile predefinito. Crea uno spartito per una tipica batteria su un rigo di cinque linee:

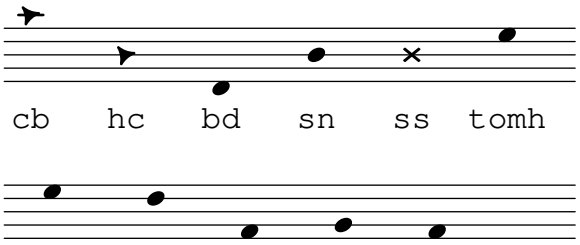



Lo schema percussivo supporta sei diversi tom. Quando si usa un numero inferiore di tom, selezionare i tom che producono il risultato desiderato. Per esempio, per avere i tom sulle tre linee centrali si usa `tommh`, `tomml` e `tomfh`.

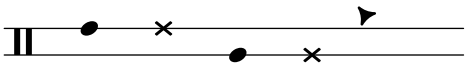
`agostini-drums-style`

Inventata dal percussionista francese Dante Agostini nel 1965, questa notazione è comunemente utilizzata in Francia ma anche altrove.

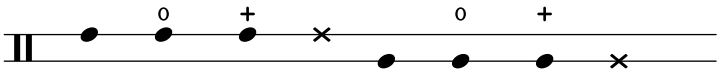




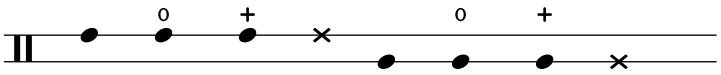

`timbales-style`
: Questo stile rappresenta i timbales su un rigo di due linee:




`congas-style`
: Questo stile rappresenta le conga su un rigo di due linee:




`bongos-style`
: Questo stile rappresenta i bongo su un rigo di due linee:




`percussion-style`
: Per rappresentare qualsiasi tipo di percussione semplice su righi di una linea:




Si possono anche definite stili percussivi personalizzati, come è spiegato in [Righi delle percussioni personalizzati], pagina 402.

## Vedi anche

Manuale di apprendimento: Sezione “Sento le Voci” in *Manuale di Apprendimento*.

Guida alla notazione: [Righi delle percussioni personalizzati], pagina 402.

File installati: `ly/drumpitch-init.ly`.

Frammenti: Sezione “Percussion” in *Frammenti di codice*.

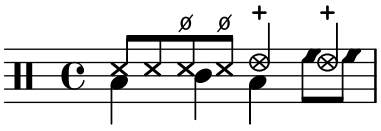
## Righi delle percussioni personalizzati

È possibile definire stili percussivi personalizzati, su cui poi impostare la proprietà `drumStyleTable`. Le notazioni esistenti possono essere ridefinite come una lista associativa in cui ogni voce deve essere costituita da quattro elementi: un nome, lo stile della testa di nota (o `default`), un segno di articolazione se necessario (o `#f` se non lo è), e la posizione della testa

di nota sul rigo. Questa lista deve poi essere convertita in una tabella hash Scheme, usando la funzione `alist->hash-table`.

```
#(define mydrums '(
         (bassdrum        default   #f           -1)
         (snare           default   #f           0)
         (hihat           cross     #f           1)
         (halfopenhihat   cross     "halfopen"   1)
         (pedalhihat      xcircle   "stopped"    2)
         (lowtom          diamond   #f           3)))
up = \drummode { hh8 hh hhho hhho hhp4 hhp }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```

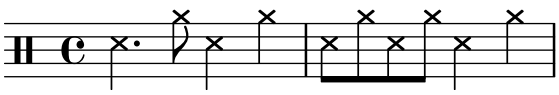


Si possono aggiungere anche nuovi nomi a queste notazioni personalizzate attraverso la variabile `drumPitchNames`, che può essere ridefinita come una lista associativa (o aumentata `append`endo una nuova lista al suo valore esistente, come illustrato sotto), ma anche attraverso le sue voci individuali. Ciò permette anche di definire alias: scorciatoie di input alternativo per alcune notazioni.

```
drumPitchNames =
  #(append
    '((leftsnap . sidestick)
      (rightsnap . ridecymbal))
    drumPitchNames)

drumPitchNames.ls = #'sidestick
drumPitchNames.rs = #'ridecymbal

\drums {
  leftsnap4. rightsnap8 leftsnap4 rightsnap
  ls8 rs ls rs ls4 rs
}
```



In modo analogo, la proprietà `drumPitchTable` associa una certa altezza (ovvero un diverso suono di strumento, così come è fornito dai soundfont MIDI disponibili) a ciascuna notazione. Questa proprietà deve essere definita come una tabella di hash, che di nuovo viene convertita da una lista associativa (salvata per impostazione predefinita come variabile `midiDrumPitches`). La ridefinizione di queste associazioni si ottiene, come spiegato sopra, o definendo un'intera lista associativa oppure attraverso voci individuali. L'esempio seguente mostra come creare un

insieme di notazione completo con la propria sintassi di input, notazioni personalizzate e output MIDI corrispondente.
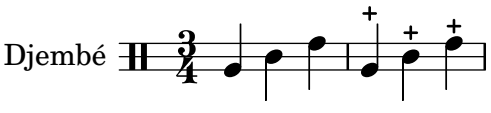
```
drumPitchNames.dbass      = #'dbass
drumPitchNames.dba        = #'dbass  % 'db è già in uso
drumPitchNames.dbassmute  = #'dbassmute
drumPitchNames.dbm        = #'dbassmute
drumPitchNames.do         = #'dopen
drumPitchNames.dopenmute  = #'dopenmute
drumPitchNames.dom        = #'dopenmute
drumPitchNames.dslap      = #'dslap
drumPitchNames.ds         = #'dslap
drumPitchNames.dslapmute  = #'dslapmute
drumPitchNames.dsm        = #'dslapmute

#(define djembe-style
  '((dbass      default  #f         -2)
    (dbassmute  default  "stopped"  -2)
    (dopen      default  #f          0)
    (dopenmute  default  "stopped"   0)
    (dslap      default  #f          2)
    (dslapmute  default  "stopped"   2)))

midiDrumPitches.dbass     = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen     = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap     = b
midiDrumPitches.dslapmute = ais

test = \drummode { dba4 do ds dbm dom dsm }

\score {
  \new DrumStaff \with {
    \override StaffSymbol.line-count = #3
    instrumentName = "Djembé "
    drumStyleTable = #(alist->hash-table djembe-style)
    drumPitchTable = #(alist->hash-table midiDrumPitches)
  } {
    \time 3/4
    \test
  }
  \layout {}
  \midi {}
}
```




## Vedi anche

File installati: `ly/drumpitch-init.ly`.

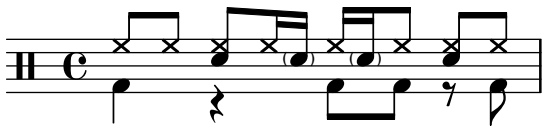
Frammenti: Sezione “Percussion” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “DrumVoice” in *Guida al Funzionamento Interno*.

### Ghost note

Le ghost note (note fantasma), note anche come note morte, mute o non suonate, possono essere create col comando `\parenthesize`, spiegato in [Parentesi], pagina 231.

```
\new DrumStaff
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ 8] <hh sn> hh16
        \parenthesize sn  hh
        \parenthesize sn  hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 8 r8 bd
      }
    >>
  }
>>
```

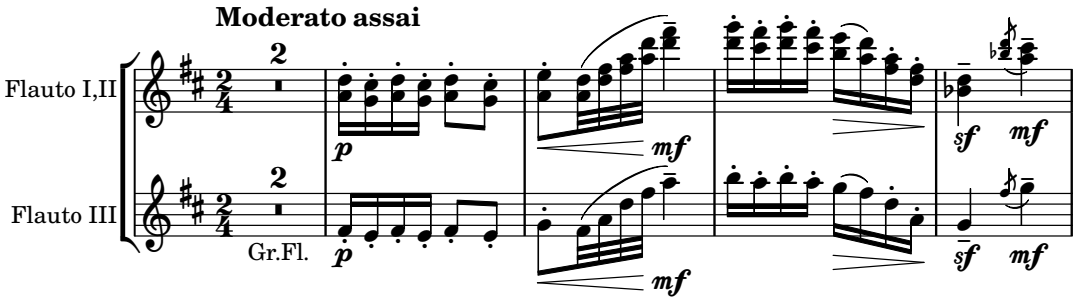


### Vedi anche

Guida alla Notazione: [Parentesi], pagina 231.

Frammenti: Sezione “Percussion” in *Frammenti di codice*.

## 2.6 Strumenti aerofoni




Questa sezione tratta gli elementi della notazione musicale specifici degli strumenti aerofoni (o strumenti a fiato).

### 2.6.1 Notazione comune per gli strumenti aerofoni

Questa sezione tratta la notazione comune alla maggior parte degli strumenti aerofoni.

## Riferimenti per strumenti aerofoni

Molte delle questioni di notazione degli strumenti aerofoni (o strumenti a fiato) riguardano la respirazione e l'uso della lingua:

- La respirazione può essere indicata con pause o [Respiri], pagina 139.
- Un'esecuzione in legato è indicata con [Legature di portamento], pagina 135.
- Diversi tipi di uso della lingua, dal legato al non legato allo staccato, sono rappresentati solitamente da segni di articolazione, talvolta insieme a legature di portamento, vedi [Articolazioni e abbellimenti], pagina 124, e Sezione A.14 [Elenco delle articolazioni], pagina 764.
- Il frullato si indica solitamente ponendo un segno di tremolo e del testo vicino alla nota. Vedi [Ripetizioni con tremolo], pagina 167.

Altri aspetti della notazione musicale che si possono applicare agli strumenti aerofoni:

- Molti aerofoni sono strumenti traspositori, vedi [Trasporto strumentale], pagina 27.
- I glissati sono caratteristici del trombone, ma altri aerofoni sono in grado di eseguire dei glissati agendo sulle chiavi o sui pistoni. Vedi [Glissando], pagina 142.
- I glissati ottenuti con una serie di armonici, possibili su tutti gli ottoni ma comuni nel corno francese, vengono scritti solitamente come [Abbellimenti], pagina 116.
- Le intonazioni dell'altezza al termine di una nota sono trattate in [Portamenti indeterminati discendenti (cadute) e ascendenti], pagina 141.
- Le chiavi o i pistoni sbattuti si mostrano spesso con lo stile `cross` (barrato) delle [Teste di nota speciali], pagina 39.
- I legni possono produrre armonici soffiando forte sulle note basse. Gli armonici sono illustrati dall'articolazione `flageolet`. Vedi Sezione A.14 [Elenco delle articolazioni], pagina 764.
- L'uso di sordine per gli ottoni viene indicato solitamente con del testo, ma se ci sono molti cambi rapidi è meglio usare le articolazioni `stopped` e `open`. Vedi [Articolazioni e abbellimenti], pagina 124, e Sezione A.14 [Elenco delle articolazioni], pagina 764.
- I corni "smorzati" si indicano con l'articolazione `stopped`. Vedi [Articolazioni e abbellimenti], pagina 124.

## Frammenti di codice selezionati

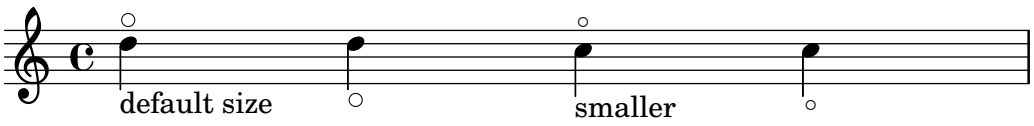
*Modifica della dimensione del segno di \flageolet*

Per ridurre la dimensione del cerchio di `\flageolet`, usare la seguente funzione Scheme.

```
smallFlageolet = \tweak font-size -3 \flageolet

\layout { ragged-right = ##f }

\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



## Vedi anche

Guida alla notazione: [Respiri], pagina 139, [Legature di portamento], pagina 135, [Articolazioni e abbellimenti], pagina 124, Sezione A.14 [Elenco delle articolazioni], pagina 764, [Ripetizioni con tremolo], pagina 167, [Trasporto strumentale], pagina 27, [Glissando], pagina 142, [Abbellimenti], pagina 116, [Portamenti indeterminati discendenti (cadute) e ascendenti], pagina 141, [Teste di nota speciali], pagina 39,

Frammenti: Sezione “Winds” in *Frammenti di codice*.

## Diteggiature

Tutti gli aerofoni eccetto il trombone richiedono l’uso di varie dita per produrre ciascuna altezza. Alcuni esempi di diteggiatura sono illustrati nei prossimi frammenti.

Si possono creare diagrammi per i legni, descritti in Sezione 2.6.3.1 [Diagrammi per legni], pagina 411.

## Frammenti di codice selezionati

*Simboli di diteggiatura per strumenti aerofoni*

Si possono ottenere simboli speciali combinando glifi esistenti; ciò è utile per gli strumenti aerofoni.

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
  (+ (ly:self-alignment-interface::centered-on-x-parent g)
     (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \relative c'{
    g\open
    \once \override TextScript.staff-padding = #-1.0
    \centermarkup
    g^\markup {
      \combine
        \musicglyph "scripts.open"
        \musicglyph "scripts.tenuto"
    }
    \centermarkup
    g^\markup {
      \combine
        \musicglyph "scripts.open"
        \musicglyph "scripts.stopped"
    }
    g\stopped
  }
}
```

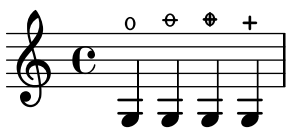


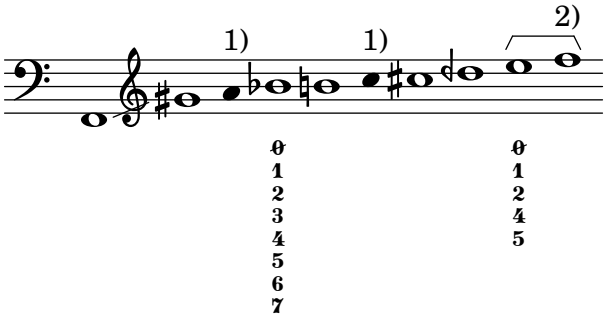
*Grafico della diteggiatura per flauto dolce*

L'esempio seguente illustra come realizzare grafici delle diteggiature per strumenti aerofoni.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset = #(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
       (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \omit Stem
    \omit Flag
    \consists "Horizontal_bracket_engraver"
  }
  {
    \clef bass
    \set Score.timing = ##f
    f,1*1/4 \glissando
    \clef violin
    gis'1*1/4
    \stemDown a'4^\markup {1)}
    \centermarkup
    \once \override TextScript.padding = #2
    bes'1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 3 \finger 4 \finger 5 \finger 6 \finger 7} }
    b'1*1/4
    c''4^\markup {1)}
    \centermarkup
    \once \override TextScript.padding = #2
    cis''1*1/4
    deh''1*1/4
    \centermarkup
    \once \override TextScript.padding = #2
    \once \override Staff.HorizontalBracket.direction = #UP
    e''1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 4 \finger 5} }\startGroup
    f''1*1/4^\markup {2)}\stopGroup
  }
}
```




## Vedi anche

Guida alla notazione: Sezione 2.6.3.1 [Diagrammi per legni], pagina 411.

Frammenti: Sezione “Winds” in *Frammenti di codice*.

## 2.6.2 Cornamusa

Questa sezione tratta la notazione comune per la cornamusa.

## Definizioni per cornamusa

LilyPond contiene delle definizioni speciali per la musica per cornamusa scozzese; per usarle, aggiungere
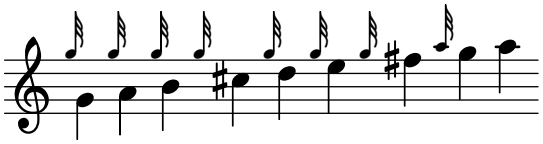
```
\include "bagpipe.ly"
```

in cima al file di input. È così possibile aggiungere con comandi brevi gli speciali abbellimenti tipici della musica per cornamusa. Per esempio, si può scrivere `\taor` invece di

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` contiene anche le definizioni delle altezze delle note della cornamusa nelle ottave appropriate, in modo da non doversi preoccupare di `\relative` o `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



In teoria, la musica per cornamusa usa la tonalità di Re maggiore (anche se ciò non è del tutto vero). Tuttavia, essendo l’unica tonalità che si può usare, normalmente non viene scritta esplicitamente. Dunque per nascondere la tonalità si può usare il comando `\hideKeySignature` prima che inizi la musica. Se per qualche ragione si desidera mostrare la tonalità, usare invece il comando `\showKeySignature`.

Alcune musiche moderne usano una diteggiatura barrata su Do e Fa per rendere bemolli queste note. Ciò si può indicare con `c-flat` o `f-flat`. Analogamente, il Sol alto del “piobaireachd” può essere scritto come `g-flat` quando si tratta di musica leggera.

## Vedi anche

Frammenti: Sezione “Winds” in *Frammenti di codice*.

## Esempio per cornamusa

Ecco come appare la famosa melodia “Amazing Grace” in notazione per cornamusa.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
```
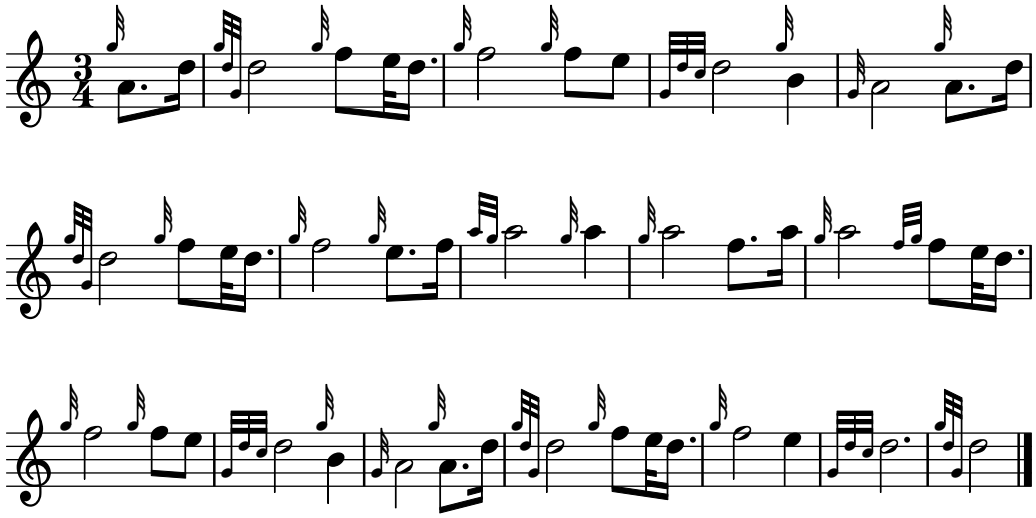
```
  \context { \Score \remove "Bar_number_engraver" }
}

\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 "|."
}
```

# Amazing Grace

Hymn Trad. arr.



### Vedi anche

Frammenti: Sezione “Winds” in *Frammenti di codice*.

## 2.6.3 Legni

Questa sezione tratta la notazione specifica per i legni.

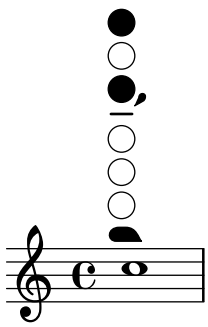
### 2.6.3.1 Diagrammi per legni

I diagrammi dei legni permettono di indicare la diteggiatura da usare per certe note e sono disponibili per i seguenti strumenti:

- ottavino
- flauto
- oboe
- clarinetto
- clarinetto basso
- sassofono
- fagotto
- controfagotto

I diagrammi dei legni vengono creati come testi di tipo markup:

```
c''1^\markup {
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
}
```



Le chiavi possono essere aperte, semichiuse, ad anello o chiuse completamente:

```
\textLengthOn
c''1^\markup {
  \center-column {
    "un quarto"
    \woodwind-diagram #'flute #'((cc . (one1q))
                                 (lh . ())
                                 (rh . ()))
  }
}

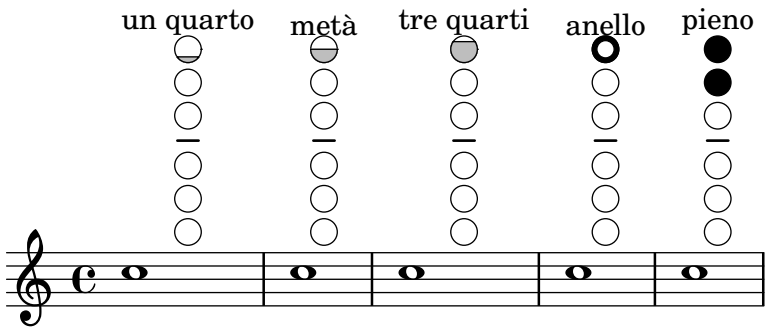
c''1^\markup {
  \center-column {
    "metà"
    \woodwind-diagram #'flute #'((cc . (one1h))
```

```
                                  (lh . ())
                                  (rh . ()))
  }
}

c''1^\markup {
  \center-column {
    "tre quarti"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                  (lh . ())
                                  (rh . ()))
  }
}

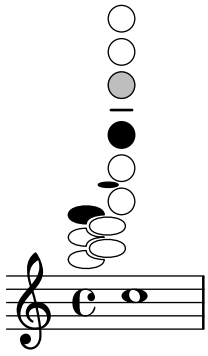
c''1^\markup {
  \center-column {
    "anello"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                  (lh . ())
                                  (rh . ()))
  }
}

c''1^\markup {
  \center-column {
    "pieno"
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                  (lh . ())
                                  (rh . ()))
  }
}
```




I trilli si indicano con delle chiavi sovrapposte:

```
c''1^\markup {
  \woodwind-diagram #'bass-clarinet
                    #'((cc . (threeT four))
                       (lh . ())
                       (rh . (b fis)))
}
```



Possono essere visualizzati vari tipi di trilli:

```
\textLengthOn
c''1^\markup {
  \center-column {
    "one quarter to ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

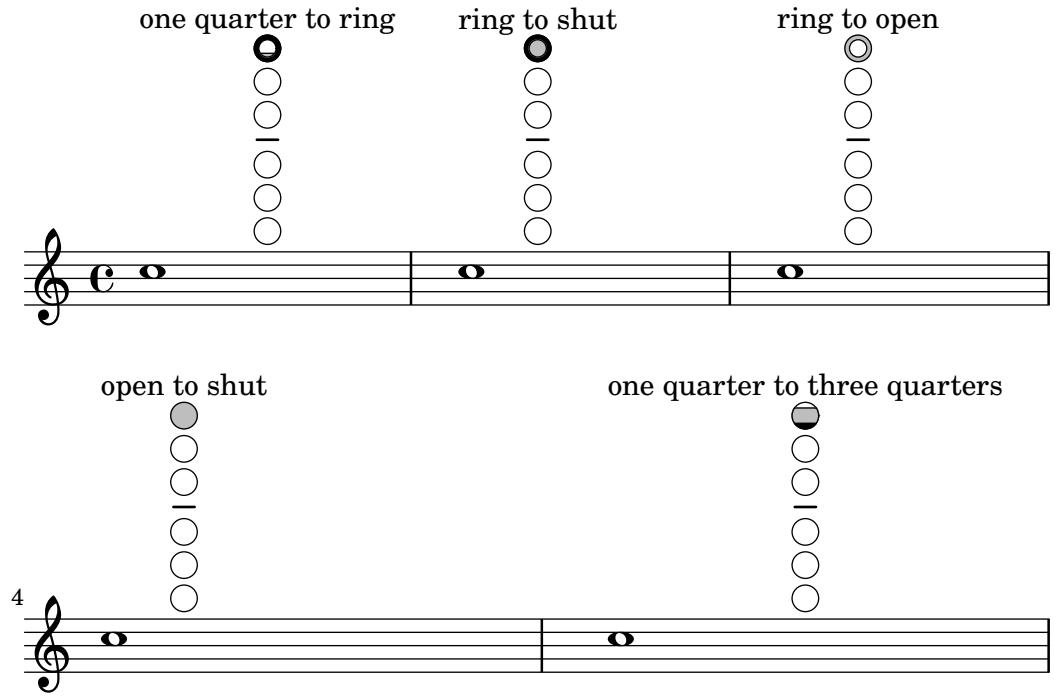
c''1^\markup {
  \center-column {
    "ring to shut"
    \woodwind-diagram #'flute #'((cc . (oneTR))
                                 (lh . ())
                                 (rh . ()))
  }
}

c''1^\markup {
  \center-column {
    "ring to open"
    \woodwind-diagram #'flute #'((cc . (oneRT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c''1^\markup {
  \center-column {
    "open to shut"
    \woodwind-diagram #'flute #'((cc . (oneT))
                                 (lh . ())
                                 (rh . ()))
  }
}

c''1^\markup {
  \center-column {
    "one quarter to three quarters"
    \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                 (lh . ())
```

```
                                (rh . ()))
  }
}
```




È possibile visualizzare nella console un elenco di tutte le chiavi e impostazioni possibili usando `#(print-keys-verbose 'flute)` oppure visualizzarlo nel file di log usando `#(print-keys-verbose 'flute (current-error-port))`, ma non appariranno nell'output.

È possibile creare nuovi diagrammi, ma è richiesta una conoscenza di Scheme e dunque tale possibilità non è a disposizione di tutti gli utenti. I modelli dei diagrammi si trovano in `scm/define-woodwind-diagrams.scm` e `scm/display-woodwind-diagrams.scm`.

## Comandi predefiniti

## Frammenti di codice selezionati

*Elenco dei diagrammi per i legni*

L'esempio seguente mostra tutti i diagrammi per i legni attualmente definiti in LilyPond.

```
\layout {
  indent = 0
}

\relative c' {
  \textLengthOn
  c1^
  \markup {
    \center-column {
      'tin-whistle
      " "
       \woodwind-diagram
                  #'tin-whistle
                  #'()
    }
```

```
}

c1^
\markup {
  \center-column {
    'piccolo
    " "
     \woodwind-diagram
                #'piccolo
                #'()
  }
}

c1^
\markup {
  \center-column {
     'flute
     " "
     \woodwind-diagram
        #'flute
        #'()
  }
}
c1^\markup {
  \center-column {
    'oboe
    " "
    \woodwind-diagram
      #'oboe
      #'()
  }
}

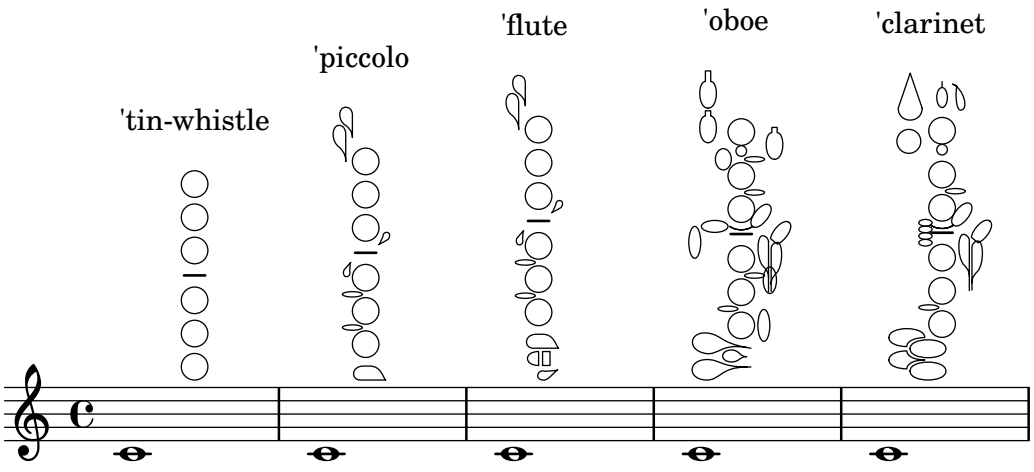
c1^\markup {
  \center-column {
    'clarinet
    " "
    \woodwind-diagram
      #'clarinet
      #'()
  }
}

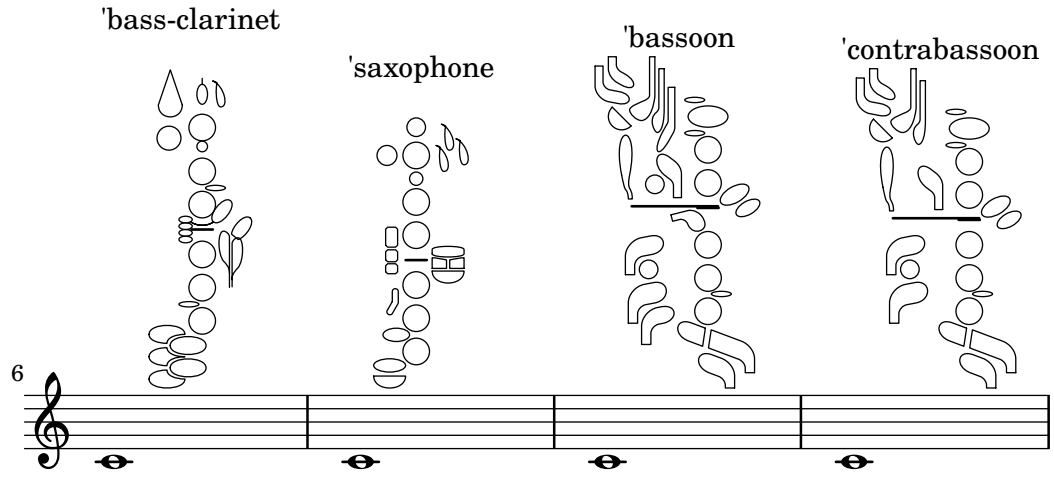
c1^\markup {
  \center-column {
    'bass-clarinet
    " "
    \woodwind-diagram
      #'bass-clarinet
      #'()
  }
}
```

```
  c1^\markup {
    \center-column {
      'saxophone
      " "
      \woodwind-diagram
        #'saxophone
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'bassoon
      " "
      \woodwind-diagram
        #'bassoon
        #'()
    }
  }

  c1^\markup {
    \center-column {
      'contrabassoon
      " "
      \woodwind-diagram
        #'contrabassoon
        #'()
    }
  }
}
```



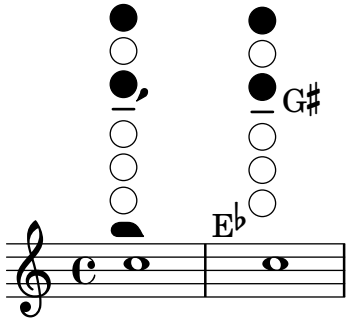



*Diagrammi grafici e testuali per i legni*

In molti casi, le chiavi diverse da quelle della colonna centrale possono essere visualizzate per nome oltre che in forma grafica.

```
\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)))
    }
}
```



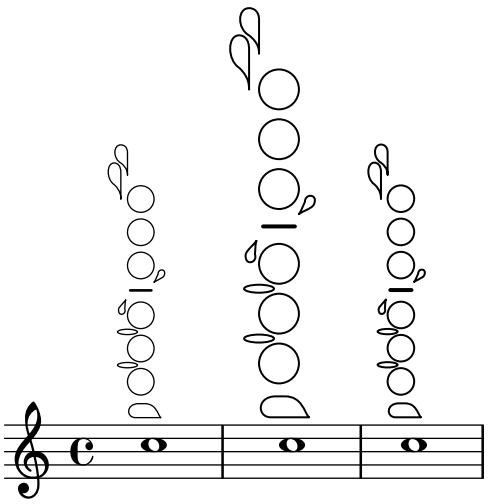

*Cambiare la dimensione dei diagrammi per legni*

La dimensione e lo spessore dei diagrammi per legni possono essere modificati.

```
\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
        #'()
    }
}
```
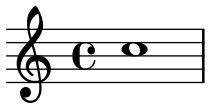


*Elenco delle chiavi dei diagrammi per legni*

Il seguente frammento produce un elenco di tutte le possibili chiavi e delle loro impostazioni per i diagrammi per legni, come sono definite in `scm/define-woodwind-diagrams.scm`. L'elenco sarà visualizzato nel file di log, ma non nello spartito. Se si desidera che l'output appaia nella console, omettere dai comandi `(current-error-port)`.

```
#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))
```

```
\score {c''1}
```



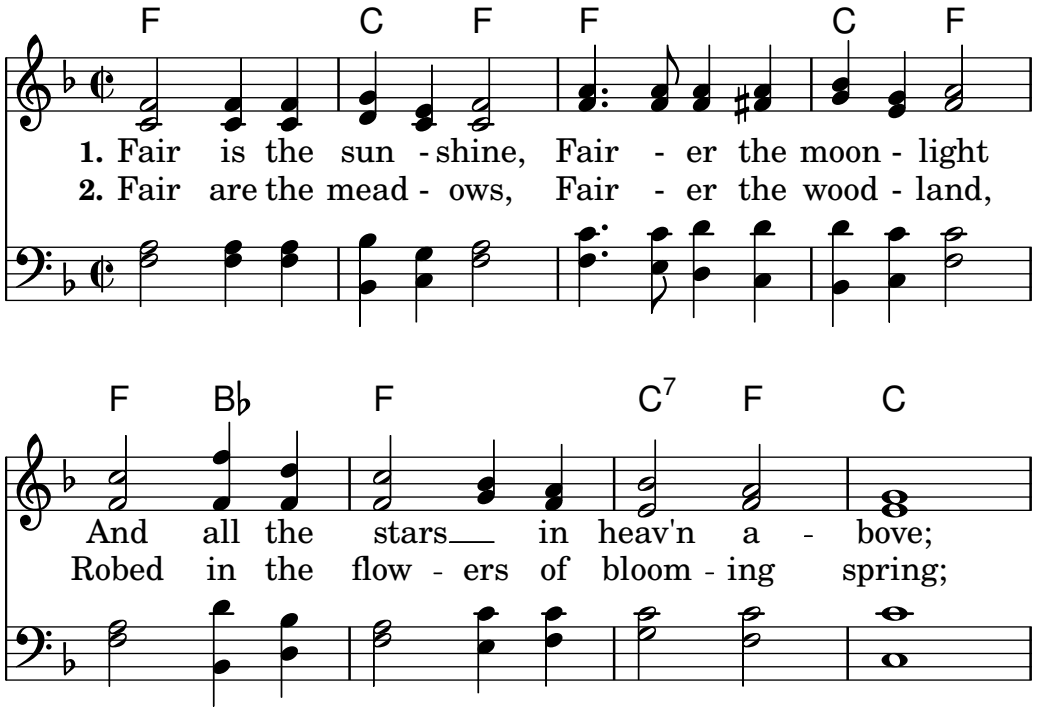
### Vedi anche

File installati: `scm/define-woodwind-diagrams.scm`, `scm/display-woodwind-diagrams.scm`.

Frammenti: Sezione “Winds” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*, Sezione “instrument-specific-markup-interface” in *Guida al Funzionamento Interno*.

## 2.7 Notazione per accordi




Gli accordi possono essere inseriti come normali note opppure nella modalità per accordi, e visualizzati usando varie convenzioni della tradizione europea. Possono essere visualizzati anche i nomi degli accordi e la notazione del basso continuo.

### 2.7.1 Modalità accordo

La modalità accordo serve a inserire accordi usando un indicatore della struttura dell’accordo invece delle altezze dell’accordo.
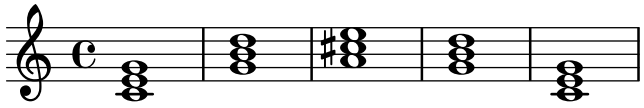
### Panoramica sulla modalità accordo

Gli accordi possono essere inseriti come musica simultanea, come è spiegato in [Note in un accordo], pagina 170.

Possono essere inseriti anche in “modalità accordo”, ovvero una modalità di input che utilizza le strutture degli accordi nella tradizione musicale europea invece di altezze specifiche. Ciò è più

comodo per chi è abituato a utilizzare i nomi degli accordi per descriverli. Maggiori informazioni sulle diverse modalità di input si trovano in Sezione 5.4.1 [Modalità di inserimento], pagina 629.

```
\chordmode { c1 g a g c }
```



Gli accordi inseriti in modalità accordo sono elementi musicali e possono quindi essere trasposti proprio come gli accordi inseriti come musica simultanea. `\chordmode` funziona in modalità assoluta, perché `\relative` non ha effetto sui blocchi `\chordmode`. Tuttavia in modalità accordo le altezze assolute sono un'ottava più alta che in modalità nota.

Nella musica sequenziale è possibile mischiare la modalità accordo con la modalità nota:

```
\relative {
  <c' e g>2 <g b d>
  \chordmode { c2 f }
  <c e g>2 <g' b d>
  \chordmode { f2 g }
}
```



## Vedi anche

Glossario musicale: Sezione "accordo" in *Glossario Musicale*.

Guida alla notazione: [Note in un accordo], pagina 170, Sezione 5.4.1 [Modalità di inserimento], pagina 629.

Frammenti: Sezione "Chords" in *Frammenti di codice*.
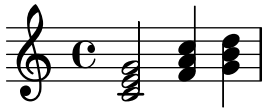
## Problemi noti e avvertimenti

Le scorciatoie predefinite per le articolazioni e gli ornamenti non possono essere usate sulle note in modalità accordo, vedi [Articolazioni e abbellimenti], pagina 124.

## Accordi comuni
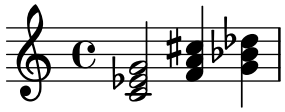
Le triadi maggiori si inseriscono scrivendo la nota fondamentale seguita da una durata opzionale:

```
\chordmode { c2 f4 g }
```



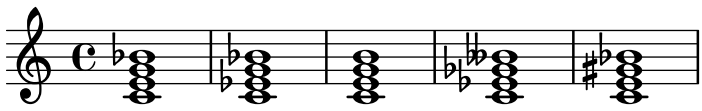
Le triadi minori, aumentate e diminuite si inseriscono aggiungendo `:` e un modificatore della qualità subito dopo la durata:

```
\chordmode { c2:m f4:aug g:dim }
```

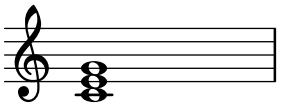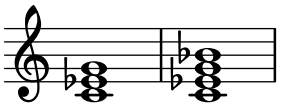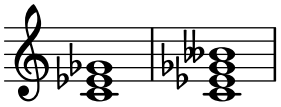
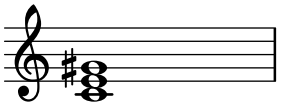
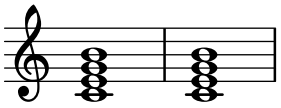


Si possono creare accordi di settima:

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



La tabella seguente mostra le azioni dei modificatori della qualità su triadi e accordi di settima. Il settimo grado predefinito aggiunto agli accordi è una settima minore, che rende la settima dominante l'accordo di settima fondamentale. Tutte le alterazioni sono relative alla settima dominante. Una tabella più completa dell'uso dei modificatori si trova in Sezione A.2 [Modificatori degli accordi comuni], pagina 664.

| **Modificatore** | **Azione** | **Esempio** |
|---|---|---|
| Nessuno | L'azione predefinita; produce una triade maggiore. |  |
| m, m7 | L'accordo minore. Questo modificatore abbassa il 3° grado. |  |
| dim, dim7 | L'accordo diminuito. Questo modificatore abbassa il 3°, il 5° e (se presente) il 7° grado. |  |
| aug | L'accordo aumentato. Questo modificatore alza il 5° grado. |  |
| maj, maj7 | L'accordo di settima maggiore. Questo modificatore aggiunge un 7° grado alzato. Il 7 che segue `maj` è opzionale. Non usare questo modificatore per creare una triade maggiore. |  |

## Vedi anche

Guida alla notazione: Sezione A.2 [Modificatori degli accordi comuni], pagina 664, [Accordi estesi e alterati], pagina 422.

Frammenti: Sezione "Chords" in *Frammenti di codice*.

## Problemi noti e avvertimenti

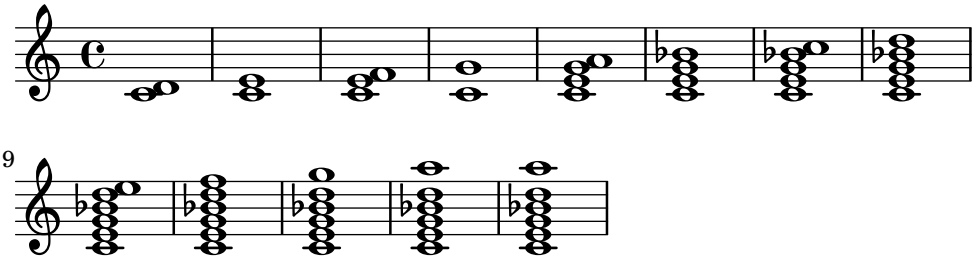
Si può usare un solo modificatore della qualità per accordo, solitamente sul grado più alto presente nell'accordo. Gli accordi con più di un modificatore verranno analizzati senza produrre

errori o avvisi, ma producono risultati imprevedibili. Gli accordi che non si possono ottenere con un singolo modificatore devono essere alterati per altezze individuali, come è spiegato in [Accordi estesi e alterati], pagina 422.

## Accordi estesi e alterati

In modalità accordo è possibile creare strutture di accordi di arbitraria complessità. Il modificatore permette di estendere un accordo, aggiungere o rimuovere i gradi di un accordo, alzare o abbassare i gradi, aggiungere una nota di basso o creare un'inversione. Il primo numero che segue il `:` identifica l'estensione dell'accordo. L'accordo viene costruito aggiungendo in modo sequenziale delle terze alla nota fondamentale finché non si raggiunge il numero specificato. Il settimo grado aggiunto come parte di un accordo esteso sarà la settima minore, non la settima maggiore. Se l'estensione non è una terza (per esempio, 6), vengono aggiunte le terze fino alla terza più alta entro l'estensione, e poi viene aggiunto il grado dell'estensione. Il valore più grande possibile è 13. Qualsiasi valore superiore viene interpretato come 13.

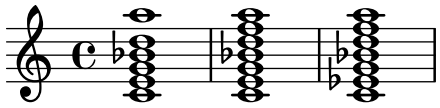
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



C'è un'eccezione speciale: `c:5` produce un "power chord" che consiste solo della nota fondamentale e della quinta.
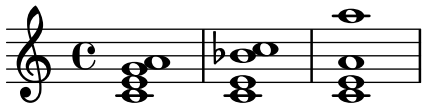
Dato che un 11° grado non alterato non suona bene se unito a un 13° grado non alterato, l'11° grado viene rimosso da un accordo `:13`, a meno che non venga aggiunto esplicitamente.

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



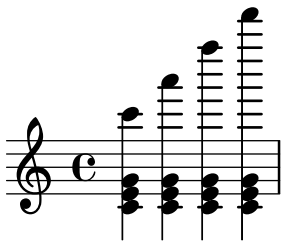
Si possono aggiungere gradi individuali a un accordo. Le aggiunte seguono l'estensione e sono precedute da un punto (`.`). Il settimo grado aggiunto a un accordo è una settima minore invece di una settima maggiore.

```
\chordmode {
  c1:3.5.6 c:3.7.8 c:3.6.13
}
```
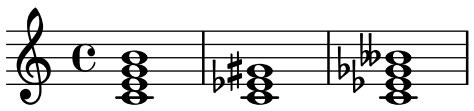


I gradi aggiunti possono essere alti quanto si vuole.

```
\chordmode {
  c4:3.5.15 c:3.5.20 c:3.5.25 c:3.5.30
}
```



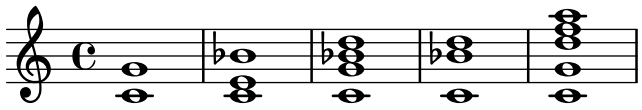
I gradi aggiunti dell'accordo possono essere alterati aggiungendo un suffisso `-` o `+` al numero. Per alterare un grado aggiunto automaticamente in quanto parte della struttura base dell'accordo, aggiungerlo come grado alterato.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



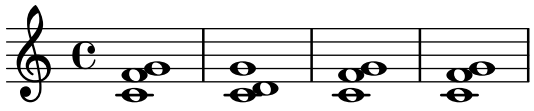
Dopo i gradi da aggiungere, una serie di gradi da togliere viene introdotta dal modificatore con prefisso `^`. Per togliere più di un grado, i gradi da rimuovere vengono separati dal `.` subito dopo il `^` iniziale.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
}
```



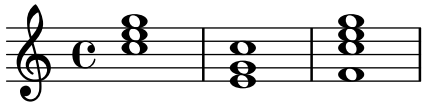
Per creare accordi sospesi si aggiunge alla stringa il modificatore `sus`, che toglie il terzo grado dall'accordo. Aggiungere `2` o `4` per aggiungere il 2° o il 4° grado dell'accordo. Se `sus` è seguito dal 2° o dal 4° grado, è equivalente a `^3`, altrimenti a `sus4`, ovvero `5.4`.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4
}
```



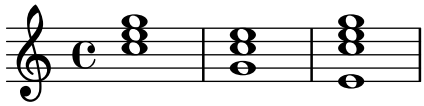
Le inversioni (ovvero l'inserimento di un'altezza diversa dalla fondamentale come nota più bassa dell'accordo) e le note di basso possono essere specificate aggiungendo */altezza* all'accordo.

```
\chordmode {
  c'1 c'/e c'/f
}
```



Una nota di basso che fa parte dell'accordo può essere aggiunta, invece di essere spostata come in un'inversione, usando /+*altezza*.

```
\chordmode {
  c'1 c'/g c'/+e
}
```



I modificatori degli accordi che possono essere usati per produrre i vari accordi standard sono elencati in Sezione A.2 [Modificatori degli accordi comuni], pagina 664.

### Vedi anche
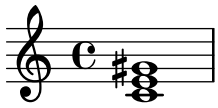
Guida alla notazione: Sezione A.2 [Modificatori degli accordi comuni], pagina 664.

Frammenti: Sezione "Chords" in *Frammenti di codice*.

### Problemi noti e avvertimenti

Ogni grado può essere presente in un accordo una volta sola. L'esempio seguente produce un accordo aumentato, perché `5+` viene interpretato per ultimo.

```
\chordmode { c1:3.5.5-.5+ }
```



## 2.7.2 Visualizzazione accordi

Gli accordi possono essere visualizzati per nome, oltre alla visualizzazione standard come note di un rigo.

### Stampa dei nomi degli accordi

Per visualizzare i nomi degli accordi si usa il contesto `ChordNames`:

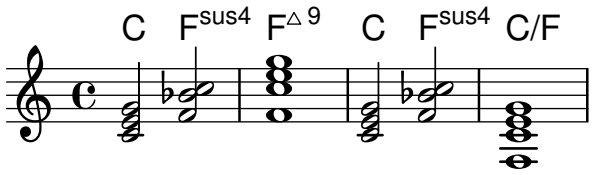
```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```

C F G

Gli accordi possono essere inseriti come note simultanee o tramite la modalità accordo. Il nome dell'accordo visualizzato sarà lo stesso, indipendentemente dalla modalità di inserimento, a meno che non ci siano inversioni o note di basso:
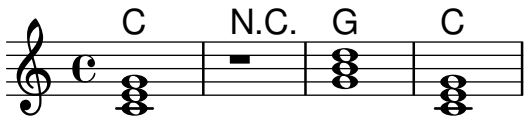
```
chordmusic = \relative {
  <c' e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
```

```
}
<<
  \new ChordNames {
    \chordmusic
  }
  {
    \chordmusic
  }
>>
```




Le pause usate nel contesto `ChordNames` faranno apparire il testo del simbolo `noChordSymbol` (nessun accordo).

```
<<
  \new ChordNames \chordmode {
    c1
    r1
    g1
    c1
  }
  \chordmode {
    c1
    r1
    g1
    c1
  }
>>
```




`\chords { ... }` è una scorciatoia per `\new ChordNames \chordmode { ... }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```

C Fm G△

```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```

C Fm G△

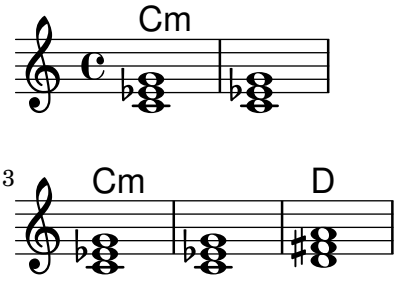
## Frammenti di codice selezionati

*Mostrare gli accordi nei cambi*

Per impostazione predefinita, ogni accordo inserito viene visualizzato; tale comportamento può essere modificato in modo che i nomi degli accordi siano mostrati solo all'inizio delle linee e quando l'accordo cambia.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}

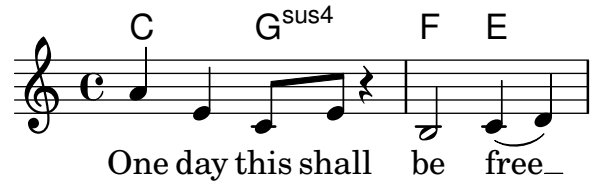
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
    \relative c' { \harmonies }
  }
>>
```




*Canzoniere semplice*

Mettendo insieme nomi degli accordi, melodia e testo si ottiene un canzoniere (in inglese "lead sheet"):

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```




## Vedi anche

Glossario musicale: Sezione "accordo" in *Glossario Musicale*.

Guida alla notazione: [Scrivere la musica in parallelo], pagina 191.

Frammenti: Sezione "Chords" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “ChordName” in *Guida al Funzionamento Interno*, Sezione “Chord_name_engraver” in *Guida al Funzionamento Interno*, Sezione “Volta_engraver” in *Guida al Funzionamento Interno*, Sezione “Bar_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

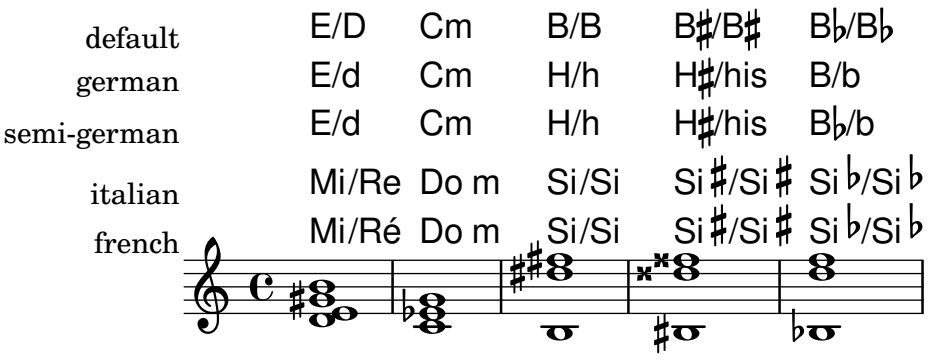
Gli accordi che contengono inversioni o note di basso alterate non avranno un nome corretto se inseriti usando la musica simultanea.

## Personalizzazione dei nomi degli accordi

Non esiste un unico sistema per il nome degli accordi. Tradizioni musicali diverse usano nomi diversi per lo stesso gruppo di accordi. Esistono anche simboli diversi per un certo nome di accordo. I nomi e i simboli degli accordi sono personalizzabili.

Il formato fondamentale dei nomi degli accordi è un sistema per la musica Jazz, proposto da Klaus Ignatzek (vedi Sezione “Bibliografia” in *Saggio*). (Si possono implementare altri sistemi per i nomi degli accordi attraverso funzioni Scheme, come illustrato nel frammento “Chord names alternative” in Sezione “Chords” in *Frammenti di codice*.) Un elenco dei comuni accordi jazz si trova nel Sezione A.1 [Grafico dei nomi degli accordi], pagina 663.

Il sistema predefinito per i nomi degli accordi può essere modificato facilmente in vari modi. Per iniziare, dei comandi predefiniti consentono di usare lingue diverse per l’altezza fondamentale. I comandi predefiniti sono `\germanChords`, `\semiGermanChords`, `\italianChords` e `\frenchChords`:




I canzonieri tedeschi talvolta indicano gli accordi minori con lettere minuscole, senza alcun suffisso *m*. Ciò si può ottenere impostando la proprietà `chordNameLowercaseMinor`:

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```




Il nome dell’accordo da visualizzare può essere aggiustato anche tramite le seguenti proprietà.

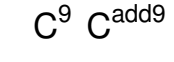
`chordRootNamer`

> Il nome dell’accordo appare solitamente come una lettera per la nota fondamentale seguita da un’alterazione opzionale. La trasformazione dell’altezza in lettera è eseguita da questa funzione. Si possono creare nomi speciali di note (per esempio, il tedesco “H” per l’accordo di Si) salvando una nuova funzione in questa proprietà.

`majorSevenSymbol`
: Questa proprietà contiene l'oggetto markup usato per seguire l'output di `chordRootNamer` e identificare un accordo di settima maggiore. Le opzioni predefinite sono `whiteTriangleMarkup` e `blackTriangleMarkup`.

`additionalPitchPrefix`
: Se il nome dell'accordo contiene ulteriori altezze, si può aggiungere un prefisso testuale. Il comportamento predefinito è senza prefisso, per evitare troppo ingombro, ma per numeri piccoli di altezze aggiuntive il risultato può essere visualmente efficace.

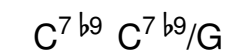
```
\new ChordNames {
  <c e g d'>    % add9
  \set additionalPitchPrefix = "add"
  <c e g d'>    % add9
}
```

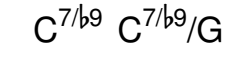


`chordNoteNamer`
: Se il nome dell'accordo contiene altre altezze oltre all'altezza fondamentale (per esempio, una nota di basso), questa funzione viene usata per mostrare l'altezza ulteriore. Per impostazione predefinita l'altezza viene stampata con la funzione `chordRootNamer`. La proprietà `chordNoteNamer` può essere impostata su una funzione specializzata per cambiare tale comportamento. Per esempio, la nota di basso può essere stampata in minuscolo.

`chordNameSeparator`
: Parti diverse del nome di un accordo sono separate da un piccolo spazio orizzontale. Impostando `chordNameSeparator`, si può usare qualsiasi testo come separatore. Ciò non ha effetti sul separatore tra un accordo e la sua nota di basso; per personalizzarlo si usa una proprietà specifica, `slashChordSeparator`.

```
\chords {
  c4:7.9- c:7.9-/g
  \set chordNameSeparator = \markup { "/" }
  \break
  c4:7.9- c:7.9-/g
}
```

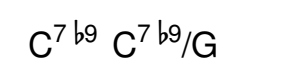


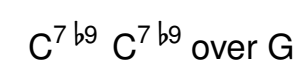


`slashChordSeparator`
: Gli accordi possono essere suonati utilizzando come nota di basso una nota diversa da quella fondamentale normalmente usata. In questo caso si parla di «inversioni» o «slash chord», perché il modo predefinito di rappresentarli è con una barra obliqua (forward slash) tra l'accordo principale e la nota di basso. Dunque il valore predefinito di `slashChordSeparator` è una barra obliqua, ma è possibile cambiarlo con qualsiasi testo.

```
\chords {
  c4:7.9- c:7.9-/g
  \set slashChordSeparator = \markup { " over " }
```

```
  \break
  c4:7.9- c:7.9-/g
}
```

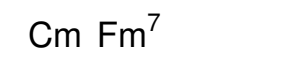




`chordNameExceptions`
: Questa proprietà è una lista di coppie. Il primo elemento di ciascuna coppia è un insieme di altezze usate per identificare i gradi presenti nell'accordo. Il secondo elemento è un testo markup che seguirà l'output di `chordRootNamer` per creare il nome dell'accordo.

`minorChordModifier`
: Gli accordi minori sono spesso indicati con un suffisso "m" a destra della fondamentale dell'accordo. Tuttavia alcuni preferiscono altri suffissi, come il segno meno.

```
\chords {
  c4:min f:min7
  \set minorChordModifier = \markup { "-" }
  \break
  c4:min f:min7
}
```





`chordPrefixSpacer`
: Il modificatore per gli accordi minori determinato da `minorChordModifier` appare solitamente subito dopo la fondamentale dell'accordo. È possibile porre uno spazio tra la fondamentale e il modificatore impostando `chordPrefixSpacer`. Lo spazio non viene usato quando la fondamentale è alterata.

## Comandi predefiniti

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

## Frammenti di codice selezionati

*Eccezioni dei nomi degli accordi*

La proprietà `chordNameExceptions` permette di salvare un elenco di notazioni speciali per accordi specifici.

```
% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

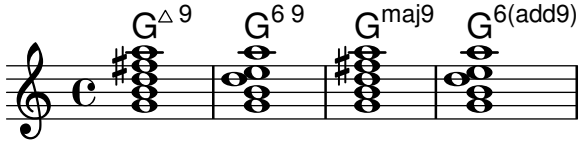
% Convert music to list and prepend to existing exceptions.
```

```
chExceptions = #(append
  (sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

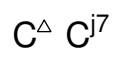
<<
   \new ChordNames \theMusic
   \new Voice \theMusic
>>
```




*Nome dell'accordo di settima maggiore*

La formattazione dell'accordo di settima maggiore può essere regolata con la proprietà `majorSevenSymbol`.

```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```

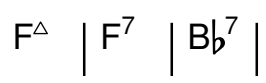



*Aggiungere stanghette al contesto ChordNames*

Per mostrare le stanghette nel contesto `ChordNames`, aggiungere l'incisore `Bar_engraver`.

```
\new ChordNames \with {
  \override BarLine.bar-extent = #'(-2 . 2)
  \consists "Bar_engraver"
}

\chordmode {
  f1:maj7 f:7 bes:7
}
```

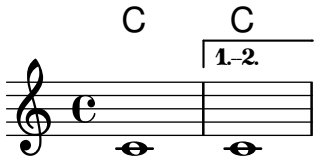



*Volta sotto gli accordi*

Aggiungendo l'incisore `Volta_engraver` al rigo, è possibile inserire le volte sotto gli accordi.

```
\score {
```

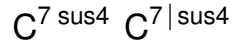
```
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```




*Modifica del separatore dell'accordo*

L'elemento che separa le diverse parti di un accordo può essere impostato su qualsiasi testo di tipo markup.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
}
```




## Vedi anche

Guida alla notazione: Sezione A.1 [Grafico dei nomi degli accordi], pagina 663, Sezione A.2 [Modificatori degli accordi comuni], pagina 664.

Saggio sull'incisione musicale automatizzata: Sezione "Bibliografia" in *Saggio*.

File installati: `scm/chords-ignatzek-names.scm`, `scm/chord-entry.scm`, `ly/chord-modifiers-init.ly`.

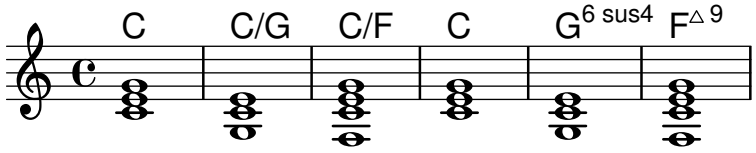
Frammenti: Sezione "Chords" in *Frammenti di codice*.

## Problemi noti e avvertimenti

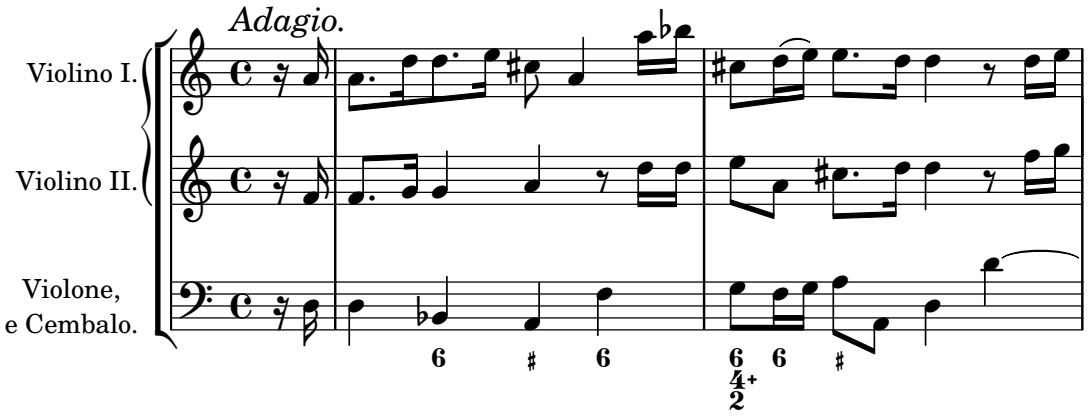
I nomi degli accordi sono determinati sia dalle altezze presenti nell'accordo sia dalle informazioni sulla struttura dell'accordo che potrebbero essere state inserite in `\chordmode`. Se si usa il metodo

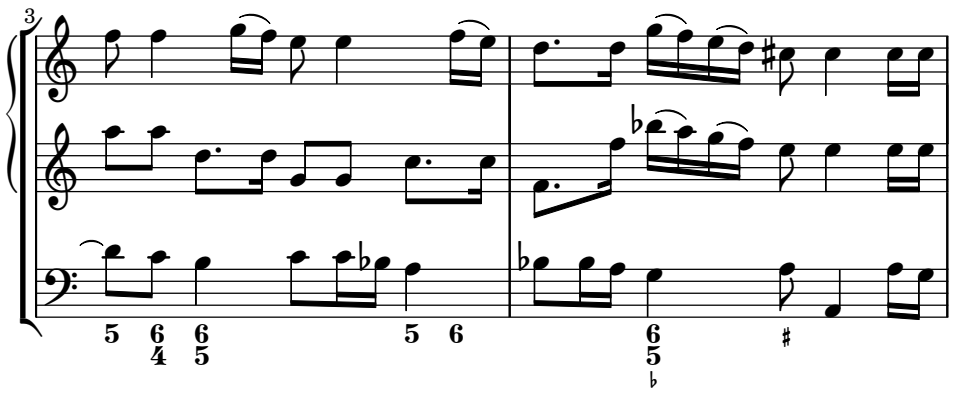
delle altezze simultanee per inserire gli accordi, in caso di inversioni o note di basso si otterranno nomi non voluti.

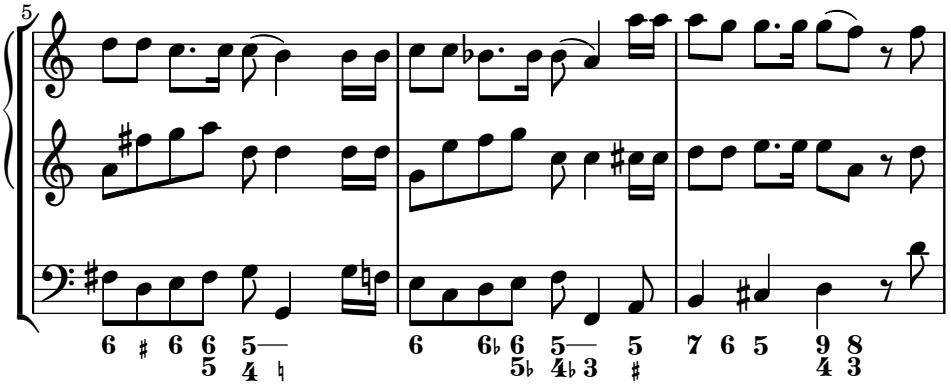
```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```




### 2.7.3 Basso continuo



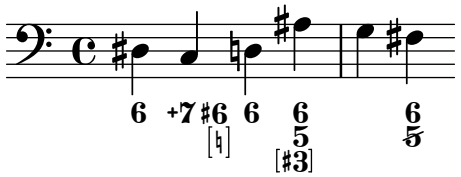


La notazione del basso continuo è supportata.

## Introduzione al basso continuo

LilyPond supporta il basso continuo, detto anche basso numerato o cifrato:

```
<<
  \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
    }
  }
>>
```



Il supporto per il basso continuo consiste in due parti: c'è una modalità di input, introdotta dal comando `\figuremode`, che accetta l'inserimento dei numeri di basso; e c'è un contesto chiamato `FiguredBass` che si occupa di mostrare gli oggetti `BassFigure`. Il basso continuo può essere mostrato anche in contesti `Staff`.

`\figures { ... }` è una scorciatoia per `\new FiguredBass \figuremode { ... }`.

Sebbene il supporto per basso continuo possa assomigliare apparentemente al supporto per accordi, è in realtà molto più semplice. La modalità `\figuremode` si limita a contenere i numeri e il contesto `FiguredBass` li stampa così come sono inseriti. Non c'è alcuna conversione in altezze.

## Vedi anche

Glossario musicale: Sezione "basso continuo" in *Glossario Musicale*.

Frammenti: Sezione "Chords" in *Frammenti di codice*.

## Inserimento del basso continuo

`\figuremode` permette di passare alla modalità di input per il basso continuo. Maggiori informazioni sulle diverse modalità di input si trovano in Sezione 5.4.1 [Modalità di inserimento], pagina 629.

In modalità basso continuo, un gruppo di numeri di basso è delimitato da `<` e `>`. La durata si inserisce dopo il `>`.

```
\new FiguredBass {
```

```
  \figuremode {
    <6 4>2
  }
}
```



Si possono usare le alterazioni (inclusi i bequadri) per modificare i gradi della scala. Si inseriscono aggiungendo `+` (per i diesis), `-` (per i bemolli) o `!` (per i bequadri) dopo il numero. Per le doppie alterazioni il modificatore va applicato due volte. Per modificare il terzo grado il numero di solito viene omesso e ciò si può fare usando `_` al posto del numero.

```
\figures {
  <7! 6+ 4-> <5++> <3--> < _+ > < 7 _!>
}
```



Si possono indicare gradi aumentati e diminuiti:

```
\figures {
  <6\+ 5/> <7/>
}
```



Si può creare una barra inversa che taglia il numero (usata comunemente per i sesti gradi alzati):

```
\figures {
  <6> <6\\>
}
```



I numeri possono essere racchiusi da parentesi:

```
\figures {
  <[12] 8 [6  4]>
}
```

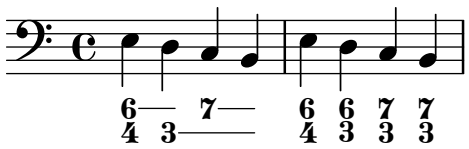


Si può inserire come numero qualsiasi testo di tipo markup:

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```

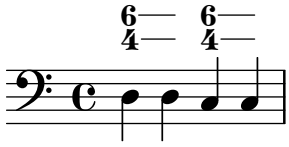
Si possono usare linee di continuazione per indicare numeri ripetuti:

```
<<
  {
    \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>
  }
>>
```

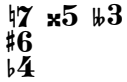


In questo caso, le linee di estensione sostituiscono i numeri esistenti, a meno che le linee di continuazione non siano state terminate esplicitamente.

```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



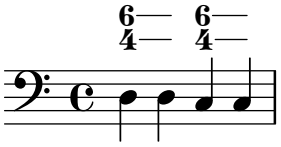
La seguente tabella riassume i modificatori del numero disponibili.

| **Modificatore** | **Scopo** | **Esempio** |
|---|---|---|
| +, -, ! | Alterazioni |  |
| \+, / | Gradi aumentati e diminuiti |  |

\\ Sesto grado alzato



\! Fine della linea di continuazione



## Comandi predefiniti

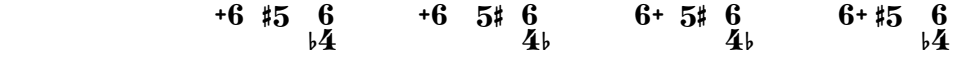
`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

## Frammenti di codice selezionati

*Cambiare la posizione delle alterazioni del basso continuo*

Le alterazioni e i segni più possono apparire prima o dopo i numeri, a seconda delle proprietà `figuredBassAlterationDirection` e `figuredBassPlusDirection`.

```
\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}
```



## Vedi anche

Frammenti: Sezione “Chords” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

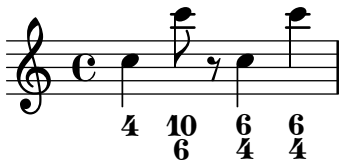
## Visualizzazione del basso continuo

Il basso continuo può essere visualizzato usando il contesto `FiguredBass` o la maggior parte dei contesti del rigo.

Se visualizzato in un contesto `FiguredBass`, la posizione verticale dei numeri è independente dalle note del rigo.

```
<<
  \relative {
    c''4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
```
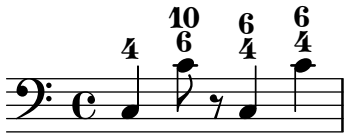
```
      <6 4>4 <6 4>
    }
  }
>>
```



Nell'esempio precedente, il contesto `FiguredBass` deve essere istanziato esplicitamente per evitare la creazione di un secondo rigo (vuoto).

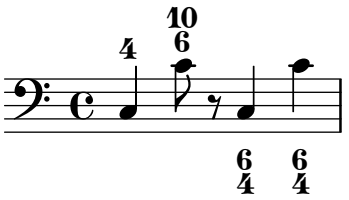
Il basso continuo può anche essere aggiunto direttamente ai contesti `Staff`. In questo caso la posizione verticale dei numeri è regolata automaticamente.

```
<<
  \new Staff = "mioRigo"
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Inserire le note nello stesso rigo dei numeri di basso
  \context Staff = "mioRigo"
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



Se aggiunto in un contesto `Staff`, il basso continuo può essere visualizzato sopra o sotto il rigo.

```
<<
  \new Staff = "mioRigo"
  \figuremode {
    <4>4 <10 6>8 s8
    \bassFigureStaffAlignmentDown
    <6 4>4 <6 4>
  }
  %% Inserire le note nello stesso rigo dei numeri di basso
  \context Staff = "mioRigo"
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



## Comandi predefiniti

`\bassFigureStaffAlignmentDown`, `\bassFigureStaffAlignmentUp`, `\bassFigureStaffAlignmentNeutral`.

## Vedi anche
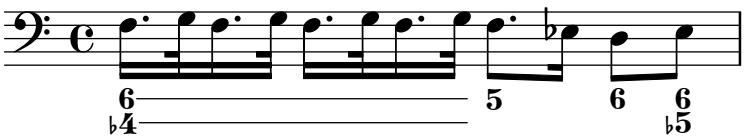
Frammenti: Sezione “Chords” in *Frammenti di codice*.
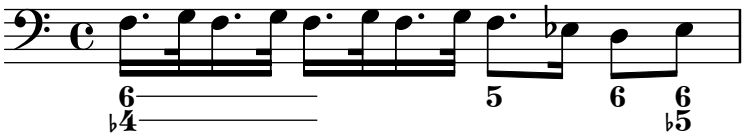
Guida al funzionamento interno: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Per assicurarsi che le linee di continuazione funzionino correttamente, è più sicuro usare lo stesso ritmo nella linea dei numeri e nella linea dei bassi.

```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % Qui gli estensori sono corretti, perché hanno lo stesso ritmo del basso
    \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
    % Qui gli estensori non sono corretti, anche se il tempo è lo stesso
    <6 4->4 <6 4->4
    <5>8. r16 <6>8 <6\! 5->
  }
>>
```





## 2.8 Musica contemporanea

A partire dall'inizio del Ventesimo secolo c'è stata un'enorme espansione di stili e tecniche compositive. Nuovi sviluppi armonici e ritmici, l'estensione dello spettro di altezze e lo sviluppo di un'ampia gamma di nuove tecniche strumentali sono stati accompagnati da una parallela evoluzione e espansione della notazione musicale. L'obiettivo di questa sezione è fornire dei riferimenti e delle informazioni per poter lavorare con queste nuove tecniche di notazione.

### 2.8.1 Altezza e armonica nella musica contemporanea

Questa sezione tratta le questioni relative alla notazione delle altezze e dell'armonia nella musica contemporanea.

#### Riferimenti per altezza e armonia nella musica contemporanea

- La notazione standard per quarti di tono è trattata in [Nomi delle note in altre lingue], pagina 8.
- Armature di chiavi non standard sono affrontate in [Armatura di chiave], pagina 22.
- Le pratiche contemporanee nella visualizzazione delle alterazioni sono discusse in [Alterazioni automatiche], pagina 28.

#### Notazione microtonale

#### Armature di chiave e armonia contemporanee

### 2.8.2 Approcci contemporanei al ritmo

Questa sezione tratta le questioni relative alla notazione del ritmo nella musica contemporanea.

#### Riferimenti per approcci contemporanei al ritmo

- Le indicazioni di tempo composto sono trattate in [Indicazione di tempo], pagina 67.
- La notazione polimetrica di base è trattata in [Notazione polimetrica], pagina 78.
- Le travature a raggiera sono discusse in [Travature a raggiera], pagina 100.
- Le stanghette dello stile "Mensurstriche" (stanghette solo tra i righi) sono trattate in [Raggruppare i righi], pagina 195.

#### Gruppi irregolari nella musica contemporanea

#### Indicazioni di tempo contemporanee

#### Notazione polimetrica estesa

#### Travature nella musica contemporanea

#### Stanghette nella musica contemporanea

### 2.8.3 Notazione grafica

### 2.8.4 Tecniche di arrangiamento contemporaneo

### 2.8.5 Nuove tecniche strumentali

### 2.8.6 Letture consigliate e partiture rilevanti

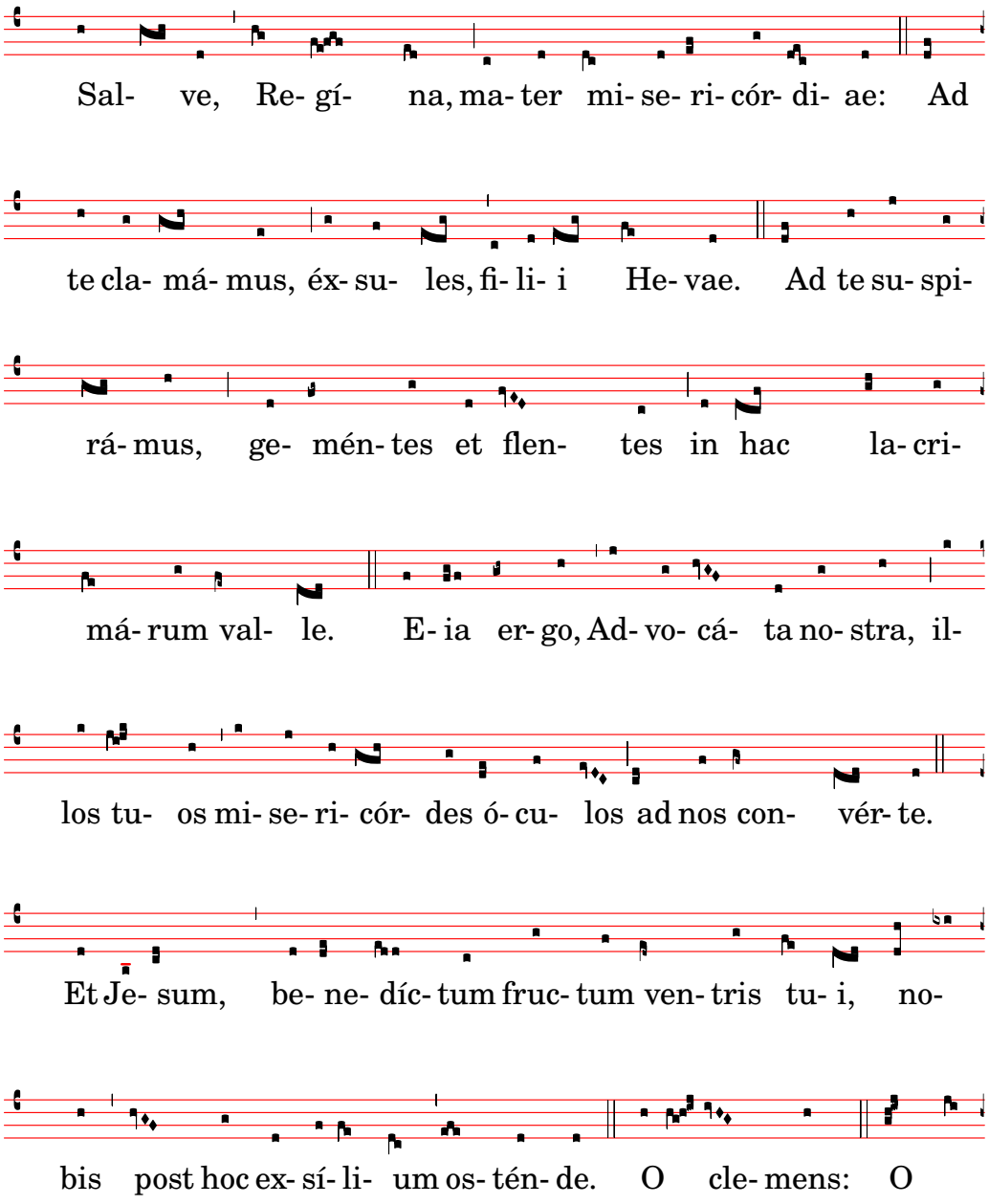
Questa sezione consiglia libri, esempi musicali e altre risorse utili nello studio della notazione musicale contemporanea.

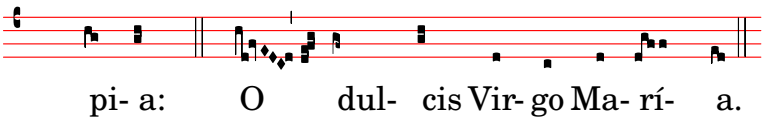
#### Libri e articoli sulla notazione musicale contemporanea

- *Music Notation in the Twentieth Century: A Practical Guidebook* di Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* di Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* di Alfred Blatter [Schirmer, 2nd ed. 1997]

#### Partiture e esempi musicali

## 2.9 Notazione antica






Il supporto per la notazione antica comprende funzionalità per la notazione mensurale, la notazione del canto gregoriano e la notazione quadrata di Kiev. Tali funzionalità sono disponibili sia modificando le proprietà stilistiche di alcuni oggetti grafici, come teste di nota e pause, sia usando uno dei contesti predefiniti per questi stili.

Molti oggetti grafici, come teste di nota e code, alterazioni, indicazioni di tempo e pause, forniscono una proprietà `style`, che può essere cambiata per emulare vari stili diversi di notazione antica. Vedere:

- [Teste di nota mensurali], pagina 447,
- [Alterazioni e armature di chiave mensurali], pagina 449,
- [Pause mensurali], pagina 448,
- [Chiavi mensurali], pagina 445,
- [Chiavi gregoriane], pagina 452,
- [Code mensurali], pagina 448,
- [Indicazioni di tempo mensurali], pagina 446.

Alcuni concetti sono presentati in modo specifico per la notazione antica:

- [Custodes], pagina 443,
- [Divisiones], pagina 454,
- [Legature], pagina 442.

### Vedi anche

Glossario musicale: Sezione “custos” in *Glossario Musicale*, Sezione “legatura” in *Glossario Musicale*, Sezione “notazione mensurale” in *Glossario Musicale*.

Guida alla notazione: [Teste di nota mensurali], pagina 447, [Alterazioni e armature di chiave mensurali], pagina 449, [Pause mensurali], pagina 448, [Chiavi gregoriane], pagina 452, [Code mensurali], pagina 448, [Indicazioni di tempo mensurali], pagina 446, [Custodes], pagina 443, [Divisiones], pagina 454, [Legature], pagina 442.

## 2.9.1 Panoramica degli stili supportati

Per scrivere il canto gregoriano sono disponibili tre stili:

- *Editio Vaticana* è uno stile completo per il canto gregoriano, che riprende l’aspetto delle edizioni Solesmes, i libri ufficiali di canti del Vaticano dal 1904. LilyPond supporta tutti i segni della notazione usati in questo stile, incluse legature, *custodes* e segni speciali come il quilisma e l’oriscus.
- Lo stile *Editio Medicaea* offre alcune funzionalità utilizzate nelle edizioni Medicaea (o Ratisbona), che venivano usate prima delle edizioni Solesmes. Le differenze più rilevanti rispetto allo stile *Vaticana* sono le chiavi, che hanno dei tratti inclinati verso il basso, e le teste di nota, che sono quadrate e regolari.
- Lo stile *Hufnagel* (“chiodo del ferro di cavallo”) o *Gotico* replica lo stile di scrittura dei manoscritti tedeschi e mitteleuropei del Medioevo. Il nome deriva dalla forma essenziale della nota (la *virga*), che assomiglia a un piccolo chiodo.

Tre stili emulano l’aspetto dei manoscritti e delle stampe di musica mensurale del tardo Medioevo e del Rinascimento:

- Lo stile *Mensurale* assomiglia molto allo stile di scrittura usato nei manoscritti del tardo Medioevo e del primo Rinascimento, con le sue teste di nota a diamante, piccole e strette, e le sue pause che ricordano uno stile a mano libera.

- Lo stile *Neomensurale* è una versione rinnovata e stilizzata del precedente: le teste di nota sono più larghe e le pause sono fatte di linee dritte. Questo stile è particolarmente adatto agli incipit dei pezzi trascritti di musica mensurale.
- Lo stile *Petrucci* prende il nome da Ottaviano Petrucci (1466-1539), il primo tipografo a usare i caratteri mobili per la musica (nel suo *Harmonice musices odhecaton*, 1501). Questo stile usa delle teste di nota più grandi rispetto agli altri stili mensurali.

Gli stili *Barocco* e *Classico* non sono stili completi ma differiscono dallo stile predefinito solo in alcuni dettagli: alcune teste di nota (Barocco) e la pausa di un quarto (Classico).

Solo lo stile mensurale ha alternative per tutti gli aspetti della notazione. Negli stili gregoriani non ci sono pause né code, perché questi segni non sono usati nella notazione per canto piano, mentre lo stile Petrucci non ha code o alterazioni specifiche.

Ogni elemento della notazione può essere modificato indipendentemente dagli altri elementi, dunque è possibile usare code mensurali, teste di nota Petrucci, pause classiche e chiavi vaticane nello stesso brano.

### Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*, Sezione “coda” in *Glossario Musicale*.

## 2.9.2 Notazione antica—funzionalità comuni

### Contesti predefiniti

Per la notazione mensurale e del canto gregoriano, sono disponibili i contesti predefiniti della voce e del rigo, che impostano tutti i vari segni della notazione su valori adatti per questi stili. Se i valori predefiniti sono soddisfacenti, si può procedere direttamente all’inserimento delle note senza preoccuparsi di come personalizzare un contesto. Consultare uno dei contesti predefiniti: `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice` e `MensuralStaff`. Vedere anche:

- [Contesti del canto gregoriano], pagina 452,
- [Contesti mensurali], pagina 444.

### Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*.

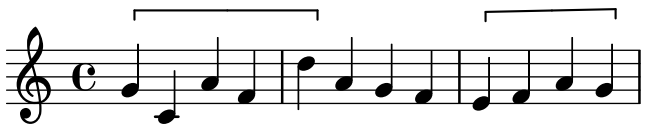
Guida alla notazione: [Contesti del canto gregoriano], pagina 452, [Contesti mensurali], pagina 444.

### Legature

Una legatura è un simbolo grafico che rappresenta almeno due diverse note. Le legature sono apparse in origine nei manoscritti dei canti gregoriani per riferirsi alle sequenze ascendenti e discendenti di note sulla stessa sillaba. Sono utilizzate anche nella notazione mensurale.

Le legature si inseriscono *racchiudendole* tra `\[` e `\]`. Alcuni stili di legatura possono richiedere ulteriore sintassi di input specifica per un particolare tipo di legatura. Nell’impostazione predefinita l’incisore `LigatureBracket` pone una parentesi quadrata sopra la legatura.

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



Sono disponibili altri due stili di legatura: lo stile vaticano per il canto gregoriano e lo stile mensurale per la musica mensurale (sono supportate solo le legature mensurali bianche e con alcune limitazioni). Per usare uno di questi stili, occorre sostituire nel contesto `Voice` l'incisore predefinito `Ligature_bracket_engraver` con uno degli incisori specializzati, come è spiegato in [Legature mensurali bianche], pagina 450, e [Legature di neumi quadrati gregoriani], pagina 456.

## Vedi anche

Glossario musicale: Sezione "legatura" in *Glossario Musicale*.

Guida alla notazione: [Legature mensurali bianche], pagina 450, [Legature di neumi quadrati gregoriani], pagina 456.

## Problemi noti e avvertimenti

La spaziatura richiesta per le legature non è ancora implementata e di conseguenza potrebbe risultare troppo spazio tra di esse. Anche l'interruzione di linea potrebbe essere insoddisfacente.

Il testo vocale potrebbe non allinearsi come desiderato quando si usano le legature.

Le alterazioni non devono apparire all'interno di una legatura, ma devono invece essere raccolte e visualizzate davanti ad essa.

La sintassi utilizza ancora lo stile deprecato "infix" `\[ espressione musicale \]`. Per ragioni di coerenza, dovrà essere cambiato in stile "postfix" `nota\[ ... nota\]`.

Altrimenti, si può includere il file `gregorian.ly`, che fornisce una funzione Scheme

```
\ligature espressione musicale
```

che produce lo stesso risultato (e si ritiene sia stabile).

## Custodes

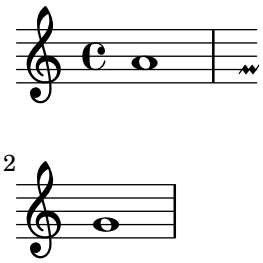
Un *custos* (plurale: *custodes*; parola latina per "custode") è un simbolo che appare alla fine di un rigo. Anticipa l'altezza della prima nota della linea seguente, aiutando quindi l'esecutore a gestire le interruzioni di linea durante l'esecuzione.

I custodes erano usati frequentemente nella notazione musicale fino al diciassettesimo secolo. Attualmente vengono usati solo in poche forme particolari di notazione musicale, per esempio nelle edizioni contemporanee dei canti gregoriani, come l'*Editio Vaticana*. Ci sono glifi diversi per il custos a seconda dei diversi stili di notazione.

Per inserire i custodes, porre l'incisore `Custos_engraver` nel contesto `Staff` del blocco `\layout` e modificare lo stile del custos con un `\override`, se necessario, come è illustrato nell'esempio seguente:

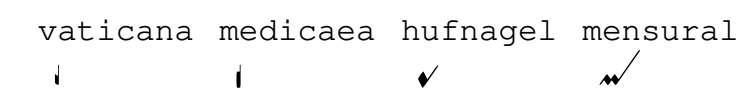
```
\score {
  \relative {
    a'1
    \break
    g
  }
  \layout {
    \context {
      \Staff
      \consists "Custos_engraver"
      \override Custos.style = #'mensural
    }
  }
```

```
}
```



Il glifo del custos viene selezionato tramite la proprietà `style`. Gli stili supportati sono `vaticana`, `medicaea`, `hufnagel` e `mensural`.

```
\new Lyrics \lyricmode {
  \markup { \column {
    \typewriter "vaticana "
    \line { " " \musicglyph "custodes.vaticana.u0" }
  } }
  \markup { \column {
    \typewriter "medicaea "
    \line { " " \musicglyph "custodes.medicaea.u0" }
  }}
  \markup { \column {
    \typewriter "hufnagel "
    \line { " " \musicglyph "custodes.hufnagel.u0" }
  }}
  \markup { \column {
    \typewriter "mensural "
    \line { " " \musicglyph "custodes.mensural.u0" }
  }}
}
```




## Vedi anche

Glossario musicale: Sezione “custos” in *Glossario Musicale*.

Frammenti: Sezione “Ancient notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Custos” in *Guida al Funzionamento Interno*.

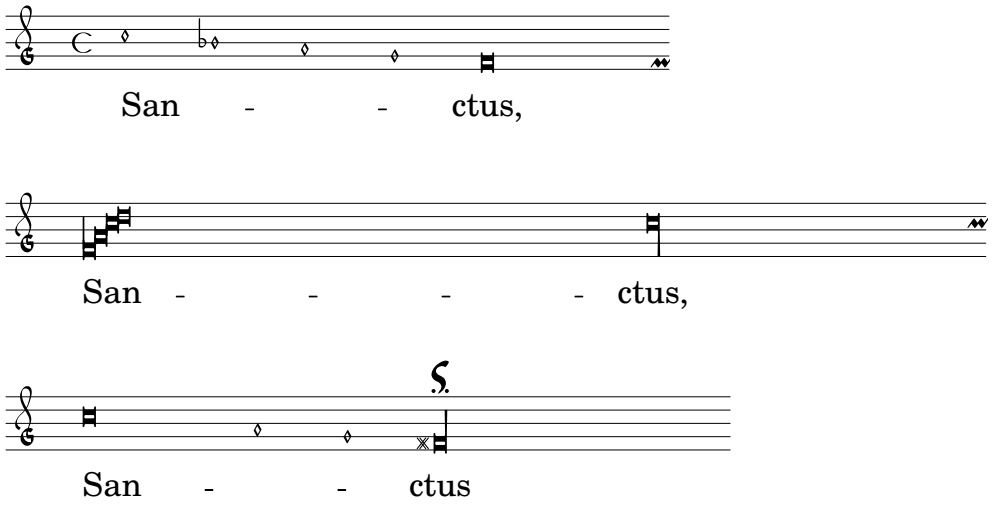
### 2.9.3 Scrivere la musica mensurale

## Contesti mensurali

I contesti predefiniti `MensuralVoice` e `MensuralStaff` permettono di comporre un brano in stile mensurale. Questi contesti inizializzano tutte le proprietà di contesto e dei grob rilevanti su valori adeguati, in modo da poter procedere direttamente con l’inserimento del canto, come dimostra il seguente estratto:

```
\score {
  <<
    \new MensuralVoice = "discantus" \relative {
      \hide Score.BarNumber {
```

```
        c''1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c\breve d\melismaEnd \]
        c\longa
        c\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```




## Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*.

## Chiavi mensurali

Le chiavi mensurali sono supportate tramite il comando `\clef`. Alcune chiavi usano lo stesso glifo, ma si differenziano soltanto per la linea su cui sono stampate. In tali casi, un numero al termine del nome permette di numerare queste chiavi, partendo dalla linea più bassa e salendo verso quella più alta.

```
\override NoteHead.style = #'vaticana.punctum
\clef "vaticana-do1"
c'1
```

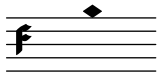


```
\override NoteHead.style = #'medicaea.punctum
\clef "medicaea-do3"
c'1
```
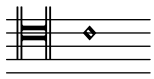


```
\override NoteHead.style = #'hufnagel.punctum
```

```
\clef "hufnagel-fa2"
c'1
```



```
\override NoteHead.style = #'neomensural
\clef "neomensural-c4"
c'1
```



È possibile forzare manualmente il posizionamento del glifo di una chiave su una linea arbitraria, come è spiegato in ⟨undefined⟩ [Clef], pagina ⟨undefined⟩. Tutte le chiavi possibili sono descritte in ⟨undefined⟩ [Clef styles], pagina ⟨undefined⟩.

## Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*, Sezione “chiave” in *Glossario Musicale*.

Guida alla notazione: [Chiavi gregoriane], pagina 452, ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

File installati: `scm/parser-clef.scm`.

Frammenti: Sezione “Pitches” in *Frammenti di codice*.

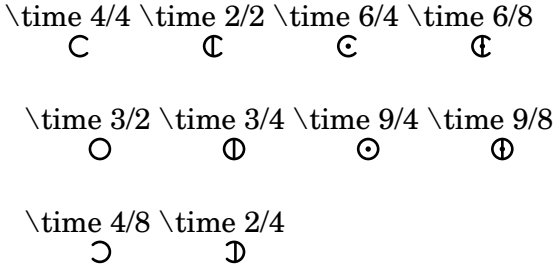
Guida al funzionamento interno: Sezione “Clef_engraver” in *Guida al Funzionamento Interno*, Sezione “Clef” in *Guida al Funzionamento Interno*, Sezione “ClefModifier” in *Guida al Funzionamento Interno*, Sezione “clef-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

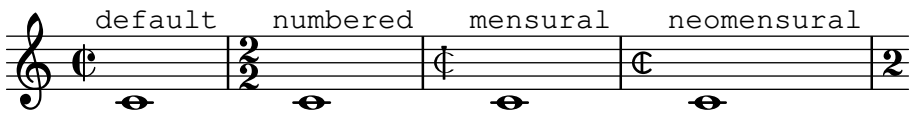
La chiave mensurale di Sol è associata alla chiave Petrucci di Sol.

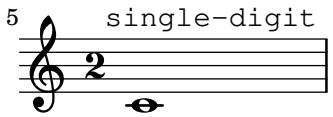
## Indicazioni di tempo mensurali

È disponibile un limitato supporto per i segni della musica mensurale (che sono simili ma non del tutto uguali alle indicazioni di tempo). I glifi sono collegati direttamente a specifiche frazioni temporali. In altre parole, per ottenere un certo segno mensurale col comando `\time n/m`, `n` e `m` devono essere scelti in base alla seguente tabella




Per selezionare le indicazioni di tempo antiche si usa la proprietà `style` del grob `TimeSignature`. Gli stili supportati sono `neomensural` e `mensural`. La tabella precedente utilizza lo stile `neomensural`. Gli esempi successivi mostrano le differenze tra gli stili:






⟨undefined⟩ [Time signature], pagina ⟨undefined⟩, fornisce un'introduzione generale all'uso delle indicazioni di tempo.

## Vedi anche

Glossario musicale: Sezione "mensural notation" in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Time signature], pagina ⟨undefined⟩.

## Problemi noti e avvertimenti

I rapporti tra le durate delle note non possono cambiare con l'indicazione di tempo, perché non sono costanti. Per esempio, il rapporto di 1 breve = 3 semibrevi (*tempus perfectum*) può essere creato manualmente impostando

```
breveTP = #(ly:make-duration -1 0 3/2)
...
{ c\breveTP f1 }
```

`breveTP` viene impostato su 3/2 moltiplicato 2 = 3 volte una nota intera.

I simboli `mensural68alt` e `neomensural68alt` (simboli alternativi per 6/8) non sono gestibili con `\time`. Usare invece `\markup {\musicglyph "timesig.mensural68alt" }`.

## Teste di nota mensurali

Per la notazione antica, è possibile assegnare alle teste di nota uno stile diverso dal predefinito `default`. Per farlo si imposta la proprietà `style` dell'oggetto `NoteHead` su `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` o `semipetrucci`.

Differenze dello stile `baroque` rispetto allo stile `default`:

- Fornisce una testa di nota `maxima`.
- Utilizza una forma quadrata per le teste di nota `\breve`.

Differenze degli stili `neomensural`, `mensural` e `petrucci` rispetto allo stile `baroque`:
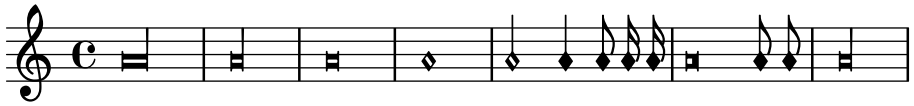
- Usano teste romboidali per le semibrevi e tutte le durate più piccole.
- Centrano i gambi sulle teste di nota.

Lo stile `blackpetrucci` produce teste di nota usabili nella notazione mensurale nera o nelle sezioni "coloratio" della notazione mensurale bianca. Dato che lo stile delle teste di nota non influenza il numero delle code, in questo stile una semiminima deve essere scritta come `a8*2` e non come `a4`, altrimenti avrà lo stesso aspetto di una minima. Il moltiplicatore può essere diverso se si usa la "coloratio", ad esempio per scrivere le terzine.

Usare lo stile `semipetrucci` per ottenere teste di nota semicolorate (breve, longa e maxima).

L'esempio seguente illustra lo stile `petrucci`:

```
\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'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```



⟨undefined⟩ [Note head styles], pagina ⟨undefined⟩, fornisce una panoramica di tutti gli stili di teste di nota disponibili.

## Vedi anche

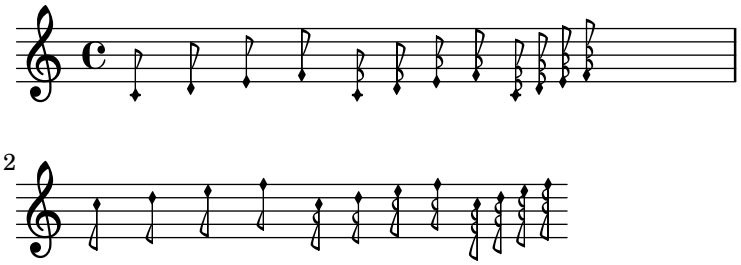
Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*, Sezione “teste di nota” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Note head styles], pagina ⟨undefined⟩.

## Code mensurali

Usare la proprietà `flag-style` del grob `Stem` per selezionare code antiche. Oltre allo stile predefinito `default`, per le code è supportato solo lo stile `mensural`.

```
\relative c' {
  \override Flag.style = #'mensural
  \override Stem.thickness = #1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8
  c'8 d e f c16 d e f c32 d e f
}
```



Si noti che l’estremità più interna di ciascuna coda mensurale è allineata verticalmente con una linea del rigo.

Non esiste uno stile di coda specifico per la notazione neomensurale o Petrucci. Non esistono code nella notazione del canto gregoriano.

## Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*, Sezione “coda” in *Glossario Musicale*.

## Problemi noti e avvertimenti

L’allineamento verticale di ciascuna coda con una linea del rigo presuppone che i gambi terminino sempre esattamente su una linea o esattamente a metà tra due linee. Ciò potrebbe non essere sempre vero se si usano funzionalità di formattazione avanzata della notazione classica (che sono tuttavia non pertinenti per la notazione mensurale).

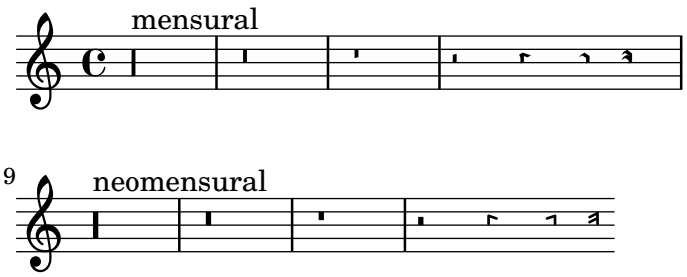
## Pause mensurali

Usare la proprietà `style` del grob `Rest` per selezionare pause antiche. Gli stili antichi supportati sono `neomensural` e `mensural`.

L’esempio seguente illustra questi stili:

```
\set Score.skipBars = ##t
```

```
\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
```




Non esistono pause di un trentaduesimo e di un sessantaquattresimo specifiche per gli stili mensurali o neomensurali; vengono quindi usate le pause dello stile predefinito.

## Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Rests], pagina ⟨undefined⟩.

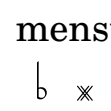
Frammenti: Sezione “Ancient notation” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il glifo della pausa di massima nello stile mensurale è in realtà una pausa di lunga perfetta; usare due (o tre) pause di lunga per ottenere una pausa di massima. Le pause di lunga non sono raggruppate automaticamente, dunque bisogna farlo manualmente usando le pause con altezza.

## Alterazioni e armature di chiave mensurali

Lo stile `mensural` fornisce uno stile dei segni diesis e bemolle diverso da quello predefinito. La notazione mensurale usava raramente il segno di bequadro: al suo posto viene usato il segno diesis o bemolle appropriato. Per esempio, un Si bequadro nella tonalità di Fa maggiore viene indicato con un diesis. Tuttavia, se richiesto esplicitamente, il segno bequadro viene preso dallo stile `vaticana`.




Lo stile di alterazioni e armature di chiave è controllato dalla proprietà `glyph-name-alist` dei grob `Accidental` e `KeySignature`. Per esempio:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

## Vedi anche

Glossario musicale: Sezione “notazione mensurale” in *Glossario Musicale*, Sezione “nomi delle altezze” in *Glossario Musicale*, Sezione “alterazione” in *Glossario Musicale*, Sezione “armatura di chiave” in *Glossario Musicale*.

Guida alla notazione: Sezione 1.1 [Altezze], pagina 1, [Alterazioni], pagina 6, [Alterazioni automatiche], pagina 28, [Armatura di chiave], pagina 22.
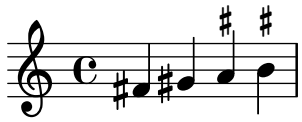
Guida al funzionamento interno: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

## Alterazioni suggerite (*musica ficta*)

Nella musica europea antecedente il 1600, ai cantanti era richiesto di alterare cromaticamente le note di loro iniziativa in base a certe regole. Questa prassi è nota come *musica ficta*. Nelle trascrizioni moderne queste alterazioni vengono poste solitamente sopra la nota.
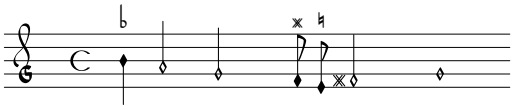
Tali alterazioni suggerite sono supportate e possono essere attivate impostando `suggestAccidentals` su vero.

```
\relative {
  fis' gis
  \set suggestAccidentals = ##t
  ais bis
}
```



In questo modo *ogni* successiva alterazione sarà trattata come *musica ficta* finché la proprietà non viene reimpostata con `\set suggestAccidentals = ##f`. È quindi più pratico usare `\once \set suggestAccidentals = ##t`, che può essere anche definito come una comoda scorciatoia:

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative
  \new MensuralVoice  {
    \once \set suggestAccidentals = ##t
    bes'4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



## Vedi anche

Guida al funzionamento interno: Sezione “Accidental_engraver” in *Guida al Funzionamento Interno*, Sezione “AccidentalSuggestion” in *Guida al Funzionamento Interno*.

## Legature mensurali bianche

Il supporto per le legature mensurali bianche è presente ma limitato.

Per scriverle, sostituire nel blocco layout l’incisore `Ligature_bracket_engraver` con l’incisore `Mensural_ligature_engraver` all’interno del contesto `Voice`:

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Mensural_ligature_engraver"
  }
}
```

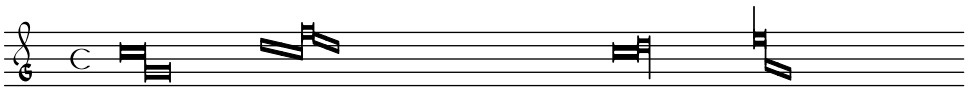
Non esiste un ulteriore linguaggio di input per descrivere la forma di una legatura mensurale bianca. La forma è determinata unicamente dall’altezza e dalla durata delle note racchiuse. Questo approccio, sebbene possa richiedere a un nuovo utente un po’ di tempo per prenderci

confidenza, ha il grande vantaggio che tutte le informazioni musicali della legatura sono note internamente. Non solo ciò è richiesto per avere un output MIDI corretto, ma permette anche la trascrizione automatica delle legature.

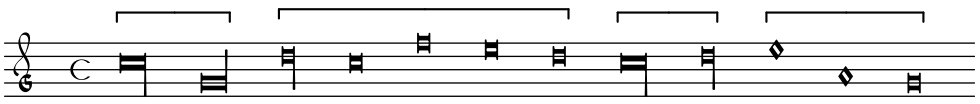
In alcuni punti due note consecutive possono essere rappresentate sia come due quadrati sia come un parallelogramma obliquo (forma *flexa*). In tali casi vengono usati automaticamente i due quadrati, ma si può scegliere una flexa impostando la proprietà `ligature-flexa` della *seconda* testa di nota. La lunghezza di una flexa si imposta tramite la proprietà della testa di nota `flexa-width`.

Per esempio,

```
\score {
  \relative {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c''\maxima g \]
    \[ d'\longa
       \override NoteHead.ligature-flexa = ##t
       \once \override NoteHead.flexa-width = #3.2
       c\breve f e d \]
    \[ c\maxima d\longa \]
    \[ e1 a, g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
    }
  }
}
```



Senza sostituire l'incisore `Ligature_bracket_engraver` con `Mensural_ligature_engraver`, la stessa musica appare così:



## Vedi anche

Glossario musicale: Sezione "legatura" in *Glossario Musicale*.

Guida alla notazione: [Legature di neumi quadrati gregoriani], pagina 456, [Legature], pagina 442.

## Problemi noti e avvertimenti

La spaziatura orizzontale delle legature può essere insoddisfacente. Le alterazioni potrebbero entrare in collisione con le note precedenti.

### 2.9.4 Scrivere il canto gregoriano

Quando si scrive un brano nella notazione del canto gregoriano, l'incisore `Vaticana_ligature_engraver` seleziona automaticamente le teste di nota adeguate, dunque non c'è bisogno di impostare esplicitamente lo stile della testa di nota. È tuttavia possibile modificare lo stile della testa di nota, per esempio su `vaticana_punctum` per produrre i neumi *punctum*. Analogamente, l'incisore `Mensural_ligature_engraver` raggruppa automaticamente le legature mensurali.

#### Vedi anche
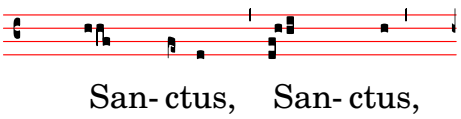
Glossario musicale: Sezione "legatura" in *Glossario Musicale*.
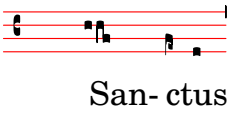
Guida alla notazione: [Legature mensurali bianche], pagina 450, [Legature], pagina 442.

#### Contesti del canto gregoriano

Si possono usare i contesti predefiniti `VaticanaVoice` e `VaticanaStaff` per scrivere un brano di canto gregoriano nello stile Editio Vaticana. Questi contesti inizializzano tutte le proprietà dei grob e di contesto rilevanti impostandole su valori adatti, in modo da non dover far altro che inserire il canto, come dimostra il passo seguente:

```
\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
    }
  >>
}
```



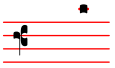
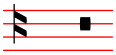
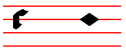




#### Chiavi gregoriane

La tabella seguente mostra tutte le chiavi gregoriane supportate dal comando `\clef`. Alcune chiavi usano lo stesso glifo, ma si differenziano rispetto alla linea in cui sono collocate. In tali casi, si usa un numero in fondo al nome per numerare queste chiavi, dalla linea più bassa a quella più alta. È tuttavia possibile forzare manualmente la posizione del glifo della chiave su una linea

arbitraria, come è spiegato in ⟨undefined⟩ [Clef], pagina ⟨undefined⟩. La nota che appare a destra di ogni chiave nella colonna degli esempi indica il `c'` (Do) relativo a quella chiave.

| **Descrizione** | **Chiavi supportate** | **Esempio** |
|---|---|---|
| stile Editio Vaticana chiave di Do | `vaticana-do1`, `vaticana-do2`, `vaticana-do3` |  |
| stile Editio Vaticana chiave di Fa | `vaticana-fa1`, `vaticana-fa2` |  |
| stile Editio Medicaea chiave di Do | `medicaea-do1`, `medicaea-do2`, `medicaea-do3` |  |
| stile Editio Medicaea chiave di Fa | `medicaea-fa1`, `medicaea-fa2` |  |
| stile hufnagel chiave di Do | `hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3` |  |
| stile hufnagel chiave di Fa | `hufnagel-fa1`, `hufnagel-fa2` |  |
| stile hufnagel chiave combinata Do/Fa | `hufnagel-do-fa` |  |

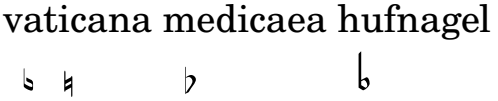
## Vedi anche

Glossario musicale: Sezione “chiave” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

## Alterazioni e armature di chiave gregoriane

Sono disponibili alterazioni per i tre diversi stili gregoriani:




Come si vede, non tutte le alterazioni sono supportate da ciascun stile. Quando si tenterà di accedere a un’alterazione non supportata, LilyPond passerà a uno stile diverso.

Lo stile per le alterazioni e le armature di chiave è controllato dalla proprietà `glyph-name-alist` dei grob `Accidental` e `KeySignature`, rispettivamente; per esempio:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

## Vedi anche

Glossario musicale: Sezione “alterazione” in *Glossario Musicale*, Sezione “armatura di chiave” in *Glossario Musicale*.

Guida alla notazione: Sezione 1.1 [Altezze], pagina 1, [Alterazioni], pagina 6, [Alterazioni automatiche], pagina 28, [Armatura di chiave], pagina 22.

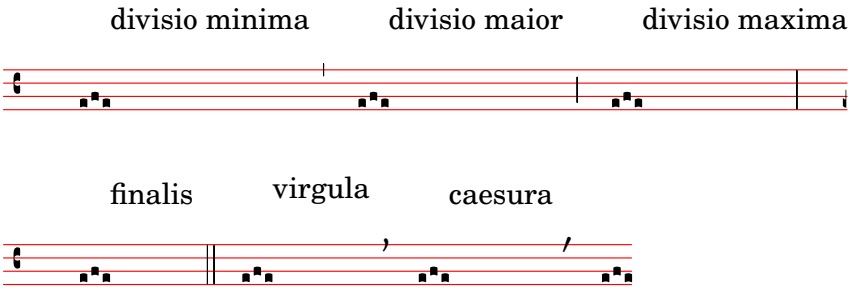
Guida al funzionamento interno: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

## Divisiones

Nella notazione del canto gregoriano non esistono pause; al loro posto si usano le [Divisiones], pagina 454.

Una *divisio* (plurale: *divisiones*; parola latina per “divisione”) è un simbolo di contesto usato per indicare la frase e la struttura delle sezioni della musica gregoriana. Il significato musicale di *divisio minima*, *divisio maior* e *divisio maxima* può essere descritto come pausa breve, media e lunga, un po’ come per i segni di respiro (vedi ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩). Il segno *finalis* non segna soltanto la fine di un canto, ma viene anche usato frequentemente all’interno di un singolo canto antifonale/di risposta per contrassegnare la fine di ciascuna sezione.

Per usare le divisiones, includere il file `gregorian.ly`. Contiene le definizioni che si possono applicare inserendo semplicemente `\divisioMinima`, `\divisioMaior`, `\divisioMaxima` e `\finalis` nei punti giusti dell’input. Alcune edizioni usano *virgula* o *caesura* al posto della divisio minima. Dunque `gregorian.ly` contiene anche una definizione per `\virgula` e `\caesura`.




## Comandi predefiniti

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

## Vedi anche

Glossario musicale: Sezione “caesura” in *Glossario Musicale*, Sezione “divisio” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩.

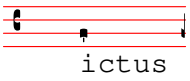
File installati: `ly/gregorian.ly`.

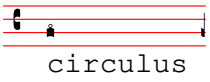
## Segni di articolazione gregoriani

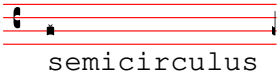
Oltre ai segni di articolazione standard descritti nella sezione [Articolazioni e abbellimenti], pagina 124, sono disponibili dei segni di articolazione progettati specificamente per la notazione in stile *Editio Vaticana*.

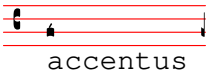
```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript.font-family = #'typewriter
```

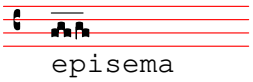
```
    \override TextScript.font-shape = #'upright
    \override Script.padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
  }
}
```




## Vedi anche

Guida alla notazione: [Articolazioni e abbellimenti], pagina 124.

Frammenti: Sezione “Ancient notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Episema” in *Guida al Funzionamento Interno*, Sezione “EpisemaEvent” in *Guida al Funzionamento Interno*, Sezione “Episema_engraver” in *Guida al Funzionamento Interno*, Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “ScriptEvent” in *Guida al Funzionamento Interno*, Sezione “Script_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti
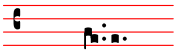
Alcune articolazioni sono posizionate troppo vicine verticalmente alle teste di nota corrispondenti.

## Punti di aumentazione (morae)

I punti di aumentazione, chiamati anche *morae*, si aggiungono con la funzione musicale `\augmentum`. Si noti che `\augmentum` è implementata come funzione musicale unaria invece che come prefisso. Ciò significa che si applica soltanto all’espressione musicale immediatamente seguente. Per esempio, `\augmentum \virga c` non avrà effetti visibili. Scrivere invece `\virga \augmentum c` o `\augmentum {\virga c}`. È possibile scrivere anche `\augmentum {a g}` come scorciatoia per `\augmentum a \augmentum g`.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
```

```
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



## Vedi anche

Guida alla notazione: ⟨undefined⟩ [Breath marks], pagina ⟨undefined⟩.

Guida al funzionamento interno: Sezione “BreathingSign” in *Guida al Funzionamento Interno*.

Frammenti: Sezione “Ancient notation” in *Frammenti di codice*.

## Legature di neumi quadrati gregoriani

LilyPond fornisce un supporto limitato alla notazione quadrata del canto gregoriano (secondo lo stile Editio Vaticana). Possono essere già scritte le legature fondamentali, ma ci sono ancora importanti limiti per una composizione tipografica professionale: tra gli altri, l’allineamento orizzontale di legature multiple, l’allineamento del testo vocale e una corretta gestione delle alterazioni.

Il supporto per i neumi gregoriani si abilita includendo il file `gregorian.ly` all’inizio del file. Così si hanno a disposizione vari comandi aggiuntivi per la produzione di simboli di neumi usati nella notazione del canto gregoriano.

Le teste di nota possono essere *modificate* e/o *unite*.

- La forma della testa di nota può essere modificata *anteponendo* al nome della nota uno dei seguenti comandi: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Le legature, ovvero le note unite insieme, si producono mettendo uno dei comandi di unione, `\pes` o `\flexa`, rispettivamente per il movimento verso l’alto e verso il basso, *tra* le note da unire.

Un nome di nota senza alcun qualificativo produrrà un *punctum*. Tutti gli altri neumi, inclusi quelli a nota singola con una forma diversa come la *virga*, sono considerati come legature e devono quindi essere posti tra `\[...\]`.

Neumi a nota singola:

- Il *punctum* è la forma principale della nota (nello stile *Vaticana*: un quadrato con una leggera curvatura per dare eleganza tipografica). Oltre al normale *punctum*, esiste anche la forma obliqua, chiamata *punctum inclinatum*, prodotta col prefisso `\inclinatum`. Il normale *punctum* può essere modificato con `\cavum`, che produce una nota vuota, e con `\linea`, che disegna linee verticali su entrambi i lati della nota.
- La *virga* ha un gambo discendente sul lato destro. È prodotta dal modificatore `\virga`.

Legature

Diversamente dalla maggior parte degli altri sistemi di notazione dei neumi, l’aspetto tipografico delle legature non è determinato direttamente dai comandi di input, ma segue certe convenzioni a seconda del significato musicale. Per esempio, una legatura di tre note con la forma musicale basso-alto-basso, come `\[ a \pes b \flexa g \]`, produce un Torculus che è composto da tre teste Punctum, mentre la forma alto-basso-alto, come `\[ a \flexa g \pes b \]`, produce un Porrectus con una forma flexa curvata e soltanto una testa Punctum. Non esiste un comando che permetta di gestire esplicitamente la forma flexa curvata; la decisione di quando utilizzarla

viene presa automaticamente da LilyPond in base all'input musicale. L'idea di fondo di questo approccio è separare gli aspetti musicali dell'input dallo stile tipografico dell'output. In questo modo lo stesso input può essere riutilizzato per comporre la stessa musica in uno stile diverso di canto gregoriano.

Neumi liquescenti

Un'altra categoria primaria di note nel canto gregoriano è composta dai cosiddetti neumi liquescenti. Vengono usati in certe circostanze alla fine di una sillaba che termina con una lettera 'liquida', ovvero le consonanti che hanno un tono (le nasali, l, r, v, j e i loro dittonghi equivalenti). Dunque i neumi liquescenti non sono mai usati da soli (sebbene alcuni di questi possano essere prodotti così) e si trovano sempre al termine di una legatura.

I neumi liquescenti sono rappresentati graficamente in due modi diversi, più o meno intercambiabili: con una nota più piccola o "ruotando" la nota principale in su o in giù. Il primo si ottiene con una normale `pes` o `flexa` e modificando la forma della seconda nota: `\[ a \pes \deminutum b \]` ; il secondo si ottiene modificando la forma di un neuma a nota singola con `\auctum` e uno dei segni di direzione `\descendens` o `\ascendens`, per esempio: `\[ \auctum \descendens a \]` .

Segni speciali

Una terza categoria di segni è composta da un piccolo gruppo di segni aventi un significato speciale (e solitamente poco conosciuto): il *quilisma*, l'*oriscus* e lo *strophicus*. Si ottengono anteponendo alla nota il modificatore corrispondente: `\quilisma`, `\oriscus` o `\stropha`.

In pratica, all'interno dei delimitatori della legatura `\[` e `\]`, è possibile ammassare qualsiasi numero di teste per formare una legatura singola; prefissi come `\pes`, `\flexa`, `\virga`, `\inclinatum`, etc., possono essere combinati a piacere. L'uso dell'insieme di regole che sta alla base della costruzione delle legature nella tabella precedente è estrapolato di conseguenza. È così possibile creare moltissime legature diverse.

L'uso di questi segni nella musica stessa segue alcune regole, che non controllate sono da LilyPond. Per esempio, il *quilisma* è sempre la nota centrale di una legatura ascendente, ma è assolutamente possibile, anche se sbagliato, produrre un quelisma di una nota singola.

Oltre ai segni delle note, `gregorian.ly` definisce anche i comandi `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ` e `\IIJ`, che produrranno i caratteri corrispondenti, per esempio per essere usati nel testo vocale, come segni di sezione, etc. Questi comandi utilizzano speciali caratteri Unicode e funzionano soltanto se si usa un font che li supporti.

La tabella seguente illustra un insieme limitato ma rappresentativo di legature gregoriane, insieme ai frammenti di codice che le producono. La tabella si basa sulla tabella dei neumi del secondo volume dell'Antiphonale Romanum (*Liber Hymnarius*), pubblicato nel 1983 dai monaci di Solesmes. La prima colonna dà il nome della legatura, con la forma principale in grassetto e quella liquescente in corsivo. La terza colonna mostra il frammento di codice che produce la legatura, usando `g`, `a` e `b` come altezze di esempio.

**Neumi a nota singola**

| Forme **principali** e *liquescenti* | **Output** | **Codice LilyPond** |
|---|---|---|
| **Punctum** |  | `\[ b \]` |

```
\[ \cavum b \]
```



```
\[ \linea b \]
```



*Punctum Auctum Ascendens*

```
\[ \auctum \ascendens b \]
```



*Punctum Auctum Descendens*

```
\[ \auctum \descendens b \]
```



**Punctum inclinatum**

```
\[ \inclinatum b \]
```



*Punctum Inclinatum Auctum*

```
\[ \inclinatum \auctum b \]
```



*Punctum Inclinatum Parvum*

```
\[ \inclinatum \deminutum b \]
```



**Virga**



**Legature a due note**

**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*



```
\[ \deminutum g \pes \auctum
\descendens b \]
```

**Legature a note multiple**

**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*



```
\[ g \pes a \pes \deminutum b \]
```

**Segni speciali**

**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*



```
\[ \stropha b \stropha b
\stropha a \]
```

### Comandi predefiniti

Sono supportati i seguenti prefissi: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`. Si può usare più di un prefisso, ma con certe restrizioni. Per esempio, si può applicare `\descendens` o `\ascendens` a una nota, ma non entrambi alla stessa nota.

Due note adiacenti possono essere unite insieme coi comandi "infix" `\pes` e `\flexa`, per una linea melodica rispettivamente crescente o calante.

Usare la funzione musicale unaria `\augmentum` per aggiungere punti di aumentazione.

### Vedi anche

Glossario musicale: Sezione "legatura" in *Glossario Musicale*.

Guida alla notazione: [Legature di neumi quadrati gregoriani], pagina 456, [Legature mensurali bianche], pagina 450, [Legature], pagina 442.

### Problemi noti e avvertimenti

Quando un punto di aumentazione creato con `\augmentum` appare in una legatura alla fine dell'ultimo rigo, talvolta viene posizionato male verticalmente. Come soluzione temporanea, aggiungere una nota spaziatrice (per esempio, `s8`) come ultima nota del rigo.

`\augmentum` dovrebbe essere implementato come un prefisso invece che come una funzione musicale unaria, in modo che possa essere combinato con prefissi in ordine arbitrario.
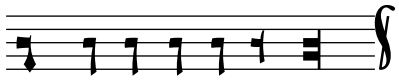
## 2.9.5 Scrivere in notazione quadrata di Kiev

### Contesti della notazione di Kiev

Come per la notazione mensurale e gregoriana, si possono usare i contesti predefiniti `KievanVoice` e `KievanStaff` per scrivere un brano in notazione quadrata. Questi contesti inizializzano tutte le proprietà di contesto e dei grob rilevanti su valori adeguati, così da poter passare immediatamente all'inserimento delle note del canto:

```
% Impostazione dei font per il cirillico
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Linux Libertine O,serif"
   ))
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        c4 c c c c2 b\longa
        \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Го -- спо -- ди по -- ми -- луй.
    }
  >>
}
```



Господи помилуй.

## Vedi anche

Glossario musicale: Sezione “notazione di Kiev” in *Glossario Musicale*.
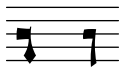
## Problemi noti e avvertimenti

LilyPond supporta la notazione di Kiev dello stile sinodale, come è stato usato nella raccolta di canti stampata dal Santo Sinodo russo nel secondo decennio del Novecento e recentemente ristampato dal Moscow Patriarchate Publishing House. LilyPond non supporta le forme più antiche (meno comuni), utilizzate in Galizia per scrivere il canto liturgico ruteno.

## Chiavi della notazione di Kiev

La notazione di Kievan utilizza una sola chiave, la chiave Tse-fa-ut, usata per indicare la posizione del Do (`c`):

```
\clef "kievan-do"
\kievanOn
c'
```



## Vedi anche

Glossario musicale: Sezione “notazione di Kiev” in *Glossario Musicale*, Sezione “chiave” in *Glossario Musicale*.
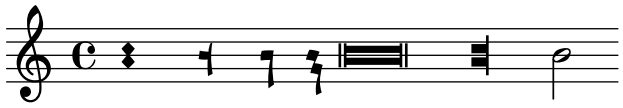
Guida alla notazione: ⟨undefined⟩ [Clef], pagina ⟨undefined⟩.

## Note della notazione di Kiev

Nella notazione quadrata di Kiev, occorre scegliere lo stile appropriato per le teste di nota e disabilitare i gambi e le code. Ciò si ottiene chiamando la funzione `\kievanOn`, che imposta le proprietà di teste di nota, gambi e code. Quando non sono più necessarie le teste di nota di Kiev, tali proprietà possono essere ripristinate tramite la funzione `\kievanOff`.

La nota finale nello stile di Kiev, solitamente al termine di un brano, può essere selezionata impostando la durata su `\longa`. Il segno recitativo di Kiev, usato per indicare il canto di varie sillabe su una nota, si seleziona impostando la durata su `\breve`. L’esempio seguente illustra le varie teste di nota di Kiev:

```
\autoBeamOff
\cadenzaOn
\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2
```



## Vedi anche

Glossario musicale: Sezione “notazione di Kiev” in *Glossario Musicale*, Sezione “testa di nota” in *Glossario Musicale*.

Guida alla notazione: ⟨undefined⟩ [Note head styles], pagina ⟨undefined⟩.
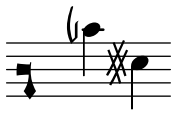
### Problemi noti e avvertimenti

LilyPond determina automaticamente se una nota debba avere il gambo in su o in giù. Tuttavia quando si scrive un canto nella notazione quadrata è consuetudine dirigere i gambi nella stessa direzione all'interno di un singolo melisma. Ciò si può fare manualmente impostando la proprietà `direction` dell'oggetto `Stem`.

### Alterazioni della notazione di Kiev

Lo stile `kievan` per le alterazioni si seleziona tramite la proprietà `glyph-name-alist` del grob `Accidental`. Lo stile `kievan` fornisce un segno di diesis e di bemolle diversi dallo stile predefinito. Nella notazione di Kiev non esiste il segno di bequadro. Il segno di diesis non viene usato nella musica sinodale ma può essere presente nei manoscritti precedenti. È stato incluso soprattutto per garantire la compatibilità.

```
\clef "kievan-do"
\override Accidental.glyph-name-alist =
 #alteration-kievan-glyph-name-alist
bes' dis'
```



### Vedi anche
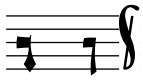
Glossario musicale: Sezione "notazione di Kiev" in *Glossario Musicale*, Sezione "alterazione" in *Glossario Musicale*.

Guida alla notazione: [Alterazioni], pagina 6, [Alterazioni automatiche], pagina 28, ⟨undefined⟩ [The Emmentaler font], pagina ⟨undefined⟩,

### Stanghetta della notazione di Kiev

Alla fine di un brano in notazione di Kiev viene solitamente posta un'immagine decorativa, che si può chiamare la stanghetta finale nello stile di Kiev. Può essere richiamata con `\bar "k"`.

```
\kievanOn
\clef "kievan-do"
c' \bar "k"
```



### Vedi anche

Guida alla notazione: ⟨undefined⟩ [Bars], pagina ⟨undefined⟩, ⟨undefined⟩ [The Emmentaler font], pagina ⟨undefined⟩,

### Melismi della notazione di Kiev

Le note all'interno di un melisma di Kiev vengono solitamente posizionate una vicina all'altra mentre il melisma è separato da uno spazio bianco. Ciò permette al cantante di identificare velocemente le strutture melodiche del canto Znamenny. In LilyPond, i melismi sono trattati come legature e la spaziatura è implementata dall'incisore `Kievan_ligature_engraver`.

Quando si usano i contesti `KievanVoice` e `KievanStaff`, l'incisore `Kievan_ligature_engraver` viene abilitato automaticamente. In altri contesti, può essere

invocato sostituendo l'incisore `Ligature_bracket_engraver` con `Kievan_ligature_engraver` del blocco layout:
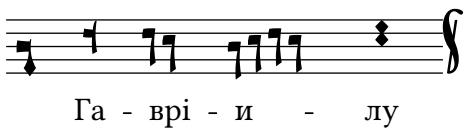
```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Kievan_ligature_engraver"
  }
}
```

La spaziatura tra le note di una legatura di Kiev può essere controllata attraverso la proprietà `padding` di `KievanLigature`.

L'esempio seguente illustra l'uso delle legature di Kiev:

```
% Impostazioni dei font cirillici
\paper {
  #(define fonts
    (set-global-fonts
     #:roman "Linux Libertine O,serif"
   ))
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
        e2 \[ e4( d4 ) \] \[ c4( d e  d ) \] e1 \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Га -- врі -- и -- лу
    }
  >>
}
```




## Vedi anche

Glossario musicale: Sezione "legatura" in *Glossario Musicale*.

Guida alla notazione: [Legature mensurali bianche], pagina 450, [Legature di neumi quadrati gregoriani], pagina 456, [Legature], pagina 442.

## Problemi noti e avvertimenti

La spaziatura orizzontale delle legature è insoddisfacente.

### 2.9.6 Lavorare con la musica antica—scenari e soluzioni

Lavorare con la musica antica richiede spesso compiti particolari che differiscono notevolmente dalla notazione moderna per cui LilyPond è progettato. Nel resto di questa sezione vengono delineati vari scenari tipici e le loro possibili soluzioni:
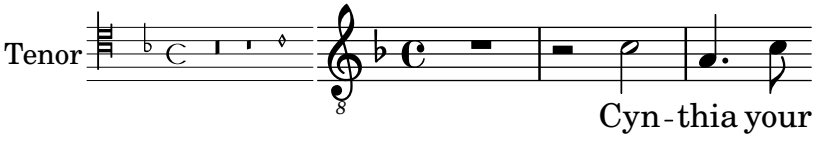
- come creare gli incipit (ovvero materiale introduttivo per indicare come appariva l'originale) a trascrizioni moderne della musica mensurale;

- come ottenere la formattazione *Mensurstriche* usata frequentemente per le trascrizioni moderne della musica polifonica;
- come trascrivere il canto gregoriano nella notazione moderna;
- come generare la notazione sia antica che moderna dallo stesso file sorgente.

## Incipit

Nella trascrizione di musica mensurale in notazione moderna è consuetudine inserire un'indicazione di come apparivano le pause e la nota o le note iniziali della versione originale (incluse le chiavi originali). Viene chiamata *incipit*. Il comando `\incipit` usa l'`indent` del rigo principale per impostare la larghezza occupata dall'incipit, e `incipit-width` per impostare la larghezza del rigo dell'incipit.

```
\score {
  \new Staff <<
    \new Voice = Tenor {
      \set Staff.instrumentName = "Tenor"
      \override Staff.InstrumentName.self-alignment-X = #RIGHT
      \incipit { \clef "mensural-c4" \key f \major r\breve r1 c'1 }
      \clef "treble_8"
      \key f \major
      R1 r2 c'2 |
      a4. c'8
    }
    \new Lyrics \lyricsto Tenor { Cyn -- thia your }
  >>
  \layout
  {
    indent = 5\cm
    incipit-width = 3\cm
  }
}
```




## Problemi noti e avvertimenti

Attenzione: `instrumentName` deve essere impostato perché l'incipit venga prodotto. Se non è richiesto il nome di uno strumento, usare `\set Staff.instrumentName = ""`.

## Formattazione mensurstriche

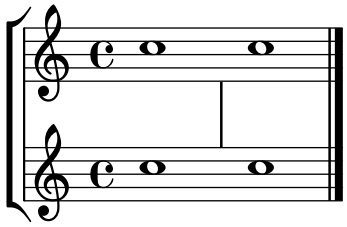
*Mensurstriche* ('linee di misurazione') è il termine accettato per le stanghette che collegano i righi di un sistema senza attraversare i righi stessi. È un modo per preservare l'aspetto ritmico dell'originale, ovvero non dover interrompere note sincopate che si trovano al cambio di battuta, continuando a fornire la funzione di orientamento che le stanghette hanno.

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta con `\hide`.

```
global = {
  \hide Staff.BarLine
```

```
    s1 s
    % the final bar line is not interrupted
    \undo \hide Staff.BarLine
    \bar "|."
  }

  \new StaffGroup \relative c'' {
    <<
      \new Staff { << \global { c1 c } >> }
      \new Staff { << \global { c c } >> }
    >>
  }
```



## Trascrivere il canto gregoriano

Il canto gregoriano può essere trascritto in notazione moderna con alcune semplici modifiche.

**Gambi**. I gambi possono essere tolti del tutto rimuovendo col comando `\remove` l'incisore `Stem_engraver` dal contesto Voice:

```
  \layout {
    ...
    \context {
      \Voice
        \remove "Stem_engraver"
    }
  }
```

**Tempo.** Per i canti senza metro esistono varie alternative.

Si può togliere l'incisore `Time_signature_engraver` dal contesto Staff senza alcun effetto collaterale negativo. Se invece lo si rende trasparente, rimarrà uno spazio bianco nella partitura, perché l'indicazione di tempo prenderà comunque spazio.

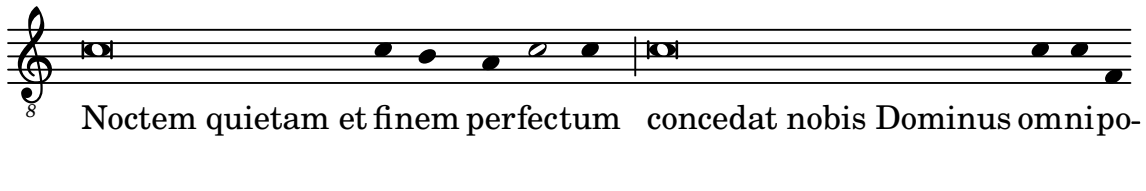
In molti casi, usare `\set Score.timing = ##f` darà buoni risultati. Un'altra possibilità è l'uso di `\cadenzaOn` e `\cadenzaOff`.

Per togliere le stanghette, l'approccio radicale è togliere l'incisore `Bar_engraver` dal contesto Staff. Ma si può anche usare `\hide BarLine`, se si desidera una stanghetta saltuariamente.

Un tipo comune di trascrizione è il canto recitativo, dove le note ripetute sono indicate con una singola breve. Il testo per il tono della recitazione può essere trattato in due modi diversi: o impostato come una singola sillaba allineata a sinistra:

```
\include "gregorian.ly"
chant = \relative {
  \clef "G_8"
  c'\breve c4 b4 a c2 c4  \divisioMaior
  c\breve c4 c f, f \finalis
}
```

```
verba = \lyricmode {
  \once \override LyricText.self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText.self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}
\score {
  \new Staff <<
  \new Voice = "melody" \chant
  \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Voice
      \remove "Stem_engraver"
    }
  }
}
```



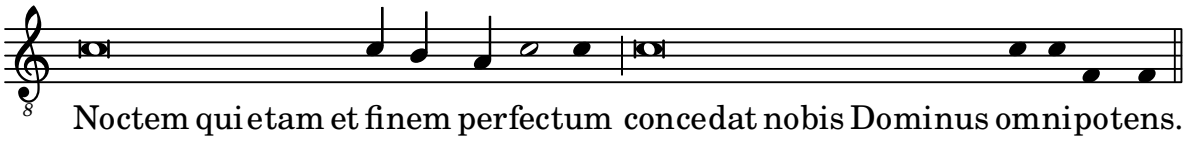

Questo esempio funziona bene, purché il testo non si estenda oltre un'interruzione di linea. Se ciò accade, un'alternativa è aggiungere delle note nascoste alla partitura, come mostrato sotto.

In alcuni stili di trascrizione, i gambi vengono usati occasionalmente, per esempio per indicare la transizione da un recitativo di un singolo tono a un atto melodico fisso. In questi casi, si può usare `\hide Stem` o `\override Stem.length = #0` invece di togliere l'incisore `Stem_engraver` e ripristinare il gambo quando serve col corrispondente `\undo \hide Stem`.

```
\include "gregorian.ly"
chant = \relative {
  \clef "G_8"
  \set Score.timing = ##f
  \hide Stem
  c'\breve \hide NoteHead  c c c c c
  \undo \hide NoteHead
  \undo \hide Stem \stemUp c4 b4 a
  \hide Stem c2 c4  \divisioMaior
  c\breve \hide NoteHead c c c c c c c
  \undo \hide NoteHead c4 c f, f \finalis
}
```

```
verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics \lyricsto "melody" \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \hide BarLine
    }
  }
}
```



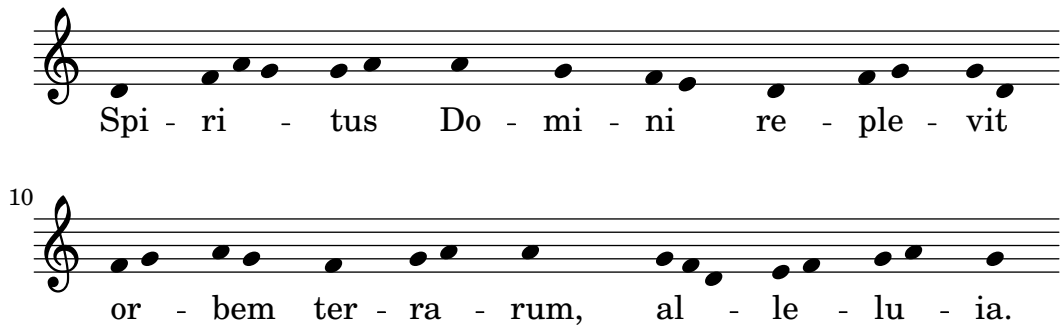

Un'altra situazione tipica è la trascrizione dei canti neumatici o melismatici, ovvero i canti che hanno un numero variabile di note per ciascuna sillaba. In questo caso si vorranno impostare i gruppi di sillabe in modo chiaramente separato, solitamente anche le suddivisioni di un melisma più lungo. Un modo per ottenere ciò è usare un tempo fisso, per esempio 1/4, e lasciare che ogni sillaba o gruppo di note riempa una di queste misure, con l'aiuto di gruppi irregolari o durate più brevi. Se le stanghette e tutte le altre indicazioni ritmiche sono rese trasparenti, e se lo spazio intorno alle stanghette viene aumentato, si otterrà una rappresentazione dell'originale in notazione moderna piuttosto buona.

Per evitare che le sillabe di larghezza diversa (come "-ri" e "-rum") si estendano sui gruppi di note delle sillabe in modo non uniforme, la proprietà `'X-extent` dell'oggetto `LyricText` può essere impostata su un valore fisso. Un altro modo, più scomodo, consiste nell'aggiungere le sillabe come elementi `\markup`. Se sono necessari ulteriori aggiustamenti, si può fare facilmente con le pause spaziatrici `s`.

```
spiritus = \relative {
  \time 1/4
  \override Lyrics.LyricText.X-extent  = #'(0 . 3)
  d'4 \tuplet 3/2 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4  s
  \tuplet 3/2 { g8 f d } e f g a g4
}

spirLyr = \lyricmode {
  Spi -- ri -- _ _ tus  _ Do -- mi -- ni  _ re -- ple -- _ vit _
  or -- _ bem _  ter -- ra -- _ rum, al -- _ _ le -- _ lu
  -- _ ia.
}
\score {
```

```
  \new Staff <<
    \new Voice = "chant" \spiritus
    \new Lyrics = "one" \lyricsto "chant" \spirLyr
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine.X-extent = #'(-1 . 1)
      \hide Stem
      \hide Beam
      \hide BarLine
      \hide TupletNumber
    }
  }
}
```




## Antico e moderno da un unico sorgente

*Usare le etichette per produrre musica mensurale e moderna dallo stesso sorgente*

Usando le etichette (tag), è possibile usare la stessa musica per produrre sia la musica mensurale che quella moderna. In questo frammento, viene introdotta la funzione `menrest`, che permette alle pause mensurali di essere posizionate precisamente sul rigo come nell'originale, ma con le pause moderne nella posizione standard. Le etichette vengono usate per produrre diversi tipi di stanghetta alla fine della musica, ma possono essere usate anche quando sono necessarie altre differenze: per esempio se si vogliono usare "pause d'intero" (R1, R\breve, etc.) nella musica moderna, ma pause normali (r1, r\breve, etc.) nella versione mensurale. La conversione di musica mensurale nel suo equivalente moderno viene solitamente chiamata `trascrizione`.

```
menrest = #(define-music-function (note)
  (ly:music?)
#{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
#})

MensStyle = {
  \autoBeamOff
  \override NoteHead.style = #'petrucci
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
}
```

```
finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
  \once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

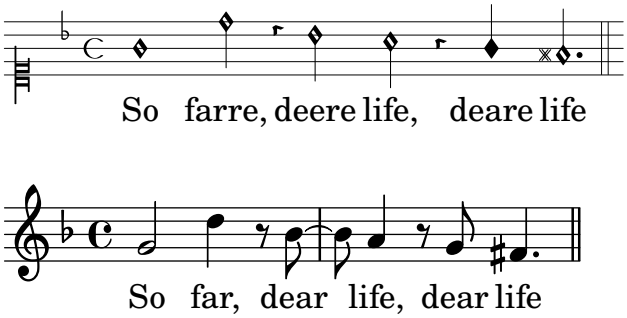
  \breathe
}

Music = \relative c'' {
  \set Score.tempoHideNote = ##t
  \key f \major
  \time 4/4
  g1 d'2 \menrest bes4 bes2 a2 r4 g4 fis2.
  \tag #'mens { \finalis }
  \tag #'mod { \bar "||" }
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {
  \keepWithTag #'mens {
    <<
      \new MensuralStaff
      {
        \new MensuralVoice = Cantus
          \clef "mensural-c1" \MensStyle \Music
      }
      \new Lyrics \lyricsto Cantus \MenLyr
    >>
  }
}

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff
      {
        \new Voice = Sop \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver" }
        {
          \shiftDurations #1 #0 { \autoBeamOff \Music }
        }
      }
      \new Lyrics \lyricsto Sop \ModLyr
    >>
  }
}
```




### Correzioni editoriali

## 2.10 World music

Questo capitolo tratta la notazione delle musiche tradizionali diverse da quelle occidentali.

### 2.10.1 Notazione comune per la musica non occidentale

Questa sezione spiega come scrivere e stampare partiture musicali che non appartengono alla tradizione classica occidentale, chiamata anche *Periodo di pratica comune*.

### Estensione dei sistemi di notazione e di accordatura

La notazione classica standard (nota anche come notazione del *Periodo di pratica comune*) viene usata in tutti i generi musicali, non solo nella musica 'classica' occidentale. Questa notazione è trattata in Sezione 1.1.1 [Inserimento delle altezze], pagina 1, mentre i vari nomi di nota che possono essere usati sono spiegati in [Nomi delle note in altre lingue], pagina 8.

Tuttavia, molti generi musicali non occidentali (e alcuni tipi di musica folk occidentale e tradizionale) utilizzano sistemi di accordatura alternativi o estesi che non rientrano facilmente nella notazione classica standard.

In alcuni casi viene usata comunque la notazione standard, mentre le differenze di altezza restano implicite. Per esempio, la *musica araba* viene rappresentata con le alterazioni standard di un semitono e di un quarto di tono, e le esatte alterazioni di altezza sono determinate in base al contesto. Vengono usati solitamente i nomi italiani delle note, mentre il file di inizializzazione `arabic.ly` fornisce un comodo insieme di macro e definizioni che estendono la notazione standard. Maggiori dettagli in Sezione 2.10.2 [Musica araba], pagina 473.

Altri generi musicali richiedono notazioni estese o uniche. La *musica classica turca* o musica ottomana, per eesempio, usa forme melodiche note come *makamlar*, i cui intervalli sono basati su divisioni di 1/9 del tono intero. Usa comunque le note sul rigo standard occidentale, ma con alterazioni speciali presenti esclusivamente nella musica turca, definite nel file `makam.ly`. Maggiori informazioni sulla musica classica turca e sui makamlar in Sezione 2.10.3 [Musica classica turca], pagina 478.

Per trovare i file di inizializzazione come `arabic.ly` o `makam.ly` nel proprio sistema, leggere Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

### Frammenti di codice selezionati
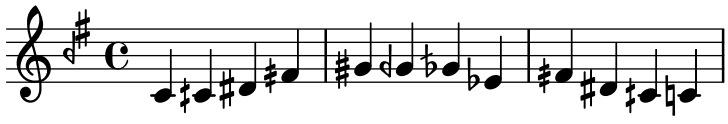
*Esempio di makam*

Makam è un tipo di melodia proveniente dalla Turchia che usa alterazioni microtonali di 1/9. Consultare il file di inizializzazione '`ly/makam.ly`' per vedere come sono definiti i nomi delle altezze e le alterazioni.

```
% Initialize makam settings
\include "makam.ly"

\relative c' {
  \set Staff.keyAlterations = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
```

```
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



## Vedi anche

Glossario musicale: Sezione “Periodo di pratica comune” in *Glossario Musicale*, Sezione “makamlar” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.1.1 [Inserimento delle altezze], pagina 1, [Nomi delle note in altre lingue], pagina 8, Sezione 2.10.2 [Musica araba], pagina 473, Sezione 2.10.3 [Musica classica turca], pagina 478.

### 2.10.2 Musica araba

Questa sezione evidenzia le questioni relative alla notazione della musica araba.

## Referenze per la musica araba

Finora la musica araba è stata soprattutto una tradizione orale. La musica, se trascritta, viene solitamente schematizzata in una sorta di abbozzo, sul quale gli esecutori hanno molta libertà di improvvisazione. Ma, per poter comunicare e preservare la musica araba, l’uso della notazione accidentale, con alcune variazioni, è sempre più diffuso.

Alcuni elementi della notazione musicale occidentale, come la trascrizione di accordi e parti independenti, non sono necessari per scrivere i brani arabi più tradizionali. Ci sono tuttavia alcune questioni differenti, come il bisogno di indicare gli intervalli intermedi che si trovano tra un semitono e un tono, oltre agli intervalli minori e maggiori usati nella musica occidentale. C’è anche il bisogno di raggruppare e indicare un gran numero di diversi maqam (modi) che fanno parte della musica araba.

In generale, la notazione della musica araba non tenta di indicare precisamente gli elementi microtonali presenti nella pratica musicale.

Varie questioni rilevanti per la musica araba sono trattate in altre sezioni della documentazione:

- I nomi delle note e le alterazioni (inclusi i quarti di tono) possono essere modificati come è spiegato in Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.
- Ulteriori armature di chiave possono essere adattate come descritto in [Armatura di chiave], pagina 22.
- Indicazioni di tempo complesse potrebbero obbligare a raggruppare le note manualmente, come descritto in [Travature manuali], pagina 97.
- I *takasim*, improvvisazioni in tempo libero, possono essere scritti omettendo le stanghette, come è spiegato in [Musica in tempo libero], pagina 76.

## Vedi anche

Guida alla notazione: Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472, [Armatura di chiave], pagina 22, [Travature manuali], pagina 97.
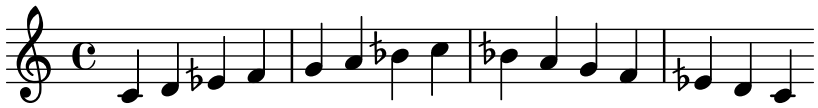
Frammenti: Sezione “World music” in *Frammenti di codice*.

## Nomi delle note in arabo

I nomi più tradizionali delle note in arabo possono essere piuttosto lunghi e non sono adatti alla scrittura musicale, quindi non vengono suati. I nomi delle note in inglese non sono molto conosciute nell'istruzione musicale araba, quindi al loro posto sono usati i nomi delle note in italiano, ovvero i nomi usati nel solfeggio (`do, re, mi, fa, sol, la, si`); si possono usare anche le alterazioni. I nomi delle note in italiano e le alterazioni sono spiegati in [Nomi delle note in altre lingue], pagina 8; l'uso della notazione occidentale standard per scrivere musica non occidentale è trattato in Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.
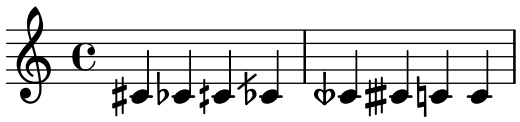
Ecco un esempio di come può essere scritta la scala araba *rast*:

```
\include "arabic.ly"
\relative {
  do' re misb fa sol la sisb do sisb la sol fa misb re do
}
```



Il simbolo di semibemolle non corrisponde al simbolo usato nella notazione araba. Per ottenere il simbolo arabo di semibemolle, si può usare il simbolo `\dwn`, definito in `arabic.ly`, prima di un simbolo di bemolle. L'aspetto del simbolo di semibemolle nell'armatura di chiave non può essere modificato con questo metodo.

```
\include "arabic.ly"
\relative {
  \set Staff.extraNatural = ##f
  dod' dob dosd \dwn dob dobsb dodsd do do
}
```



## Vedi anche

Guida alla notazione: [Nomi delle note in altre lingue], pagina 8, Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

Frammenti: Sezione "World music" in *Frammenti di codice*.

## Armature di chiave arabe

Oltre alle armature di chiave minori e maggiori, sono definite le seguenti armature in `arabic.ly`: *bayati*, *rast*, *sikah*, *iraq* e *kurd*. Queste armature definiscono un piccolo numero di gruppi di maqam invece del gran numero di maqam di uso comune.

In generale, un maqam usa l'armatura di chiave del suo gruppo o di un gruppo vicino, e le variazioni di alterazioni sono contrassegnate nel corso della musica.

Per esempio, per indicare l'armatura di chiave di un brano maqam muhayer:

```
\key re \bayati
```

dove *re* è l'altezza predefinita del maqam muhayer, e *bayati* è il nome del maqam di base nel gruppo.

Sebbene l'armatura di chiave indichi il gruppo, di solito si precisa nel titolo il maqam più specifico, dunque in questo esempio il nome di maqam muhayer dovrebbe apparire nel titolo.

Altri maqam nello stesso gruppo bayati, come mostrato nella tabella in basso (bayati, hussaini, saba, and ushaq), possono essere indicati nello stesso modo. Sono tutte variazioni del maqam di base e più comune nel gruppo, ovvero bayati. Solitamente differiscono dal maqam di base nei tetracordi superiori, o in certi dettagli di disposizione che non cambiano fondamentalmente la loro natura.

L'altro maqam nello stesso gruppo (Nawa) è correlato con bayati per la modulazione, indicata in tabella tra parentesi per quei maqam che sono modulazioni del loro maqam fondamentale. I maqam arabi permettono soltanto modulazioni limitate, a causa della natura degli strumenti musicali arabi. Nawa può essere indicato nel modo seguente:

```
\key sol \bayati
```

Nella musica araba, lo stesso termine, come bayati, usato per indicare un gruppo maqam, è anche un maqam che è solitamente il più importante nel gruppo, e può anche essere considerato come il maqam di base.

Ecco un raggruppamento consigliato che mostra le armature di chiave dei più comuni maqam:

| **gruppo maqam** | **tonalità** | **finalis** | **Altri maqma nel gruppo (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 |

## Frammenti di codice selezionati

*Armature di chiave non tradizionali*

Il comando `\key` comunemente usato imposta la proprietà `keyAlterations` del contesto `Staff`. Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente.

Il formato di questo comando è una lista:
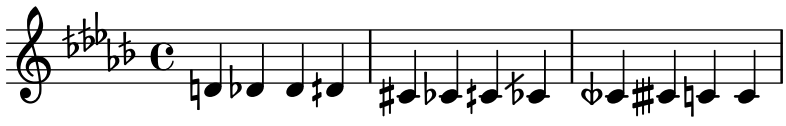
`\set Staff.keyAlterations = #`(((ottava . grado) . alterazione) ((ottava . grado) . alterazione) ...)`

dove, per ogni elemento della lista, `ottava` indica l'ottava (0 è l'ottava dal Do centrale al Si precedente), `grado` indica la nota all'interno dell'ottava (0 significa Do e 6 significa Si) e `alterazione` può essere `,SHARP` `,FLAT` `,DOUBLE-SHARP` etc.

Altrimenti, usando il formato breve per ogni elemento della lista, `(grado . alterazione)`, ciò indica che la stessa alterazione deve essere presente in tutte le ottave. Per le scale microtonalidove un "diesis" non è 100 centesimi, `alterazione` si riferisce alla proporzione di un duecentesimo di tono intero.

```
\include "arabic.ly"
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                 (1 . ,SEMI-FLAT)
                                 (2 . ,FLAT)
                                 (5 . ,FLAT)
                                 (6 . ,SEMI-FLAT))
%\set Staff.extraNatural = ##f
```

```
  re reb \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dodsd do do |
}
```



## Vedi anche

Glossario musicale: Sezione “maqam” in *Glossario Musicale*, Sezione “bayati” in *Glossario Musicale*, Sezione “rast” in *Glossario Musicale*, Sezione “sikah” in *Glossario Musicale*, Sezione “iraq” in *Glossario Musicale*, Sezione “kurd” in *Glossario Musicale*.

Guida alla notazione: [Armatura di chiave], pagina 22.

Manuale di apprendimento: Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.

Guida al funzionamento interno: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

Frammenti: Sezione “World music” in *Frammenti di codice*, Sezione “Pitches” in *Frammenti di codice*.

## Indicazioni di tempo arabe

Alcune forme di musica classica araba e turca, come *Semai*, usano indicazioni di tempo insolite, come 10/8. Ciò può determinare un raggruppamento automatico delle note piuttosto diverso dalle partiture esistenti, dove le note possono non essere raggruppate sul tempo, ma in un modo che è difficile da cogliere aggiustando la disposizione automatica delle travature. L’alternativa consiste nel disattivare la disposizione automatica delle travature e disporre le travature manualmente. Anche se non è richiesto di riprodurre lo stile di una partitura esistente, può essere comunque opportuno regolare il comportamento della disposizione automatica delle travature e/o usare indicazioni di tempo composto.

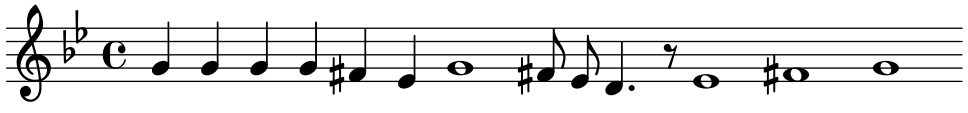
## Frammenti di codice selezionati

*Improvvisazione araba*

Per improvvisazioni o *taqasim* a tempo libero, si può omettere l’indicazione di tempo e usare `\cadenzaOn`. Può essere necessaria la modifica dello stile delle alterazioni, perché l’assenza delle stanghette farà sì che l’alterazione sia contrassegnata una volta sola. Ecco un esempio di quello che potrebbe essere l’inizio di un’improvvisazione *hijaz*:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



## Vedi anche

Glossario musicale: Sezione “semai” in *Glossario Musicale*, Sezione “taqasim” in *Glossario Musicale*.

Guida alla notazione: [Travature manuali], pagina 97, [Travature automatiche], pagina 85, [Musica in tempo libero], pagina 76, [Alterazioni automatiche], pagina 28, [Impostare il comportamento delle travature automatiche], pagina 88, [Indicazione di tempo], pagina 67.
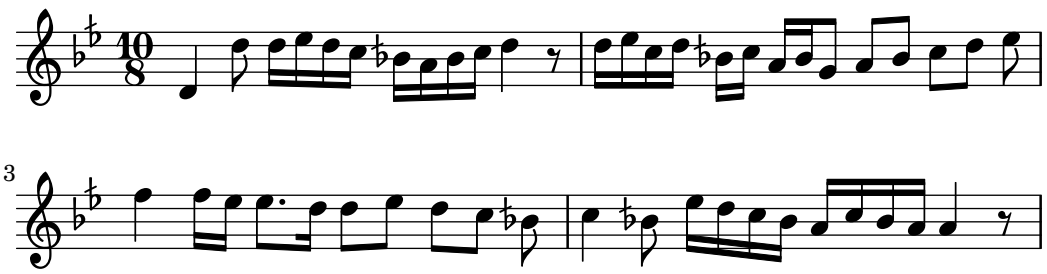
Frammenti: Sezione “World music” in *Frammenti di codice*.

## Esempio di musica araba

Ecco un modello che usa l’inizio di un *Semai* turco, conosciuto nell’educazione musicale araba, per illustrare le peculiarità della notazione musicale araba, come gli intervalli intermedi e i modi inusuali discussi in questa sezione.

```
\include "arabic.ly"
\score {
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
  \relative {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re'4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re  [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
}
```



## Vedi anche

Frammenti: Sezione “World music” in *Frammenti di codice*.

## Letture complementari sulla musica araba

1. *The music of the Arabs* di Habib Hassan Touma [Amadeus Press, 1996], contiene uno studio dei maqam e il loro metodo di classificazione.

   Ci sono anche vari siti web che spiegano i maqam e alcuni offrono degli esempi audio, come:

   - `http://www.maqamworld.com/`
   - `http://www.turath.org/`

Nonostante ci sia un generale consenso sui criteri di classificazione dei maqam che sono collegati tra loro a causa di comuni tetracordi inferiori o della modulazione, i metodi di classificazione possono variare in alcuni dettagli.

2. Non c'è una coerenza perfetta, talvolta perfino nello stesso testo, su come specificare le armature di chiave per certi maqam. È tuttavia pratica comune l'utilizzo di una armatura di chiave per gruppo, invece di un'armatura diversa per ogni maqam.

   I metodi dei seguenti autori per l'*Oud*, il liuto arabo, contegono esempi di composizioni in gran parte turche e arabe.

   - Charbel Rouhana
   - George Farah
   - Ibrahim Ali Darwish Al-masri

### 2.10.3 Musica classica turca

Questa sezione evidenzia le questioni rilevanti per la notazione di musica classica turca.

#### Riferimenti per la musica classica turca

La musica classica turca si è sviluppata nell'impero ottomano in un periodo più o meno contemporaneo a quello della musica classica in Europa, e ha proseguito nei secoli ventesimo e ventunesimo come una tradizione viva e distinta, con le sue forme compositive, la sua teoria e i propri stili di esecuzione. Tra le sue principali peculiarità c'è l'uso degli intervalli microtonali basati sui 'commi' di 1/9 di un tono, dal quale vengono costruite le forme melodiche note come *makam* (plurale *makamlar*).

Alcune questioni rilevanti per la musica classica turca sono trattate in altre sezioni della documentazione:

- I nomi per le note speciali e le alterazioni sono spiegati in Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

#### Nomi delle note in turco

Nella musica classica turca le altezze tradizionalmente hanno nomi unici, e a causa della divisione del tono in nove parti i makamlar usano un insieme di intervalli completamente diverso dalle scale e dai modi occidentali: *koma* (1/9 di un tono), *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (un tono intero) e *artık ikili* (12/9 o 13/9 di un tono).

Dal punto di vista della notazione moderna conviene usare le note standard occidentali sul rigo (do, re, mi, ...) con delle alterazioni speciali che alzano o abbassano le note di intervalli di 1/9, 4/9, 5/9 e 8/9 di tono. Queste alterazioni sono definite nel file `makam.ly`.

La tabella seguente elenca:

- il nome di queste alterazioni speciali,
- il suffisso di alterazione da aggiungere alle note,
- e l'alterazione di altezza rappresentata come una frazione dell'intero tono.

| **Nome alterazione** | **suffissp** | **alterazione** |
|---|---|---|
| büyük mücenneb (diesis) | -bm | +8/9 |
| kücük mücenneb (diesis) | -k | +5/9 |
| bakiye (sharp) | -b | +4/9 |
| koma (sharp) | -c | +1/9 |

| | | | |
|---|---|---|---|
| koma (flat) | | -fc | -1/9 |
| bakiye (flat) | | -fb | -4/9 |
| kücük (bemolle) | mücenneb | -fk | -5/9 |
| büyük (bemolle) | mücenneb | -fbm | -8/9 |

Per una spiegazione più generale della notazione musicale non occidentale, leggere Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

## Vedi anche

Glossario musicale: Sezione “makam” in *Glossario Musicale*, Sezione “makamlar” in *Glossario Musicale*.

Guida alla notazione: Sezione 2.10.1 [Notazione comune per la musica non occidentale], pagina 472.

# 3 Input e output

Questa sezione tratta le questioni generali relative all'input e all'output di LilyPond, non specifiche di un certo tipo di notazione.

## 3.1 Struttura dell'input

Il principale formato di input di LilyPond sono i file di testo. Per convenzione, questi file hanno estensione `.ly`.

### 3.1.1 Struttura di una partitura

Un blocco `\score` deve contenere una singola espressione musicale delimitata da parentesi graffe:
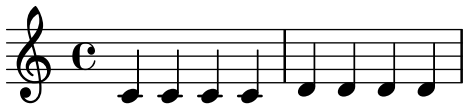
```
\score {
  ...
}
```

> **Nota:** Ci deve essere **solo una** espressione musicale più esterna in un blocco `\score` e **deve** essere racchiusa tra parentesi graffe.

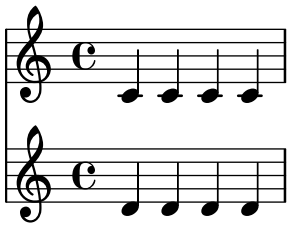
Questa espressione musicale singola può essere di qualsiasi dimensione e contenere altre espressioni musicali di qualsiasi complessità. Tutti gli esempi seguenti sono espressioni musicali:

```
{ 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 { \flauto }
      \new Staff { \oboe }
    >>
    \new StaffGroup <<
      \new Staff { \violinoI }
```

```
        \new Staff { \violinoII }
      >>
    >>
  }
```

I commenti sono un'eccezione a questa regola generale (altre eccezioni sono spiegate in Sezione 3.1.5 [Struttura del file], pagina 484). Sia i commenti su una singola riga che quelli multirigo delimitati da `%{ ... %}` possono essere inseriti ovunque nel file di input: dentro o fuori un blocco `\score` e dentro o fuori la singola espressione musicale di un blocco `\score`.

È bene ricordare che anche se un file contiene soltanto un blocco `\score`, questo è implicitamente racchiuso in un blocco \book. Un blocco \book in un file sorgente produce almeno un file di output e il nome predefinito del file di output deriva dal nome del file di input, quindi `fandangoperelefanti.ly` genererà `fandangoperelefanti.pdf`.

Maggiori dettagli sui blocchi `\book` si trovano in Sezione 3.1.2 [Molteplici partiture in un libro], pagina 481, Sezione 3.1.3 [Molteplici file di output da un unico file di input], pagina 482, e Sezione 3.1.5 [Struttura del file], pagina 484.

### Vedi anche

Manuale di apprendimento: Sezione "Lavorare sui file di input" in *Manuale di Apprendimento*, Sezione "Espressioni musicali" in *Manuale di Apprendimento*, Sezione "La partitura è una (singola) espressione musicale composta" in *Manuale di Apprendimento*.

## 3.1.2 Molteplici partiture in un libro

Un documento può contenere più brani di musica e testo, come, per esempio, uno studio o una parte orchestrale con vari movimenti. Ogni movimento si inserisce con un blocco `\score`,

```
\score {
  ...musica...
}
```

e il testo si inserisce con un blocco `\markup`,

```
\markup {
  ...testo...
}
```

Tutti i movimenti e i testi che appaiono nello stesso file `.ly` normalmente vengono elaborati in un singolo file di output.

```
\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}
```

Un'importante eccezione è costituita dai documenti da elaborare con lilypond-book, dove occorre aggiungere esplicitamente un blocco `\book`, altrimenti apparirà nell'output solo il primo blocco `\score` o `\markup`.

L'intestazione di ogni brano musicale può essere inserita nel blocco `\score`. Il nome definito nel campo `piece` (brano) dell'intestazione apparirà all'inizio di ogni movimento. Il titolo dell'intero libro può trovarsi all'interno del blocco `\book` oppure, se questo non è presente, nel blocco `\header` all'inizio del file.

```
\header {
```

```
  title = "Otto miniature"
  composer = "Igor Stravinsky"
}
\score {
  \header { piece = "Romanza" }
  ...
}
\markup {
   ...testo della seconda strofa...
}
\markup {
   ...testo della terza strofa...
}
\score {
  \header { piece = "Minuetto" }
  ...
}
```

I brani musicali possono essere raggruppati in parti di libro tramite i blocchi `\bookpart`. Le parti di libro sono separate da un'interruzione di pagina e possono iniziare con un titolo, come il libro stesso, specificandolo in un blocco `\header`.

```
\bookpart {
  \header {
    title = "Titolo del libro"
    subtitle = "Prima parte"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Seconda parte"
  }
  \score { ... }
  ...
}
```

### 3.1.3 Molteplici file di output da un unico file di input

Per generare molteplici file di output dallo stesso file `.ly`, basta aggiungere molteplici blocchi `\book`, ognuno dei quali produrrà un file di output separato. Se non è specificato alcun blocco `\book` nel file di input, LilyPond tratterà implicitamente l'intero file come un singolo blocco `\book`, come è spiegato in Sezione 3.1.5 [Struttura del file], pagina 484.

Nel generare molteplici file da un singolo file sorgente, LilyPond controlla che nessuno dei file di output di alcun blocco `\book` sovrascriva il file di output prodotto da un blocco `\book` precedente dello stesso file di input.

Per farlo, aggiunge un suffisso al nome del file di output di ogni blocco `\book`, derivato dal nome del file di input (se viene lasciata l'impostazione predefinita).

Il comportamento predefinito consiste quindi nell'appendere un suffisso numerico a ogni nome che potrebbe entrare in conflitto, quindi

```
\book {
  \score { ... }
```

```
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
```

nel file sorgente `ottominiature.ly` genererà

- `ottominiature.pdf`,
- `ottominiature-1.pdf` e
- `ottominiature-2.pdf`.

### 3.1.4 Nomi dei file di output

LilyPond permette di decidere quali nomi di file debbano essere usati dai vari backend quando questi generano i file di output.

Nella sezione precedente abbiamo visto come LilyPond prevenga i conflitti di nome quando genera molti file di output da un singolo file sorgente. È possibile anche specificare un proprio suffisso per ogni blocco `\book`. Quindi si possono, per esempio, generare file chiamati `ottominiature-Romanza.pdf`, `ottominiature-Minuetto.pdf` e `ottominiature-Notturno.pdf` aggiungendo una dichiarazione `\bookOutputSuffix` all'interno di ogni blocco `\book`.

```
\book {
  \bookOutputSuffix "Romanza"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Minuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Notturno"
  \score { ... }
  \paper { ... }
}
```

È possibile anche specificare un diverso nome del file di output per ciascun blocco `book`, tramite le dichiarazioni `\bookOutputName`

```
\book {
  \bookOutputName "Romanza"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Minuetto"
  \score { ... }
  \paper { ... }
```

```
}
\book {
  \bookOutputName "Notturno"
  \score { ... }
  \paper { ... }
}
```

Questo file produrrà i seguenti file di output:

- `Romanza.pdf`,
- `Minuetto.pdf` e
- `Notturno.pdf`.

### 3.1.5 Struttura del file

Un file `.ly` può contenere un qualsiasi numero di espressioni di livello superiore (in inglese, *toplevel expressions*). Per espressione di livello superiore si intende una delle seguenti:

- Una definizione di output, come `\paper`, `\midi` e `\layout`. Tale definizione, se posta nel livello superiore, cambia le impostazioni predefinite al livello del libro. Se più di una di queste definizioni viene inserita nel livello superiore, le definizioni vengono combinate, ma in caso di conflitto hanno precedenza le definizioni più recenti. Per sapere con precisione come ciò influisca sul blocco `\layout`, leggere Sezione 4.2.1 [Il blocco layout], pagina 550.
- Un'espressione Scheme diretta, come `#(set-default-paper-size "a7" 'landscape)` o `#(ly:set-option 'point-and-click #f)`.
- Un blocco `\header`. Se all'inizio del file, imposta il blocco dell'intestazione globale. Questo è il blocco che contiene le impostazioni predefinite dei campi dei titoli come compositore, titolo, etc. per tutti i libri del file (vedi [Come funzionano i titoli], pagina 486).
- Un blocco `\score`. Questa partitura e altre eventuali partiture di livello superiore saranno combinate insieme in un singolo blocco `\book`. Tale comportamento può essere modificato impostando la variabile `toplevel-score-handler` nel livello superiore. Il gestore (in inglese *handler*) predefinito è definito nel file `../scm/lily-library.scm` e viene impostato nel file `../ly/declarations-init.ly`.
- Un blocco `\book` combina logicamente molteplici movimenti (ovvero molteplici blocchi `\score`) in un documento. Se ci sono vari blocchi `\score`, verrà creato un file di output per ogni blocco `\book`, in cui saranno concatenati tutti i movimenti corrispondenti. Ha senso specificare esplicitamente i blocchi `\book` in un file `.ly` solo se si desidera creare vari file di output da un solo file di input. Un'eccezione è data dai documenti lilypond-book, dove bisogna aggiungere esplicitamente un blocco `\book` se si vuole più di un singolo blocco `\score` o `\markup` nello stesso esempio. Tale comportamento può essere modificato impostando la variabile `toplevel-book-handler` nel livello superiore. Il gestore predefinito è definito nel file di inizializzazione `../scm/lily.scm`.
- Un blocco `\bookpart`. Un libro può essere suddiviso in varie parti, tramite blocchi `\bookpart`, per semplificare le interruzioni di pagina o per usare impostazioni `\paper` diverse nelle varie parti.
- Un'espressione musicale composta, come

  ```
  { c'4 d' e'2 }
  ```

  pone il brano in un blocco `\score` e lo formatta in un unico libro insieme a tutti gli altri blocchi `\score` e espressioni musicali di livello superiore. In altre parole, un file che contiene soltanto l'espressione musicale precedente verrà trasformato in

  ```
  \book {
    \score {
      \new Staff {
  ```

```
        \new Voice {
          { c'4 d' e'2 }
        }
      }
      \layout { }
    }
    \paper { }
    \header { }
  }
```

Tale comportamento può essere modificato impostando la variabile `toplevel-music-handler` nel livello superiore. Il gestore predefinito è definito nel file di inizializzazione `../scm/lily.scm`.

- Un testo, per esempio una strofa

```
\markup {
   2.  La prima riga della seconda strofa.
}
```

I testi possono trovarsi sopra, sotto o in mezzo alle partiture o espressioni musicali, ovunque esse appaiano.

- Una variabile, come

```
foo = { c4 d e d }
```

può essere utilizzata in un punto successivo del file scrivendo `\foo`. Il nome di una variabile deve avere solo caratteri alfabetici; nessun numero, trattino o trattino basso.

L'esempio seguente mostra tre elementi che possono essere inseriti nel livello superiore

```
\layout {
  % Non giustificare l'output
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

Ciascuna delle seguenti istruzioni lessicali può essere inserita in qualsiasi punto di un file:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- Un commento su riga singola, introdotto da un segno `%`.
- Un commento multirigo delimitato da `%{ ... %}`.

Lo spazio bianco tra gli elementi dell'input viene generalmente ignorato e può essere liberamente omesso o aumentato per migliorare la leggibilità. Tuttavia esistono dei casi in cui lo spazio bianco deve essere sempre usato per non incorrere in errori:

- Intorno ad ogni parentesi graffa di apertura e di chiusura.
- Dopo ogni comando o variabile, ovvero qualsiasi elemento che inizi con un segno `\`.
- Dopo ogni elemento che debba essere interpretato come un'espressione Scheme, ovvero ogni elemento che inizi con un segno `#`.

- Per separare tutti gli elementi di un'espressione Scheme.
- Nella modalità `lyricmode` prima e dopo i comandi `\set` e `\override`.

### Vedi anche

Manuale di apprendimento: Sezione "Come funzionano i file di input di LilyPond" in *Manuale di Apprendimento*.

Guida alla notazione: [Come funzionano i titoli], pagina 486, Sezione 4.2.1 [Il blocco layout], pagina 550.

## 3.2 Titoli e intestazioni

Quasi tutte le partiture musicali hanno il titolo e il nome del compositore e alcuni brani comprendono molte altre informazioni.

### 3.2.1 Creazione di titoli intestazioni e piè di pagina

### Come funzionano i titoli

Ogni blocco `\book` in un singolo file di input produce un diverso file di output, vedi Sezione 3.1.5 [Struttura del file], pagina 484. In ciascun file di output sono disponibili tre tipi di titolazioni: *titoli del libro* all'inizio di ogni libro (*book*), *titoli della parte* all'inizio di ciascuna parte (*bookpart*) e *titoli del brano* all'inizio di ciascun brano (*score*).

I valori dei campi dei titoli come `title` (titolo) e `composer` (compositore) sono definiti nei blocchi `\header` (la sintassi dei blocchi `\header` e un elenco completo dei campi disponibili si trovano in [Formattazione predefinita dei titoli delle parti e dei brani], pagina 489). I titoli del libro, delle parti e dei brani possono avere tutti gli stessi campi, sebbene per impostazione predefinita i campi dei titoli del brano siano limitati a `piece` e `opus`.

I blocchi `\header` possono essere inseriti in quattro diversi punti formando una gerarchia discendente di blocchi `\header`:

- All'inizio del file di input, prima di tutti i blocchi `\book`, `\bookpart` e `\score`.
- All'interno di un blocco `\book` ma fuori da tutti i blocchi `\bookpart` e `\score` compresi in quel libro.
- All'interno di un blocco `\bookpart` ma fuori da tutti i blocchi `\score` compresi in quella parte.
- In un blocco `\score`.

I valori dei campi vengono filtrati attraverso questa gerarchia, con i valori più in alto nella gerarchia che persistono finché un valore più in basso nella gerarchia non ha la precedenza. In sintesi:

- Il titolo di un libro deriva dai campi impostati all'inizio del file di input, modificati dai campi definiti nel blocco `\book`. I valori risultanti vengono usati per stampare il titolo di quel libro, purché ci sia altro materiale che generi una pagina all'inizio del libro, prima della prima parte. Una singola interruzione di pagina (`\pageBreak`) è sufficiente.
- Il titolo di una parte deriva dai campi impostati all'inizio del file di input, modificati dai campi definiti nel blocco `\book` e poi da quelli definiti nel blocco `\bookpart`. I valori risultanti vengono usati per stampare il titolo di quella parte del libro.
- Il titolo di un brano deriva dai campi all'inizio del file di input, modificati dai campi definiti nel blocco `\book` e poi da quelli definiti nel blocco `\bookpart` e infine da quelli definiti nel blocco `\score`. I valori risultanti vengono usati per stampare il titolo di quel brano. Nota bene: per impostazione predefinita nei titoli del brano appaiono soltanto i campi `piece` e `opus` a meno che la variabile `print-all-headers` di `\paper` non sia impostata su `#t`.

Non è necessario inserire blocchi `\header` in tutti e quattro i luoghi: alcuni o perfino tutti possono essere omessi. Analogalmente, in semplici file di input si possono omettere i blocchi `\book` e `\bookpart`, lasciando che questi siano creati implicitamente.

Se il libro ha un solo brano, il blocco `\header` viene di solito messo all'inizio del file, in modo che sia prodotto soltanto il titolo della parte e lasciando disponibili tutti i campi di titolazione.

Se il libro ha vari brani, è possibile usare diverse disposizioni dei blocchi `\header`, corrispondenti ai diversi tipi di pubblicazione musicale. Per esempio, se la pubblicazione contiene vari brani dello stesso compositore, la soluzione più adatta prevederebbe un blocco `\header` all'inizio del file che specifichi il titolo del libro e il nome del compositore, e dei blocchi `\header` in ciascun blocco `\score` che specifichino il titolo del brano (`piece`) e dell'opera (`opus`), come in questo esempio:

```
\header {
  title = "SUITE I."
  composer = "J. S. Bach."
}
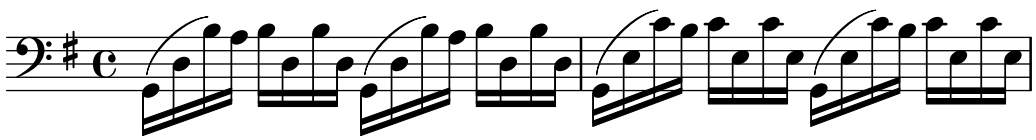
\score {
  \header {
    piece = "Prélude."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \repeat unfold 2 { g,16( d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16( e' c') b c e, c' e, } |
  }
}

\score {
  \header {
    piece = "Allemande."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b(c d e) fis( g a b) |
  }
}
```
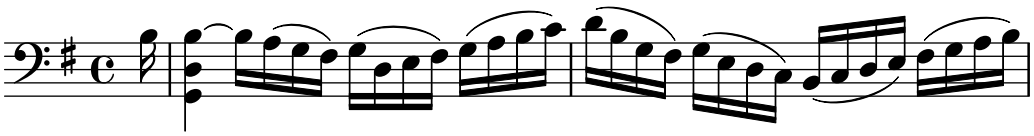
# SUITE I.

J. S. Bach.

Prélude.



**Allemande.**



Sono possibili disposizioni più complesse. Per esempio, i campi testuali del blocco `\header` di un libro possono essere stampati nei titoli di tutti i brani, magari sovrascrivendo alcuni campi e sopprimendone altri:

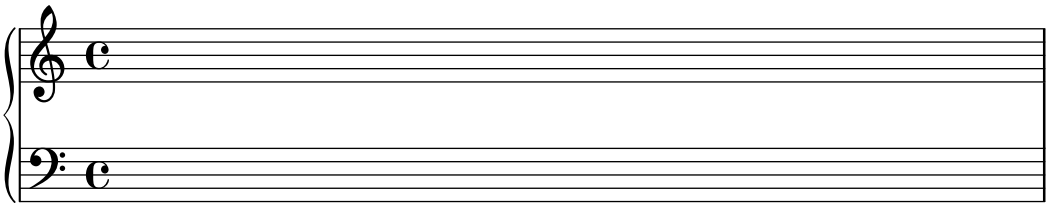
```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Non mostrare il piè di pagina predefinito nell'ultima pagina di questo libro
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Non mostrare il sottotitolo in questo brano
      subtitle = ##f
    }
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
  \score {
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOCI"
      opus = "BWV 846"
      % Non mostrare il sottotitolo in questo brano
      subtitle = ##f
    }
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
}
```

# DAS WOHLTEMPERIRTE CLAVIER

### TEIL I

## PRAELUDIUM I

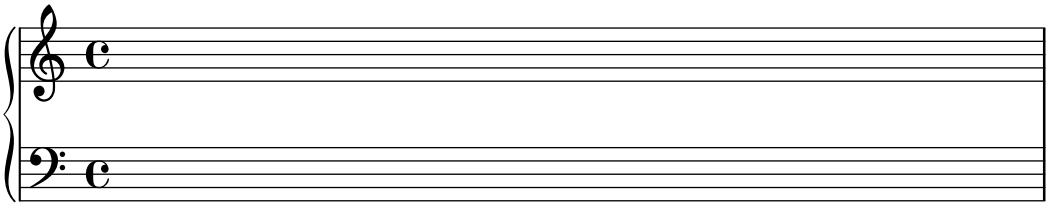
BWV 846



## FUGA I

### A 4 VOCI

BWV 846



### Vedi anche
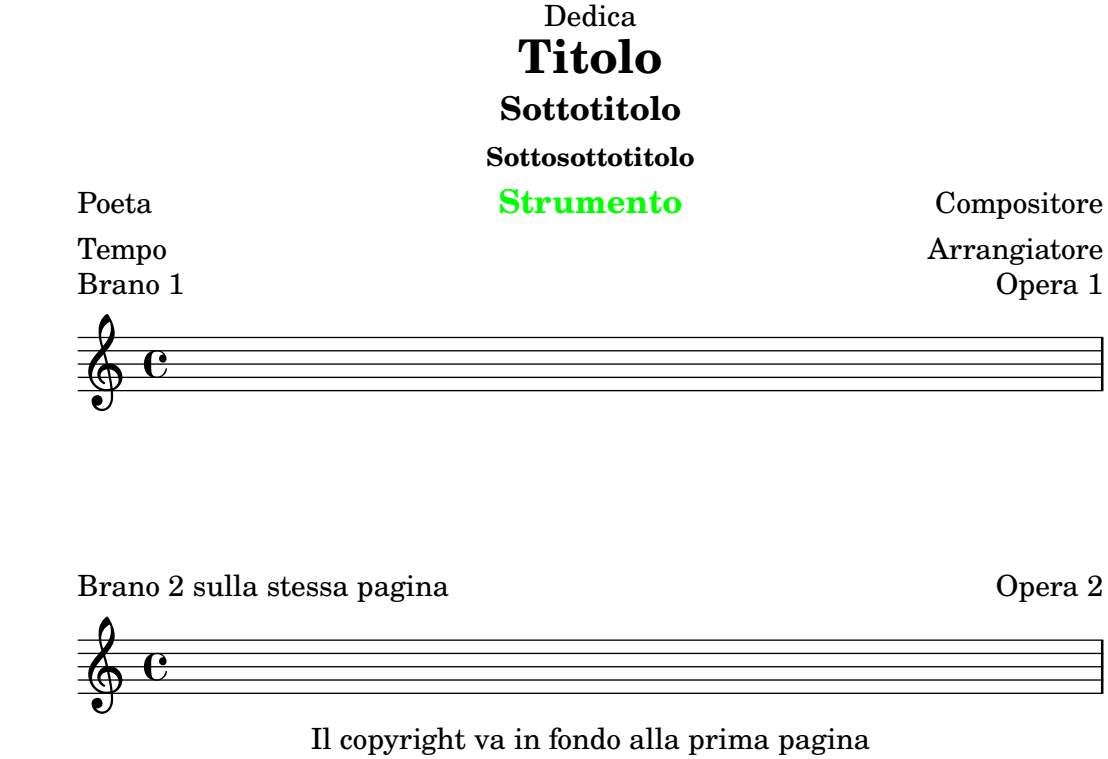
Guida alla notazione: Sezione 3.1.5 [Struttura del file], pagina 484, [Formattazione predefinita dei titoli delle parti e dei brani], pagina 489, [Formattazione personalizzata dei titoli], pagina 494.

### Formattazione predefinita dei titoli delle parti e dei brani

Questo esempio illustra visivamente tutte le variabili del blocco `\header`:

```
\book {
  \header {
      % I seguenti campi sono centrati
    dedication = "Dedica"
    title = "Titolo"
    subtitle = "Sottotitolo"
    subsubtitle = "Sottosottotitolo"
      % I seguenti campi sono distribuiti uniformemente su una riga
      % il campo "instrument" appare anche nelle pagine seguenti
    instrument = \markup \with-color #green "Strumento"
    poet = "Poeta"
    composer = "Compositore"
      % I campi seguenti sono posti agli estremi opposti della stessa riga
    meter = "Tempo"
    arranger = "Arrangiatore"
      % I campi seguenti sono centrati in fondo
    tagline = "Lo slogan va in fondo all'ultima pagina"
    copyright = "Il copyright va in fondo alla prima pagina"
  }
  \score {
```

```
    \header {
        % I campi seguenti sono posti agli estremi opposti della stessa riga
      piece = "Brano 1"
      opus = "Opera 1"
    }
    { s1 }
  }
  \score {
    \header {
        % I campi seguenti sono posti agli estremi opposti della stessa riga
      piece = "Brano 2 sulla stessa pagina"
      opus = "Opera 2"
    }
    { s1 }
  }
  \pageBreak
  \score {
    \header {
        % I campi seguenti sono posti agli estremi opposti della stessa riga
      piece = "Brano 3 su una nuova pagina"
      opus = "Opera 3"
    }
    { s1 }
  }
}
```






Lo slogan va in fondo all'ultima pagina

Notare che:

- Il nome dello strumento sarà ripetuto su ogni pagina.
- Appaiono soltanto i campi `piece` e `opus` di un blocco `\score` quando la variabile `print-all-headers` del foglio è impostata su `##f` (valore predefinito).
- I campi testuali non specificati in un blocco `\header` sono sostituiti con `\null` in modo da non sprecare spazio.
- Le impostazioni predefinite per `scoreTitleMarkup` posizionano i campi `piece` e `opus` alle estremità opposte della stessa riga.

Per cambiare la formattazione predefinita leggere [Formattazione personalizzata dei titoli], pagina 494.

Se un blocco `\book` inizia subito con un blocco `\bookpart`, non verrà stampato alcun titolo per il libro, dato che non esiste una pagina in cui farlo apparire. Se il titolo del libro deve comunque apparire, conviene inserire all'inizio di `\book` del testo inserito con `\markup` oppure un comando `\pageBreak`.

Usare la variabile `breakbefore` all'interno di un blocco `\header` racchiuso in un blocco `\score` per far sì che i titoli del blocco `\header` di più alto livello appaiano da soli nella prima pagina, mentre la musica (definita nel blocco `\score`) inizia nella pagina successiva.
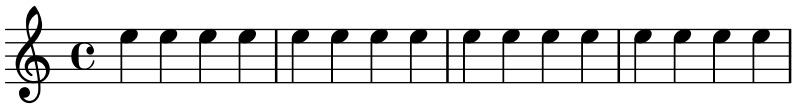
```
\book {
  \header {
    title = "Titolo del libro"
    subtitle = "Sottotitolo del libro"
    copyright = "Fine della prima pagina"
  }
  \score {
    \header {
      piece = "Titolo del brano"
      breakbefore = ##t
    }
    \repeat unfold 4 { e'' e'' e'' e'' }
  }
```

```
}
```

# Titolo del libro
## Sottotitolo del libro

Fine della prima pagina

2
Titolo del brano



Music engraving by LilyPond 2.21.82—www.lilypond.org

### Vedi anche

Manuale di apprendimento: Sezione “Come funzionano i file di input di LilyPond” in *Manuale di Apprendimento*,

Guida alla notazione: [Formattazione personalizzata dei titoli], pagina 494, Sezione 3.1.5 [Struttura del file], pagina 484.

File installati: `ly/titling-init.ly`.

### Formattazione predefinita delle intestazioni e dei piè di pagina

Le *intestazioni* e i *piè di pagina* sono linee di testo che appaiono in cima e in fondo alle pagine, distinte dal testo principale di un libro. Possono essere definite nelle seguenti variabili del blocco `\paper`:

- `oddHeaderMarkup`
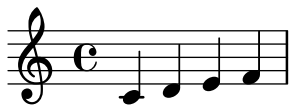
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

Queste variabili di markup possono soltanto accedere ai campi testuali definiti nei blocchi `\header` del livello superiore (che vengono applicati a tutti i brani del libro) e sono definiti in `ly/titling-init.ly`. Impostazioni predefinite:

- i numeri di pagina sono collocati automaticamente in alto a sinistra (se la pagina è pari) o in alto a destra (se la pagina è dispari), a partire dalla seconda pagina.
- il campo `instrument` viene ripetuto al centro di ogni pagina, a partire dalla seconda pagina.
- il testo del `copyright` è centrato in fondo alla prima pagina.
- lo "slogan" (o firma) – `tagline` – è centrato in fondo all'ultima pagina o sotto il campo del `copyright` se c'è una sola pagina.

Il testo del piè di pagina predefinito per l'ultima pagina può essere modificato aggiungendo il campo `tagline` al blocco `\header` del livello superiore.

```
\book {
  \header {
    tagline = "... notazione musicale per tutti"
  }
  \score {
    \relative {
      c'4 d e f
    }
  }
}
```



... notazione musicale per tutti

Per toglierlo, impostare `tagline` su `##f`.

### 3.2.2 Titoli intestazioni e piè di pagina personalizzati

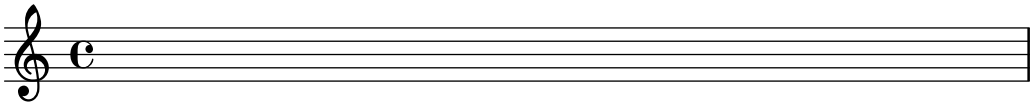
#### Titoli personalizzati

Si possono usare i normali comandi `\markup` per personalizzare qualsiasi intestazione, piè di pagina e titolo di un blocco `\header`.

```
\score {
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
  { s1 }
```

```
}
```

**PRAELUDIUM I** *BWV 846*



### Vedi anche

Guida alla notazione: Sezione 1.8.2 [Formattazione del testo], pagina 246.

### Formattazione personalizzata dei titoli

I comandi `\markup` nel blocco `\header` sono utili solo per la formattazione del testo, ma non consentono un controllo preciso sul posizionamento dei titoli. Per personalizzare il posizionamento dei campi testuali, cambiare una o entrambe le seguenti variabili `\paper`:

- `bookTitleMarkup`
- `scoreTitleMarkup`

Il posizionamento dei titoli secondo i valori predefiniti di queste variabili `\markup` è mostrato negli esempi in [Formattazione predefinita dei titoli delle parti e dei brani], pagina 489.

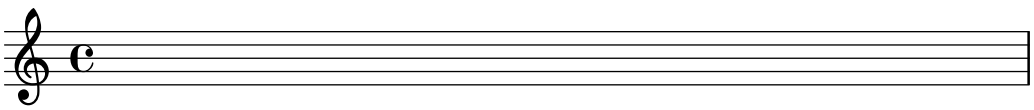
Le impostazioni predefinite di `scoreTitleMarkup`, definite in `ly/titling-init.ly`, sono:

```
scoreTitleMarkup = \markup { \column {
  \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
}
```

Questo pone i campi testuali `piece` e `opus` alle estremità opposte della stessa riga:

```
\score {
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
  { s1 }
}
```

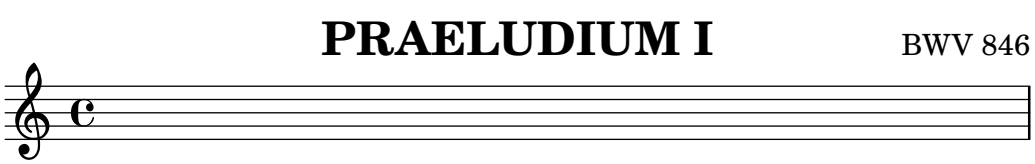
PRAELUDIUM I BWV 846



L'esempio seguente ridefinisce `scoreTitleMarkup` in modo che il campo testuale di `piece` sia centrato e in un tipo di carattere più grande e in grassetto.

```
\book {
  \paper {
    indent = 0\mm
```

```
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "PRAELUDIUM I"
      opus = "BWV 846"
    }
    { s1 }
  }
}
```

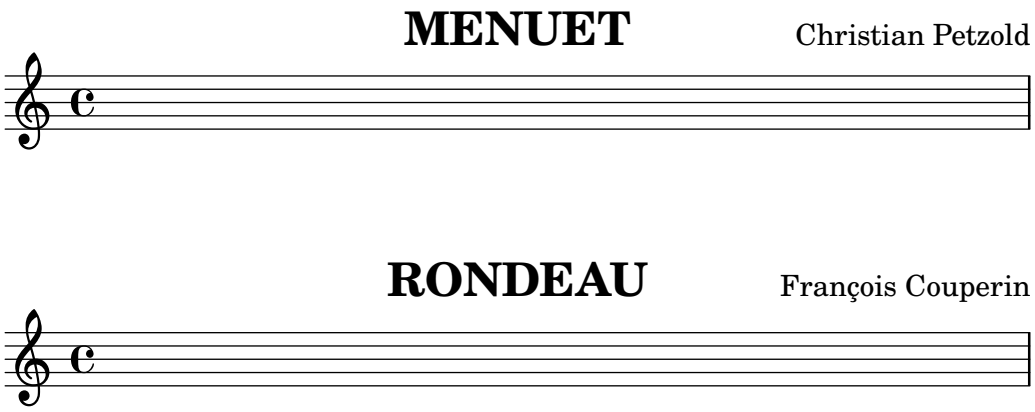



I campi testuali che non sono normalmente attivi nei blocchi `\header` compresi in un blocco `\score` possono essere stampati nello spazio dedicato al titolo del brano aggiungendo `print-all-headers` nel blocco `\paper`. Lo svantaggio di questo metodo è che i campi testuali intesi esclusivamente per lo spazio del titolo della parte devono essere manualmente soppressi in ogni blocco `\score`. Vedi [Come funzionano i titoli], pagina 486.

Per evitare ciò, è meglio mettere il campo testuale desiderato nella definizione di `scoreTitleMarkup`. Nell'esempio seguente il campo `composer` (solitamente associato a `bookTitleMarkup`) viene aggiunto a `scoreTitleMarkup`, facendo sì che ogni brano possa elencare un diverso compositore:
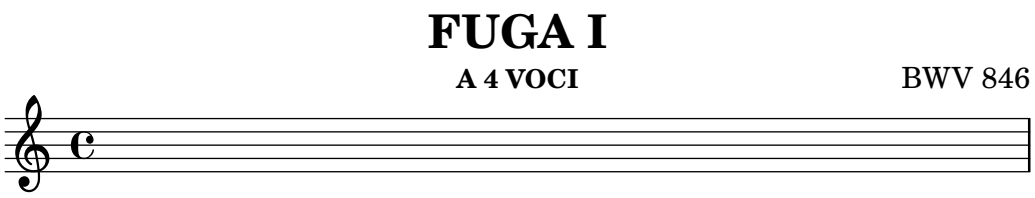
```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:composer
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "MENUET"
      composer = "Christian Petzold"
    }
    { s1 }
  }
```

```
  \score {
    \header {
      piece = "RONDEAU"
      composer = "François Couperin"
    }
    { s1 }
  }
}
```




È anche possibile creare un campo testuale personalizzato e fare riferimento ad esso nella definizione di markup.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP)
        \dir-column {
          \center-align \fontsize #-1 \bold
            \fromproperty #'header:mycustomtext %% Campo definito dall'utente
          \center-align \fontsize #4 \bold
            \fromproperty #'header:piece
        }
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "FUGA I"
      mycustomtext = "A 4 VOCI" %% Campo definito dall'utente
      opus = "BWV 846"
    }
    { s1 }
  }
}
```




## Vedi anche

Guida alla notazione: [Come funzionano i titoli], pagina 486.

## Formattazione personalizzata di intestazioni e piè di pagina

I comandi `\markup` nel blocco `\header` sono utili solo per la formattazione del testo, ma non consentono un controllo preciso sul posizionamento di intestazioni e piè di pagina. Per personalizzare il posizionamento dei campi testuali, usare una o entrambe le seguenti variabili `\paper`:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

Il comando `\on-the-fly` – usato all'interno di un blocco `\markup` – permette di aggiungere del testo a intestazioni e piè di pagina definiti nel blocco `\paper`, solo se certe condizioni sono soddisfatte, tramite la seguente sintassi:

```
variabile = \markup {
  ...
  \on-the-fly  \procedura  testo
  ...
}
```

La *procedura* viene chiamata ogni volta che viene elaborato il comando `\markup` nel quale essa si trova. La *procedura* verifica una precisa condizione e interpreta (ovvero stampa) l'argomento *testo* se e solo se la condizione è vera.

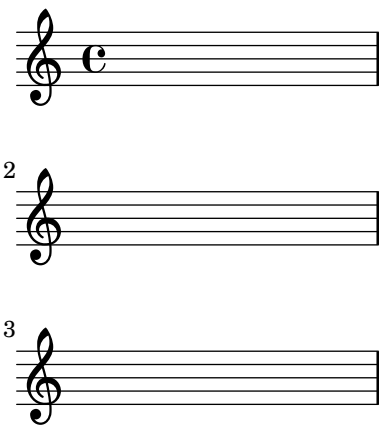
Sono disponibili varie procedure pronte per verificare varie condizioni:

| **Nome della procedura** | **Condizione verificata** |
|---|---|
| print-page-number-check-first | stampare questa pagina? |
| create-page-number-stencil | print-page-numbers è vero? |
| print-all-headers | print-all-headers è vero? |
| first-page | prima pagina del libro? |
| not-first-page | non la prima pagina del libro? |
| (on-page nmbr) | numero pagina = nmbr? |
| last-page | ultima pagina del libero? |
| part-first-page | prima pagina della parte? |
| not-part-first-page | non la prima pagine della parte? |
| part-last-page | ultima pagina della parte? |
| not-single-page | pagine della parte > 1? |

L'esempio seguente centra i numeri di pagina in fondo a ogni pagina. Innanzitutto vengono annullate le impostazioni predefinite per `oddHeaderMarkup` e `evenHeaderMarkup` definendo ciascuno di essi come un markup *null*. Poi `oddFooterMarkup` viene ridefinito col numero di pagina centrato. Infine a `evenFooterMarkup` viene assegnata la stessa formattazione di `\oddFooterMarkup`:

```
\book {
  \paper {
```

```
    print-page-number = ##t
    print-first-page-number = ##t
    oddHeaderMarkup = \markup \null
    evenHeaderMarkup = \markup \null
    oddFooterMarkup = \markup {
      \fill-line {
        \on-the-fly \print-page-number-check-first
        \fromproperty #'page:page-number-string
      }
    }
    evenFooterMarkup = \oddFooterMarkup
  }
  \score {
    \new Staff { s1 \break s1 \break s1 }
  }
}
```




1

Varie condizioni `\on-the-fly` possono essere combinate insieme come se si utilizzasse l'operatore logico 'AND' e il testo apparirà solo se tutte le condizioni sono vere. Per esempio, queste due condizioni

```
\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }
```

verificano se l'output è una pagina singola.

## Vedi anche

Guida alla notazione: [Come funzionano i titoli], pagina 486, [Formattazione predefinita dei titoli delle parti e dei brani], pagina 489.

File installati: `../ly/titling-init.ly`.

### 3.2.3 Creazione di metadati per i file di output

Oltre ad apparire nell'output, le variabili di `\header` vengono usate anche per impostare i metadati per i file di output. Per esempio, con i file PDF, questi metadati possono essere mostrati dai lettori PDF come `proprietà` del file PDF. Per ciascun tipo di file di output, verranno

considerate solo le definizioni `\header` dei blocchi specifici per quel tipo e dei blocchi che si trovano al livello superiore della gerarchia. Dunque per i file PDF solo le definizioni `\header` del livello `\book` e del livello superiore vengono utilizzate per i metadati PDF del documento, mentre per i file MIDI vengono usate le definizioni che si trovano in un livello inferiore o in un blocco `\score`.

Per esempio, impostando la proprietà `title` del blocco `header` su 'Sinfonia I', questo nome verrà usato come titolo del documento PDF e come nome della sequenza del file MIDI.

```
\header {
  title = "Sinfonia I"
}
```

Se si desidera impostare il titolo dell'output su un valore e la proprietà titolo del PDF su un valore diverso, si può usare `pdftitle`, nel modo seguente:

```
\header {
  title = "Sinfonia I"
  pdftitle = "Sinfonia I di Beethoven"
}
```

Le variabili `title`, `subject`, `keywords`, `subtitle`, `composer`, `arranger`, `poet`, `author` e `copyright` impostano anche le proprietà del PDF e possono essere tutte prefissate con 'pdf' per impostare una proprietà del PDF su un valore diverso da quello dell'output.

La proprietà PDF `Creator` è automaticamente impostata su 'LilyPond' più la versione di LilyPond utilizzata, e `CreationDate` (data di creazione) e `ModDate` (data di modifica) sono entrambe impostate sulla data e ora correnti. `ModDate` può essere sovrascritta impostando nel blocco `header` la variabile `moddate` (o `pdfmoddate`) su una valida data per il PDF.

La variabile `title` imposta anche il nome della sequenza dei file MIDI. La variabile `miditítle` serve a impostare il nome della sequenza in modo indipendente dal valore usato per l'output.

### 3.2.4 Creazione di note a piè di pagina

Le note a piè di pagina possono essere usate in situazioni diverse. In tutti i casi, un 'segno della nota a piè di pagina' viene inserito come riferimento vicino al testo o alla musica a cui si riferisce e il corrispondente 'testo della nota a piè di pagina' appare in fondo alla stessa pagina.

Le note a piè di pagina si creano diversamente a seconda che siano applicate a espressioni musicali o a del testo separato e fuori dalle espressioni musicali.

#### Note a piè di pagina nelle espressioni musicali

##### *Panoramica sulle note a piè di pagina attaccate alla musica*

Le note a piè di pagina nelle espressioni musicali appartengono a due categorie:

*Note a piè di pagina basate su un evento*
: sono collegate a un preciso evento. Esempi di tali eventi sono note singole, articolazioni (come le indicazioni di diteggiatura, gli accenti, le dinamiche), e ciò che è successivo a un evento (come le legature di portamento e le travature manuali). La forma generale per le note a piè di pagina basate su un evento è la seguente:

  `[direzione] \footnote [segno] offset nota musica`

*Note a piè di pagina basate sul tempo*
: sono collegate a un preciso momento temporale in un contesto musicale. Alcuni comandi come `\time` e `\clef` non usano in realtà degli eventi per creare oggetti come le indicazioni di tempo e le chiavi. E nemmeno un accordo crea un suo evento: il suo gambo o coda sono creati alla fine di un'unità di tempo (attraverso un evento di una delle note al suo interno). Non è definito esattamente quale dei molteplici

eventi nota di un accordo sarà giudicato la causa ultima di un gambo o di una coda. Dunque per legare una nota a questi, sono preferibili le note a piè di pagina basate sul tempo.

Una nota a piè di pagina basata sul tempo permette a tali oggetti della formattazione di avere delle note senza che sia necessario riferirsi a un evento. La forma generale per le note a piè di pagina basate sul tempo è la seguente:

```
\footnote [segno] offset nota [Contesto].NomeGrob
```

Gli elementi di entrambe le forme sono:

*direzione* se (e solo se) `\footnote` viene applicato a un post-evento o a una articolazione, deve essere preceduto da un indicatore di direzione (`-`, `_`, `^`) per poter collegare la *musica* (col segno della nota a piè di pagina) alla nota o pausa precedenti.

*segno* è del testo – racchiuso tra virgolette o in un blocco `\markup` – che specifica il segno della nota usata per contrassegnare sia il punto di riferimento che la nota stessa in fondo alla pagina. Può essere omesso (o sostituito con `\default`), nel qual caso sarà generato automaticamente un numero in sequenza. Tali sequenze numeriche ricominciano su ogni pagina contenente una nota.

*offset* è una coppia di numeri come '`#(2 . 1)`' che specificano la distanza orizzontale e verticale (X e Y offset), in unità di spazio rigo, dal bordo dell'oggetto in cui il segno deve essere posizionato. Valori positivi degli offset vengono presi dal bordo in alto a destra, valori negativi dal bordo in basso a sinistra e lo zero significa che il segno è centrato sul bordo.

*Contesto* è il contesto in cui è creato il grob a cui è collegata la nota. Può essere omesso se il grob si trova in un contesto più basso, per esempio un contesto `Voice`.

*NomeGrob*
indica un tipo di grob a cui assegnare la nota (come '`Flag`'). Se viene specificato, la nota a piè di pagina non è collegata a un'espressione musicale in particolare, bensì a tutti i grob di quel tipo che si trovano in quel momento del tempo musicale.

*nota* è il testo – racchiuso tra virgolette o in un blocco `\markup` – che contiene il testo da usare per la nota a piè di pagina.
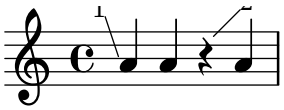
*musica* è l'evento musicale o il post-evento o articolazione a cui viene collegata la nota.

### *Note a piè di pagina basate su un evento*

Una nota a piè di pagina può essere collegata a un oggetto della formattazione direttamente causato dall'evento corrispondente a *musica* con la sintassi:

```
\footnote [segno] offset nota musica
```

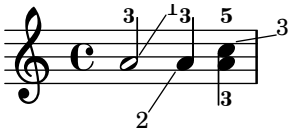
```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "Una nota" a4
    a4
    \footnote #'(2 . 2) "Una pausa" r4
    a4
  }
}
```



[1]Una nota
[2]Una pausa

Contrassegnare un *intero* accordo con una nota a piè di pagina basata su un evento non è possibile: infatti un accordo, perfino uno che contenga una sola nota, non produce un vero evento specifico. Tuttavia possono essere contrassegnate singole note *dentro* l'accordo:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(2 . 3) "Non funziona" <a-3>2
    <\footnote #'(-2 . -3) "Funziona" a-3>4
    <a-3 \footnote #'(3 . 1/2) "Anche questo funziona" c-5>4
  }
}
```

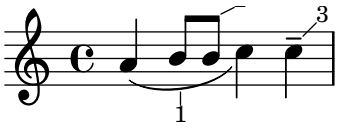


[1]Non funziona
[2]Funziona
[3]Anche questo funziona

Se la nota deve essere attaccata a un post-evento o un'articolazione il comando `\footnote` *deve* essere preceduto da un indicatore di direzione, `-`, `_`, `^`, e seguito dal post-evento o dall'articolazione che si desidera commentare nell'argomento *musica*. In questa forma `\footnote` può essere considerato semplicemente una copia del suo ultimo argomento con un segno di nota a piè di pagina collegato a esso. La sintassi è:

```
direzione \footnote [segno] offset nota musica
```

```
\book {
  \header { tagline = ##f }
  \relative {
    a'4_\footnote #'(0 . -1) "Una legatura di portamento forzata in giù" (
    b8^\footnote #'(1 . 0.5) "Una travatura manuale forzata in su" [
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
}
```

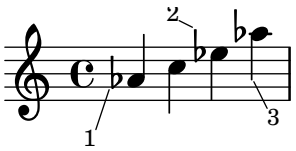


[1]Una legatura di portamento forzata in giù
[2]Una travatura manuale forzata in su
[3]Tenuto

### *Note a piè di pagina basate sul tempo*

Se l'oggetto della formattazione a cui attaccare la nota è *indirettamente* causato da un evento – come un `Accidental` (alterazione) o `Stem` (gambo) causati da un evento `NoteHead` (testa di nota), è necessario specificare il *NomeGrob* dell'oggetto di formattazione al posto di *musica* dopo il testo della nota:
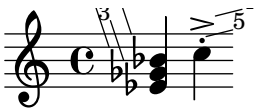
```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . -3) "Un bemolle" Accidental
    aes4 c
    \footnote #'(-1 . 0.5) "Un altro bemolle" Accidental
    ees
    \footnote #'(1 . -2) "Un gambo" Stem
    aes
  }
}
```



[1]Un bemolle
[2]Un altro bemolle
[3]Un gambo

Tuttavia nota che, quando si specifica un NomeGrob, una nota a piè di pagina sarà attaccata a tutti i grob di quel tipo che si trovano in quel momento musicale:
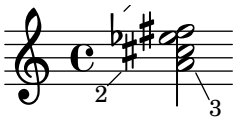
```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "Un bemolle" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Articolazione" Script
    c'->-.
  }
}
```



[1]Un bemolle
[2]Un bemolle
[3]Un bemolle
[4]Articolazione
[5]Articolazione

È possibile assegnare a una nota di un accordo una singola nota a piè di pagina (basata su un evento). '`NoteHead`' è l'unico grob causato direttamente dalla nota di un accordo, dunque una nota a piè di pagina basata su un evento è adatta *soltanto* ad aggiungere una nota a piè di pagina al '`NoteHead`' all'interno di un accordo. Tutti gli altri grob delle note di un accordo sono causati indirettamente. Il comando `\footnote` non ha una sintassi per specificare *sia* un particolare tipo di grob *sia* un particolare evento a cui collegare la nota. Tuttavia si può usare un comando `\footnote` basato sul tempo per specificare il tipo di grob e poi precedere tale comando con `\single` perché venga applicato soltanto all'evento che segue:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    < \footnote #'(1 . -2) "An A" a
      \single \footnote #'(-1 . -1) "Un diesis" Accidental
      cis
      \single \footnote #'(0.5 . 0.5) "Un bemolle" Accidental
      ees fis
    >2
  }
}
```



[1]Un bemolle
[2]Un diesis
[3]An A

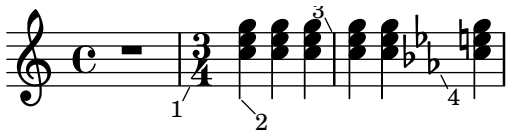
> **Nota:** Quando le note a piè di pagina sono collegate a diversi elementi musicali nello stesso momento musicale, come nell'esempio precedente, le note sono numerate dall'elemento più alto a quello più in basso come questi appaiono nell'output e non nell'ordine in cui sono inseriti nell'input.

Gli oggetti della formattazione come le chiavi e i cambi di armatura di chiave sono causati principalmente da proprietà modificate piuttosto che da veri eventi. Per questo motivo le note su tali oggetti devono essere basate sul loro tempo musicale. Le note a piè di pagina basate sul tempo sono da preferire anche quando si vogliono contrassegnare elementi come i gambi e le

travature in un *accordo*: sebbene tali elementi dell'accordo siano nominalmente assegnati a *un* evento all'interno dell'accordo, affidarsi a una scelta particolare sarebbe imprudente.

L'oggetto della formattazione in questione deve essere sempre specificato esplicitamente nelle note a piè di pagina basate sul tempo, e il contesto appropriato deve essere indicato se il grob viene creato in un contesto diverso da quello più basso.
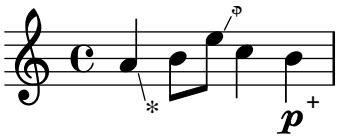
```
\book {
  \header { tagline = ##f }
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Cambio di tempo" Staff.TimeSignature
    \time 3/4
    \footnote #'(1 . -1) "Gambo dell'accordo" Stem
    <c e g>4 q q
    \footnote #'(-0.5 . 1) "Stanghetta" Staff.BarLine
    q q
    \footnote #'(0.5 . -1) "Cambio di armatura" Staff.KeySignature
    \key c \minor
    q
  }
}
```




[1]Cambio di tempo
[2]Gambo dell'accordo
[3]Stanghetta
[4]Cambio di armatura

Si possono usare segni personalizzati in alternativa a quelli numerici, e si può sopprimere la linea che collega l'oggetto commentato al segno:

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "* La prima nota" } a'4
    b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
      \markup { \super "$" \italic " La seconda nota" } e
    c4
    \once \override Score.FootnoteItem.annotation-line = ##f
    b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
      \markup { \super "+" \italic " Editoriale" } \p
  }
}
```



* *La prima nota*
$ *La seconda nota*
+ *Editoriale*

Altri esempi di segni personalizzati si trovano in [Note a piè di pagina nel testo separato], pagina 505.

## Note a piè di pagina nel testo separato

Vengono usate all'interno di blocchi `\markup` che si trovano fuori dalle espressioni musicali. Non hanno una linea che le unisce al loro punto di riferimento: i loro segni seguono semplicemente il testo citato. I segni possono essere inseriti automaticamente, nel qual caso sono numerici; altrimenti è possibile inserire manualmente dei segni personalizzati.

Le note a piè di pagina su testo separato vengono create in modo diverso a seconda che si scelgano segni automatici oppure segni personalizzati.

### *Note a piè di pagina nel testo separato con segni automatici*

La sintassi di una nota a piè di pagina nel testo separato con segni automatici è

```
\markup { ... \footnote testo nota ... }
```

Gli elementi sono:
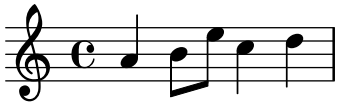
*testo* è il testo – racchiuso tra virgolette doppie o in un blocco markup – da contrassegnare.

*nota* è il testo della nota a piè di pagina.

Per esempio:

```
\book {
  \header { tagline = ##f }
  \markup {
    "Una semplice"
    \footnote "canzone" \italic " Scritta da me"
    "è mostrata sotto.  È una composizione"
    \footnote "recente" \italic " Agosto 2012"
    "."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

Una semplice canzone* è mostrata sotto. È una composizione recente†.



*Scritta da me*
*Agosto 2012*

### *Note a piè di pagina nel testo separato con segni personalizzati*

La sintassi di una nota a piè di pagina nel testo separato con segni personalizzati è

```
\markup { ... \footnote segno nota ... }
```

Gli elementi sono:
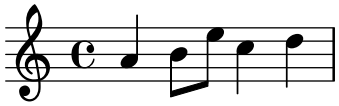
*segno* è una stringa di testo o un markup che indicano il segno da usare per contrassegnare il punto di riferimento. Tale segno *non* viene inserito automaticamente prima della nota stessa.

*nota* è una stringa di testo o un markup che indicano il testo della nota a piè di pagina, preceduto dal *segno*.

Qualsiasi carattere facile da scrivere, come * o +, può essere usato come segno, come è spiegato in [Note a piè di pagina nelle espressioni musicali], pagina 499. Altrimenti, si possono usare gli alias ASCII (vedi [Alias ASCII], pagina 520):

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "Una semplice canzone"
    \footnote "*" \italic "* Scritta da me"
    "è mostrata sotto.  È una composizione recente"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Agosto 2012"
    }
    "."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

Una semplice canzone * è mostrata sotto. È una composizione recente † .
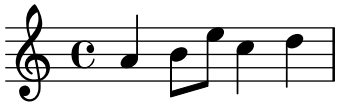


* *Scritta da me*
† *Agosto 2012*

È possibile usare anche i caratteri Unicode per indicare i segni (vedi [Unicode], pagina 519):

```
\book {
  \header { tagline = ##f }
  \markup {
    "Una semplice canzone"
    \footnote \super \char##x00a7 \concat {
      \super \char##x00a7 \italic " Scritta da me"
    }
    "è mostrata sotto.  È una composizione recente"
    \footnote \super \char##x00b6 \concat {
      \super \char##x00b6 \italic " Agosto 2012"
    }
    "."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

Una semplice canzone § è mostrata sotto. È una composizione recente ¶ .



§ *Scritta da me*
¶ *Agosto 2012*

## Vedi anche

Manuale di apprendimento: Sezione “Oggetti e interfacce” in *Manuale di Apprendimento*.

Guida alla notazione: [Alias ASCII], pagina 520, [Nuvoletta di aiuto], pagina 233, Sezione A.13 [Elenco dei caratteri speciali], pagina 763, [Indicazioni testuali], pagina 242, [Scritte], pagina 239, [Unicode], pagina 519.

Guida al funzionamento interno: Sezione “FootnoteEvent” in *Guida al Funzionamento Interno*, Sezione “FootnoteItem” in *Guida al Funzionamento Interno*, Sezione “FootnoteSpanner” in *Guida al Funzionamento Interno*, Sezione “Footnote_engraver” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Varie note a piè di pagina sulla stessa pagina possono essere messe soltanto una dopo l’altra; non è possibile disporle sulla stessa riga.

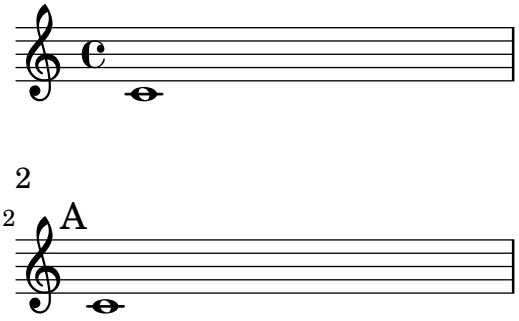
Le note a piè di pagina non possono essere collegate a `MultiMeasureRests`, a travature automatiche o a testo vocale.

I segni delle note a piè di pagina potrebbero entrare in collisione con righi, oggetti `\markup`, altri segni e linee delle note a piè di pagina.

## 3.2.5 Riferimento ai numeri di pagina

È possibile contrassegnare un punto specifico di una partitura usando il comando `\label` (etichetta) nel livello superiore o all’interno della musica. Questa etichetta può quindi essere citata all’interno di un blocco markup, per ottenere il numero di pagina in cui è stato inserito il punto contrassegnato, tramite il comando markup `\page-ref`.

```
\header { tagline = ##f }
\book {
  \label #'primoBrano
  \score {
    {
      c'1
      \pageBreak \mark A \label #'segnoA
      c'1
    }
  }
  \markup { Il primo brano inizia a pagina \page-ref #'primoBrano "0" "?" }
  \markup { Il segno A è a pagina \page-ref #'segnoA "0" "?" }
}
```



Il primo brano inizia a pagina 1
Il segno A è a pagina 2

Il comando markup `\page-ref` prende tre argomenti:

1. l'etichetta – un simbolo Scheme – per esempio `#'primoBrano`;
2. un testo markup da usare come misura di riferimento per stimare le dimensioni del testo;
3. un testo markup che verrà usato al posto del numero di pagina se l'etichetta non viene trovata.

Il motivo per cui è necessario una misura di riferimento è che, nel momento in cui vengono interpretati i testi (markup), le interruzioni di pagina non sono state decise, quindi i numeri di pagina non sono ancora noti. Per aggirare il problema, la vera interpretazione del testo viene rimandata a un momento successivo; tuttavia le dimensioni del testo devono essere conosciute prima, ecco perché serve una misura di riferimento per decidere tali dimensioni. Se il libro ha un numero di pagine compreso tra 10 e 99, tale misura sarà `"00"`, ovvero un numero di due cifre.

### Comandi predefiniti

`\label`, `\page-ref`.

## 3.2.6 Indice

L'indice si include col comando `\markuplist \table-of-contents`. Gli elementi che devono apparire nell'indice si inseriscono col comando `\tocItem`, che può essere usato nel livello superiore o all'interno di un'espressione musicale.

```
\markuplist \table-of-contents
\pageBreak

\tocItem \markup "Primo brano"
\score {
  {
    c'4  % ...
    \tocItem \markup "Un punto preciso nel primo brano"
    d'4  % ...
  }
}

\tocItem \markup "Secondo brano"
\score {
  {
    e'4 % ...
  }
}
```

I testi markup usati per formattare l'indice sono definiti nel blocco `\paper`. Ce ne sono due predefiniti:

- `tocTitleMarkup`

  Usato per formattare il titolo dell'indice.

  ```
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Indice" \null }
    \null
  }
  ```

- `tocItemMarkup`

  Usato per formattare gli elementi dell'indice.

  ```
  tocItemMarkup = \markup \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  ```

```
    }
```

Queste variabili possono essere entrambe modificate.

Ecco un esempio che mostra come cambiare il titolo dell'indice in francese:

```
\paper {
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
}
```

Ecco un esempio che mostra come cambiare il corpo dei caratteri nell'indice:

```
\paper {
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Nota il modo in cui sono citati il testo e il numero di pagina dell'elemento dell'indice nella definizione di `tocItemMarkup`.

Includendo il comando `\tocItemWithDotsMarkup` dentro `tocItemMarkup` lo spazio tra un elemento dell'indice e la sua pagina corrispondente sarà riempito con dei punti:

```
\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}
```

Table of Contents

Allegro . . . . . 1
Largo . . . . . 1

Si possono anche definire comandi personalizzati con markup specifici per creare un indice più complesso. Nell'esempio seguente, viene definito un nuovo stile per inserire i nomi degli atti nell'indice di un'opera.

Una nuova variabile di markup (chiamata `tocActMarkup`) viene definita nel blocco `\paper`:

```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}
```

Viene quindi aggiunta una funzione musicale personalizzata (`tocAct`), che usa la nuova definizione di markup `tocActMarkup`:

```
tocAct =
```

```
#(define-music-function (text) (markup?)
   (add-toc-item! 'tocActMarkup text))
```

Un file di input di LilyPond, che usi queste definizioni personalizzate, potrebbe avere il seguente output:

Table of Contents

*Atto Primo*

**Coro. Viva il nostro Alcide** **1**
**Cesare. Presti omai l'Egizia terra** **1**

*Atto Secondo*

**Sinfonia** **1**
**Cleopatra. V'adoro, pupille, saette d'Amore** **1**

Ecco un esempio del comando `\fill-with-pattern` usato nel contesto di un indice:

```
\paper {
  tocItemMarkup = \markup { \fill-line {
    \override #'(line-width . 70)
    \fill-with-pattern #1.5 #CENTER . \fromproperty #'toc:text \fromproperty #'toc:page
    }
  }
}
```

### Vedi anche

File installati: `ly/toc-init.ly`.

### Comandi predefiniti

`\table-of-contents`, `\tocItem`.

## 3.3 Lavorare coi file di input

### 3.3.1 Inclusione di file LilyPond

Se un progetto è grande e complesso, conviene suddividerlo in file separati. Per citare un altro file si usa

```
\include "altrofile.ly"
```

La riga `\include "altrofile.ly"` equivale a incollare i contenuti di `altrofile.ly` nel file corrente nel punto in cui appare il comando `\include`. Per esempio, in un progetto complesso si possono scrivere file separati per ogni parte strumentale e creare un file per l'"intera partitura" che raccoglie i file dei singoli strumenti. Di norma il file incluso definisce una serie di variabili che poi diventano disponibili nel file della partitura completa. Le sezioni contrassegnate con delle etichette nei file inclusi possono essere usate in varie parti di una partitura, vedi Sezione 3.3.2 [Edizioni diverse da un unico sorgente], pagina 513.

I file nella directory di lavoro corrente possono essere citati indicando semplicemente il nome del file dopo il comando `\include`. I file in altre posizioni possono essere inclusi sia con un percorso assoluto che con un percorso relativo (ma come separatore delle directory occorre

usare la barra, o slash, come in UNIX, piuttosto che la barra inversa, o backslash, come in DOS/Windows). Per esempio, se `cose.ly` si trova una directory prima della directory di lavoro corrente, usare

```
\include "../cose.ly"
```

oppure se le parti orchestrali incluse si trovano tutte in una sottodirectory chiamata `parti` all'interno della directory corrente, usare

```
\include "parti/VI.ly"
\include "parti/VII.ly"
... etc
```

I file che devono essere inclusi possono contenere essi stessi delle dichiarazioni `\include`. Per impostazione predefinita, queste dichiarazioni `\include` di secondo livello non sono interpretate finché non vengono portate nel file principale, dunque i nomi dei file che specificano devono essere tutti relativi alla directory del file principale, non alla directory del file incluso. Tuttavia tale comportamento può essere cambiato globalmente tramite l'opzione `-drelative-includes` da linea di comando (oppure agggiungendo `#(ly:set-option 'relative-includes #t)` in cima al file di input principale).

Quando `relative-includes` viene impostato su `#t`, il percorso di ogni comando `\include` sarà considerato relativo al file che contiene quel comando. Questo comportamento è raccomandato e diventerà il comportamento predefinito in una versione futura di lilypond.

È possibile includere sia file relativi alla directory principale sia file relativi a qualche altra directory impostando `relative-includes` su `#t` o `#f` nei punti giusti dei file. Per esempio, se è stata creata una libreria generale, libA, che usa altri file inclusi dal file di base di quella libreria, tali dichiarazioni `\include` dovranno essere precedute da `#(ly:set-option 'relative-includes #t)` per poter essere interpretate correttamente quando riportate nel file `.ly` principale:

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

quindi il file di base, `libA.ly`, conterrà

```
#(ly:set-option 'relative-includes #t)
\include "A1.ly"
\include "A2.ly"
...
% ritorna alle impostazioni predefinite
#(ly:set-option 'relative-includes #f)
```

Qualsiasi file `.ly` può quindi includere l'intera libreria semplicemente con

```
\include "~/libA/libA.ly"
```

Si possono ideare strutture di file più complesse facendo dei cambi nei punti giusti.

È possibile includere dei file anche da una directory che si trova in un percorso di ricerca specificato come opzione quando si lancia LilyPond da linea di comando. I file inclusi possono allora essere specificati usando soltanto il loro nome. Per esempio, per compilare con questo metodo il file `principale.ly` che include i file di una sottodirectory chiamata `parti`, entrare nella directory di `principale.ly` e eseguire questo comando

```
lilypond --include=parti principale.ly
```

e in principale.ly scrivere

```
\include "VI.ly"
\include "VII.ly"
```

```
... etc
```

I file che devono essere inclusi in molte partiture possono essere salvati nella directory di installazione di LilyPond `../ly`. La posizione di questa directory dipende dal tipo di installazione (vedi Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*). Questi file possono quindi essere inclusi semplicemente nominandoli in una dichiarazione `\include`. Questo è il modo in cui sono inclusi i file che dipendono dalla lingua, come `english.ly`.

LilyPond include un certo numero di file quando si lancia il programma. Queste inclusioni non sono evidenti all'utente, ma i file possono essere identificati eseguendo `lilypond --verbose` dalla linea di comando. Così si vedrà un elenco di percorsi e file che LilyPond usa, insieme a tante altre informazioni. I più importanti di questi file sono trattati in Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*. Tali file possono essere modificati, ma le modifiche saranno perse quando si installa una nuova versione di LilyPond.

Alcuni semplici esempi d'uso di `\include` si trovano in Sezione "Partiture e parti" in *Manuale di Apprendimento*.

### Vedi anche

Manuale di apprendimento: Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*, Sezione "Partiture e parti" in *Manuale di Apprendimento*.

### Problemi noti e avvertimenti

Se a un file incluso viene assegnato un nome identico a uno dei file di installazione di LilyPond, quest'ultimo ha la precedenza.

## 3.3.2 Edizioni diverse da un unico sorgente

Esistono vari metodi per generare versioni diverse di una partitura dalla stessa sorgente di musica. Le variabili sono forse le più utili per combinare lunghe sezioni musicali e/o note. Le etichette (*tag*) sono più utili per selezionare una sezione da varie sezioni brevi alternative e possono essere usate anche per unire insieme dei brani in punti diversi.
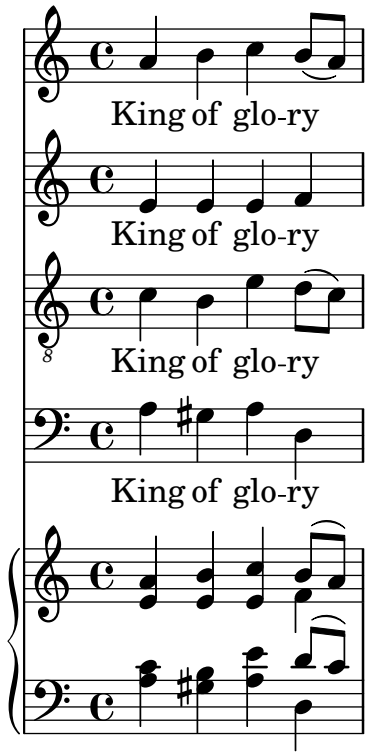
Qualsiasi metodo venga usato, la separazione delle note dalla struttura della partitura permetterà di cambiare la struttura lasciando le note intatte.

### Uso delle variabili

Se le sezioni musicali sono definite in variabili, possono essere riutilizzate in varie parti della partitura, come è stato spiegato in Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*. Per esempio, una partitura vocale *a cappella* spesso comprende, ai fini delle prove, una riduzione per pianoforte delle parti, identiche alla musica vocale, dunque la musica deve essere inserita una volta sola. La musica definita in due variabili può essere combinata in un rigo, come è spiegato in [Combinazione automatica delle parti], pagina 185. Ecco un esempio:

```
sopranoMusic = \relative { a'4 b c b8( a) }
altoMusic = \relative { e'4 e e f }
tenorMusic = \relative { c'4 b e d8( c) }
bassMusic = \relative { a4 gis a d, }
allLyrics = \lyricmode { King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenore" {
    \clef "treble_8"
```

```
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Basso" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "Destra" {
      \partCombine
      \sopranoMusic
      \altoMusic
    }
    \new Staff = "Sinistra" {
      \clef "bass"
      \partCombine
      \tenorMusic
      \bassMusic
    }
  >>
>>
```




Partiture separate che mostrino soltanto le parti vocali o soltanto quelle per pianoforte possono essere prodotte semplicemente cambiando le dichiarazioni della struttura, lasciando la notazione musicale intatta.

Nel caso di partiture lunghe, conviene mettere le definizioni delle variabili in file separati da includere, vedi Sezione 3.3.1 [Inclusione di file LilyPond], pagina 511.

## Uso delle etichette

Il comando `\tag #'parteA` contrassegna un'espressione musicale col nome *parteA*. Le espressioni contrassegnate in questo modo possono essere incluse o rimosse in base al loro nome successivamente, usando `\keepWithTag #'nome` oppure `\removeWithTag #'nome`. Il risultato dell'applicazione di questi filtri alla musica etichettata è il seguente:

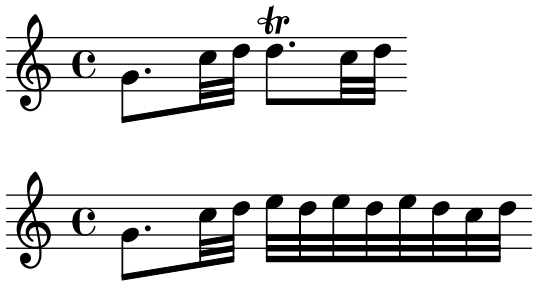
| **Filtro** | **Risultato** |
|---|---|
| Musica etichettata preceduta da `\keepWithTag #'nome` o `\keepWithTag #'(nome1 nome2...)` | Viene inclusa la musica non etichettata e quella etichettata con uno dei nomi specificati; la musica etichettata con un nome etichetta diverso viene esclusa. |
| Musica etichettata preceduta da `\removeWithTag #'nome` o `\removeWithTag #'(nome1 nome2...)` | Viene inclusa la musica non etichettata e quella non etichettata con uno dei nomi specificati; la musica etichettata con uno dei nomi specificati viene esclusa. |
| Musica etichettata non preceduta da `\keepWithTag` o `\removeWithTag` | Viene inclusa tutta la musica etichettata e non etichettata. |

Gli argomenti dei comandi `\tag`, `\keepWithTag` e `\removeWithTag` devono essere un simbolo o una lista di simboli (come `#'score` o `#'(violinoI violinoII`), seguiti da un'espressione musicale. Se *e solo se* i simboli sono identificatori LilyPond validi (solo caratteri alfabetici, nessun numero, trattino o trattino basso) che non possono essere confusi con le note, si può omettere il `#'` e, come scorciatoia, una lista di simboli può usare il punto come separatore: quindi `\tag #'(violinoI violinoII)` può essere riscritto come `\tag violinoI.violinoII`. Lo stesso vale per `\keepWithTag` e `\removeWithTag`.

Nell'esempio seguente, vediamo due versioni di un brano musicale, una che mostra i trilli con la notazione abituale e l'altra con i i trilli espansi esplicitamente:

```
musica = \relative {
  g'8. c32 d
  \tag #'trilli { d8.\trill }
  \tag #'espandi { \repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \keepWithTag #'trilli \musica
}
\score {
  \keepWithTag #'espandi \musica
}
```

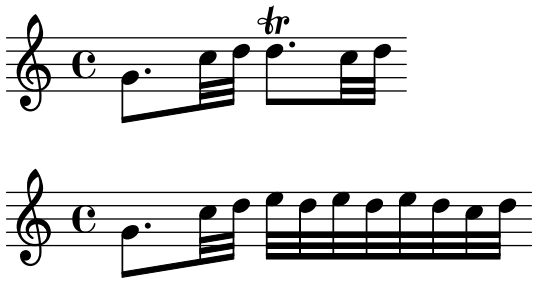


Altrimenti, talvolta è più facile escludere sezioni musicali:

```
musica = \relative {
```

```
  g'8. c32 d
  \tag #'trilli { d8.\trill }
  \tag #'espandi {\repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \removeWithTag #'espandi
  \musica
}
\score {
  \removeWithTag #'trilli
  \musica
}
```



Il filtro delle etichette può essere applicato a articolazioni, testo, etc. scrivendo

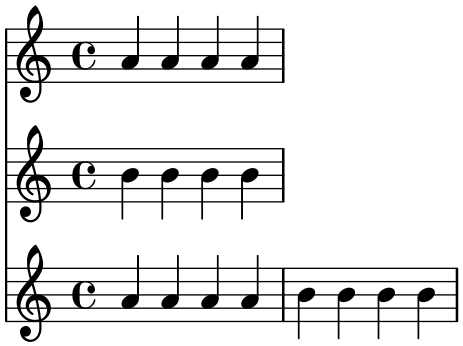
```
-\tag #'tua-etichetta
```

prima di un'articolazione. L'esempio seguente definisce una nota con un'indicazione di diteggiatura condizionale e una nota con un commento condizionale:

```
c1-\tag #'dito ^4
c1-\tag #'attenzione ^"Fai attenzione!"
```
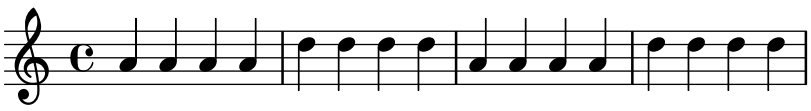
Varie etichette possono essere associate a delle espressioni tramite molteplici `\tag`, o unendo molteplici etichette in una lista di simboli:

```
musica = \relative c'' {
  \tag #'a \tag #'entrambi { a4 a a a }
  \tag #'(b entrambi) { b4 b b b }
}
<<
\keepWithTag #'a \musica
\keepWithTag #'b \musica
\keepWithTag #'entrambi \musica
>>
```



Si possono applicare molteplici filtri `\removeWithTag` a una singola espressione musicale per togliere varie sezioni etichettate con nomi diversi. Altrimenti si può usare un solo `\removeWithTag` seguito da una lista di etichette.

```
musica = \relative c'' {
  \tag #'A { a4 a a a }
  \tag #'B { b4 b b b }
  \tag #'C { c4 c c c }
  \tag #'D { d4 d d d }
}
\new Voice {
  \removeWithTag #'B
  \removeWithTag #'C
  \musica
  \removeWithTag #'(B C)
  \musica
}
```



Applicando due o più filtri `\keepWithTag` su una singola espressione musicale toglieranno *tutte* le sezioni etichettate. Il primo filtro toglie tutto tranne la sezione menzionata e qualsiasi filtro successivo toglie il resto. Usando un singolo comando `\keepWithTag` con una lista di varie etichette vengono rimosse soltanto le sezioni etichettate non specificate in quella lista.

```
music = \relative c'' {
  \tag #'violinoI { a4 a a a }
  \tag #'violinoII { b4 b b b }
  \tag #'viola { c4 c c c }
  \tag #'cello { d4 d d d }
}

\new Staff {
  \keepWithTag #'(violinI violinII)
  \music
}
```

farà apparire le etichette *violinoI* e *violinoII* ma non *viola* o *cello*.

Mentre `\keepWithTag` è comodo con *un* gruppo di alternative, la rimozione di musica contrassegnata con etichette *indipendenti* è problematico se si usano le etichette con diverse finalità. In questo caso, è possibile dichiarare 'gruppi' di etichette:

```
\tagGroup #'(violinoI violinoII viola cello)
```

dichiara che queste etichette appartengono a un gruppo.
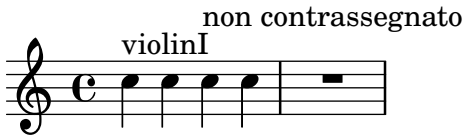
```
\keepWithTag #'violinoI ...
```

ora mostrerà soltanto la musica contrassegnata con le etichette presenti nel gruppo cui appartiene `violinoI` e verrà tolta la musica contrassegnata con una delle *altre* etichette.

```
music = \relative {
  \tagGroup #'(violinoI violinoII viola cello)
  \tag #'violinoI { c''4^"violinI" c c c }
  \tag #'violinoII { a2 a }
  \tag #'viola { e8 e e2. }
```

```
  \tag #'cello { d'2 d4 d }
  R1^"non contrassegnato"
}

\new Voice {
  \keepWithTag #'violinoI
  \music
}
```
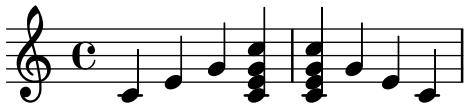



Quando si usa il comando `\keepWithTag`, sono visibili solo le etichette provenienti dai gruppi cui appartengono le etichette specificate dopo il comando.

Talvolta si ha necessità di combinare insieme della musica in un punto preciso di un'espressione musicale esistente. `\pushToTag` e `\appendToTag` permettono di aggiungere materiale prima o dopo gli elementi di un costrutto musicale esistente. Non tutti i costrutti musicali hanno elementi, ma nel caso di musica sequenziale e simultanea si può esserne sicuri:

```
music = { \tag #'qui { \tag #'qui <<c''>> } }

{
  \pushToTag #'qui c'
  \pushToTag #'qui e'
  \pushToTag #'qui g' \music
  \appendToTag #'qui c'
  \appendToTag #'qui e'
  \appendToTag #'qui g' \music
}
```



Entrambi i comandi prendono tre argomenti: un'etichetta, il materiale da combinare ad ogni occorenza dell'etichetta e l'espressione contrassegnata.

## Vedi anche

Manuale di apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

Guida alla notazione: [Combinazione automatica delle parti], pagina 185, Sezione 3.3.1 [Inclusione di file LilyPond], pagina 511.

## Problemi noti e avvertimenti

Se si usa `\relative` prima di un'espressione musicale ottenuta filtrando la musica con `\keepWithTag` o `\removeWithTag`, i rapporti di ottava potrebbero cambiare, perché verranno considerate solo le altezze rimaste nell'espressione filtrata. Per evitare questo rischio, usare `\relative` prima di `\keepWithTag` o `\removeWithTag`, in modo che `\relative` agisca su tutte le altezze prima del filtro.

### Impostazioni globali

È possibile includere impostazioni globali da un altro file:

```
lilypond -dinclude-settings=MIE_IMPOSTAZIONI.ly MIA_PARTITURA.ly
```

Gruppi di impostazioni come dimensioni del foglio e tipo di carattere possono essere salvati in file separati. Ciò permette di ottenere diverse edizioni dalla stessa partitura o di applicare delle impostazioni tipiche a molte partiture, semplicemente indicando il relativo file delle impostazioni.

Questa tecnica è la stessa usata per i fogli di stile, trattati in Sezione "Fogli di stile" in *Manuale di Apprendimento*.

### Vedi anche

Manuale di apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*, Sezione "Fogli di stile" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 3.3.1 [Inclusione di file LilyPond], pagina 511.

## 3.3.3 Caratteri speciali

### Codifica del testo

LilyPond usa il repertorio di caratteri definito dall'Unicode Consortium e dalla ISO/IEC 10646. Questo sistema di codifica definisce un nome e un numero univoci per gli insiemi di caratteri utilizzati in tutte le lingue moderne e anche in molte altre. Unicode può essere implementato in varie codifiche diverse. LilyPond usa la codifica UTF-8 (UTF sta per Unicode Transformation Format) che rappresenta tutti i comuni caratteri latini con un byte, e gli altri caratteri con un formato di lunghezza variabile fino a quattro byte.
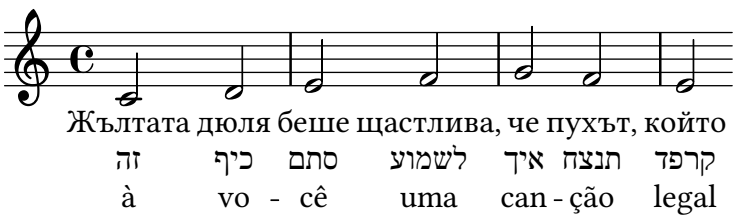
L'aspetto grafico dei caratteri è determinato dai glifi definiti nei tipi di carattere disponibili - un tipo di carattere definisce la mappatura tra un sottoinsieme dei numeri Unicode e i glifi. LilyPond usa la libreria Pango per rappresentare e formattare i testi multilingua.

LilyPond non esegue alcuna conversione della codifica dell'input. Ciò significa che qualsiasi testo, sia esso un titolo, un testo vocale o un'istruzione musicale contenente caratteri non-ASCII, deve essere codificato in UTF-8. Il modo più semplice per inserire tale testo è usare un editor che sappia riconoscere la codifica Unicode e salvare il file con la codifica UTF-8. La maggior parte dei moderni editor supporta la codifica UTF-8, per esempio vim, Emacs, jEdit e Gedit. Tutti i sistemi MS Windows successivi a NT usano Unicode come codifica dei caratteri nativa, quindi perfino Notepad può modificare e salvare un file in formato UTF-8. Un'alternativa più efficiente per Windows è BabelPad.

Se un file di input LilyPond contenente un caratter non-ASCII non viene salvato in formato UTF-8, apparirà il seguente messaggio di errore:

```
FT_Get_Glyph_Name () error: invalid argument
```

Ecco un esempio che mostra del testo cirillico, ebraico e portoghese:




### Unicode

Per inserire un singolo carattere per il quale è noto il codice Unicode ma che non è disponibile nell'editor in uso, usare `\char ##xhhhh` o `\char #dddd` dentro un blocco `\markup`, dove `hhhh` è

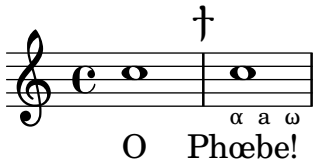
il codice esadecimale del carattere richiesto e `dddd` è il valore decimale corrispondente. Gli zero iniziali possono essere omessi, ma di norma nella rappresentazione esadecimale si specificano tutti e quattro i caratteri. (Fare attenzione al fatto che la codifica UTF-8 del codice *non* deve essere usata dopo `\char`, perché le codifiche UTF-8 contengono bit extra che indicano il numero di ottetti.) Le tabelle dei codici Unicode e un indice dei nomi dei caratteri col proprio codice esadecimale sono disponibili sul sito dell'Unicode Consortium, `http://www.unicode.org/`.

Per esempio, `\char ##x03BE` e `\char #958` corrispondono entambi al carattere Unicode U+03BE, che ha il nome Unicode "Greek Small Letter Xi".

Qualsiasi codice Unicode può essere inserito in questo modo e se tutti i caratteri speciali sono inseriti in questo formato non è necessario salvare il file di input in formato UTF-8. Ovviamente, un tipo di carattere contenente tutti questi caratteri codificati deve essere installato e disponibile per LilyPond.

L'esempio seguente mostra valori esadecimali Unicode inseriti in quattro posti diversi: come numero di chiamata, come articolazione, nel testo vocale e come testo separato sotto il brano:

```
\score {
  \relative {
    c''1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " a " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2020" \char ##x00A9 }
```




Copyright 2008--2020 ©

Per inserire il segno del copyright nell'apposito campo usare:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

## Alias ASCII

È possibile includere una lista di alias ASCII per i caratteri speciali:
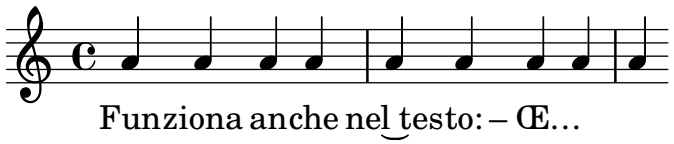
```
\paper {
  #(include-special-characters)
}

\markup "&flqq; &ndash; &OE;uvre incomplète&hellip; &frqq;"

\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    Fun -- ziona an -- che nel~tes -- to: &ndash;_&OE;&hellip;
  }
}
```

```
\markup \column {
  "La sostituzione può essere disabilitata:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète… »



Funziona anche nel testo: – Œ…

La sostituzione può essere disabilitata:
– Œ …
– &OE; …

Si possono creare i propri alias, sia globalmente:

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

che localmente:

```
\markup \replace #'(("100" . "hundred")
                    ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

### Vedi anche

Guida alla notazione: Sezione A.13 [Elenco dei caratteri speciali], pagina 763.

File installati: `ly/text-replacements.ly`.

## 3.4 Controllo dell'output

### 3.4.1 Estrarre frammenti musicali

È possibile creare output separati di uno o più frammenti di una partitura definendo i punti della musica da estrarre nel blocco `\layout` del file di input tramite la funzione `clip-regions`, e poi eseguendo LilyPond con l'opzione `-dclip-systems`;

```
\layout {
  clip-regions
  = #(list
      (cons
       (make-rhythmic-location 5 1 2)
       (make-rhythmic-location 7 3 4)))
```

```
}
```

Questo esempio estrarrà dal file di input un unico frammento che *inizia* dopo una minima nella quinta misura (`5 1 2`) e *termina* dopo la terza semiminima nella settima misura (`7 3 4`).

Si possono estrarre ulteriori frammenti aggiungendo altre coppie di `make-rhythmic-location` alla lista `clip-regions` del blocco `\layout`.

Ciascun frammento musicale verrà salvato in formato `EPS`, ma se necessario si possono creare anche altri formati come `PDF` o `PNG`. La musica estratta viene generata come se fosse letteralmente 'tagliata' dalla partitura a stampa originale; ciò significa che se un frammento supera una o più linee, verrà creato un file di output separato per ciascuna linea.

### Vedi anche

Guida alla notazione: Sezione 4.2.1 [Il blocco layout], pagina 550.

Utilizzo: Sezione "Uso da linea di comando" in *Uso del Programma*.

## 3.4.2 Saltare la musica già corretta

Quando si inserisce o si copia della musica, di solito è utile visualizzare e correggere soltanto la musica vicina alla fine (ovvero dove si stanno inserendo le note). Per velocizzare il processo di correzione, è possibile far sì che il compositore tipografico salti tutte le misure eccetto le ultime. Per farlo basta inserire per esempio
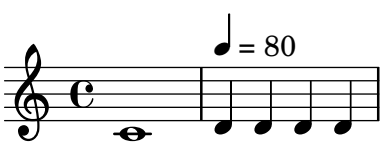
```
showLastLength = R1*5
\score { ... }
```

nel file sorgente. In questo modo verranno elaborate soltanto le ultime 5 misure (assumendo che il tempo sia 4/4) di ogni blocco `\score` nel file di input. Per i brani più lunghi, elaborare solo una piccola parte è spesso molto più veloce di elaborarli completamente. Se si lavora sull'inizio di un brano già scritto (per esempio per aggiungere una nuova parte), si userà invece la proprietà `showFirstLength`.

È possibile saltare parti di una partitura in un modo più preciso tramite la proprietà `Score.skipTypesetting`. Quando è impostata su vero, la composizione tipografica è disattivata.

Questa proprietà viene usata anche per controllare l'output da inviare al file MIDI. Attenzione: salta tutti gli eventi, inclusi i cambi di tempo e di strumento.

```
\relative c' {
  c1
  \set Score.skipTypesetting = ##t
  \tempo 4 = 80
  c4 c c c
  \set Score.skipTypesetting = ##f
  d4 d d d
}
```




Nella musica polifonica, `Score.skipTypesetting` agisce su tutte le voci e su tutti i righi, facendo risparmiare tempo ulteriormente.

### 3.4.3 Formati di output alternativi

I formati di output predefiniti per la partitura stampata sono Portable Document Format (PDF) e PostScript (PS). Sono disponibili anche i formati di output Portable Network Graphics (PNG), Scalable Vector Graphics (SVG) e Encapsulated PostScript (EPS) attraverso le opzioni da linea di comando, vedi Sezione "Opzioni di base della linea di comando per LilyPond" in *Uso del Programma*.

#### Output SVG

L'output SVG può contenere opzionalmente dei metadati per gli oggetti grafici (grob) come teste di nota, pause, etc. Questi metadati possono essere attributi SVG standard come `id` e `class` oppure attributi personalizzati e non standard. Gli attributi e i loro valori si specificano sovrascrivendo la proprietà `output-attributes` di un grob con una lista associativa Scheme (alist). I valori possono essere numeri, stringhe o simboli. Per esempio:

```
{
  \once \override NoteHead.output-attributes =
  #'((id . 123)
     (class . "questo quello")
     (data-testo . qualcosa))
  c
}
```

L'input precedente produrrà il seguente elemento `<g>` (gruppo) nel file SVG:

```
<g id="123" class="questo quello" data-testo="qualcosa">
  ...Elementi SVG del grob NoteHead...
</g>
```

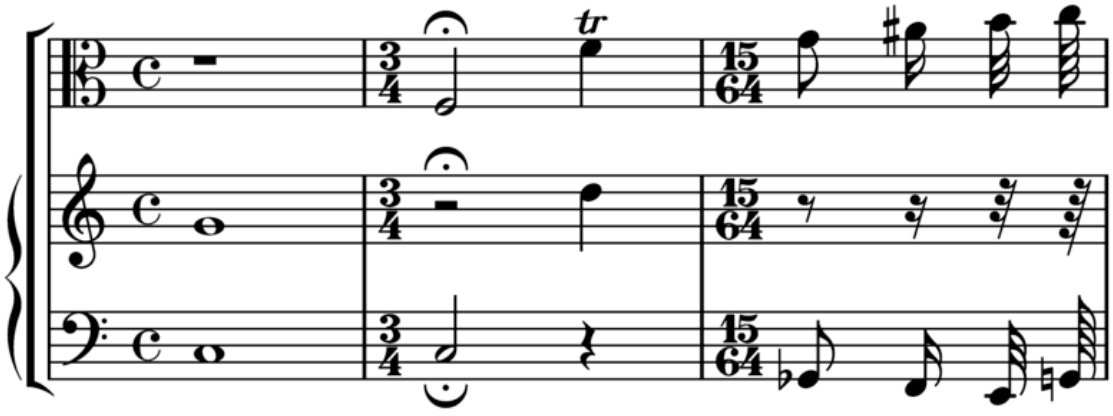
L'elemento `<g>` contiene tutti gli elementi SVG di un certo grob (alcuni grob generano molteplici elementi SVG). Nella sintassi SVG il prefisso `data-` è utilizzato per gli attributi di metadati personalizzati non standard.

### 3.4.4 Cambiare il tipo di carattere della notazione
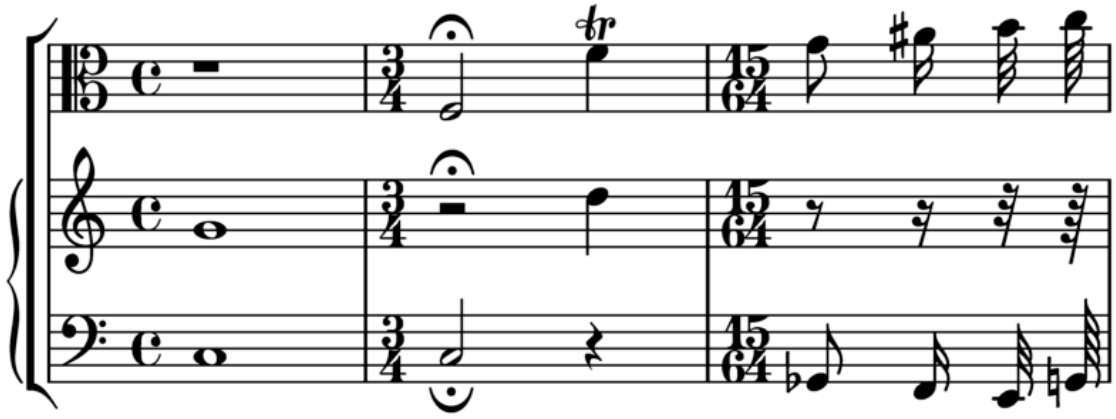
Gonville è un insieme di glifi alternativo a *Feta*, parte del font Emmentaler, e può essere usato in LilyPond. Può essere scaricato da:

`http://www.chiark.greenend.org.uk/~sgtatham/gonville/ (http://www.chiark.greenend.org.uk/~sgtatham/gonville/ )`

Ecco alcune battute di musica che usa Gonville:



E alcune battute di musica che usa i glifi Feta di LilyPond:



## Instruzioni di installazione

Scaricare e estrarre i file dei font. Copiare i file `gonville-11.otf`, `gonville-13.otf`, `gonville-14.otf`, `gonville-16.otf`, `gonville-18.otf`, `gonville-20.otf`, `gonville-23.otf`, `gonville-26.otf` e `gonville-brace.otf` in `.../share/lilypond/current/fonts/otf` o `.../share/lilypond/X.Y.Z/fonts/otf`. Se sono presenti file `gonville-*.svg` e `gonville-*.woff`, copiarli in `.../share/lilypond/current/fonts/svg` o `.../share/lilypond/X.Y.Z/fonts/svg`. Maggiori informazioni in Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Nota: i file `gonville-*.otf` sono utilizzati dal backend `ps` e `eps` (per la creazione di file PDF e PostScript). I file `gonville-*.svg` sono utilizzati dal backend `svg` quando non è attiva l'opzione `svg-woff`. I file `gonville-*.woff` sono utilizzati dal backend `svg` quando è attiva l'opzione `svg-woff`. Maggiori informazioni in Sezione "Opzioni avanzate della linea di comando per LilyPond" in *Uso del Programma*.

La sintassi seguente imposta il font Gonville come font della notazione (generale e delle parentesi).

```
\paper {
  #(define fonts
     (set-global-fonts
      #:music "gonville"
      #:brace "gonville"
    ))
}
```

Nota: ogni chiamata della funzione `set-global-fonts` reimposta completamente i font sia della notazione principale che del testo. Se una categoria non è specificata, allora verrà usato il font predefinito per quella categoria. Ogni chiamata di `set-global-fonts` cambia i font di ciascun blocco `\book` che la segue, sia esso creato esplicitamente o implicitamente. Ciò significa che ogni blocco `\book` può avere il suo insieme di font principali usando `set-global-fonts` prima di esso. Maggiori informazioni in [Tipi di carattere per l'intero documento], pagina 264.

## Vedi anche

Manuale di apprendimento: Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione A.8 [Il font Emmentaler], pagina 679, [Tipi di carattere per l'intero documento], pagina 264.

### Problemi noti e avvertimenti

Gonville non può essere usato per scrivere notazione in 'musica antica' ed è probabile che i nuovi glifi introdotti in rilasci successivi di LilyPond non esistano in Gonville. Fare riferimento al sito web dell'autore per maggiori informazioni su queste e altre problematiche, inclusa la licenza di Gonville.

### Altri font della notazione

Se si hanno altri font della notazione come *nomefont*`-*.otf`, *nomefont*`-*.svg` e *nomefont*`-*.woff`, si possono usare nello stesso modo che abbiamo visto con Gonville.

Ovvero si copiano i file *nomefont*`-*.otf` in `.../share/lilypond/current/fonts/otf` o `.../share/lilypond/`*X.Y.Z*`/fonts/otf`. E i file *nomefont*`-*.svg` e *nomefont*`-*.woff`, se presenti, in `.../share/lilypond/current/fonts/svg` o `.../share/lilypond/`*X.Y.Z*`/fonts/svg`.

Nota: per il momento, LilyPond richiede che i nomi dei file dei font abbiano i seguenti suffissi, che devono essere tutti presenti nelle cartelle di installazione menzionate sopra perché possa funzionare correttamente: `-11`, `-13`, `-14`, `-16`, `-18`, `-20`, `-23`, `-26`, `-brace`. Per esempio, `emmentaler-11.otf`, `emmentaler-20.svg` e `emmentaler-brace.woff`, etc.

La sintassi seguente imposta il font *nomefont* come font della notazione (generale e delle parentesi).

```
\paper {
  #(define fonts
    (set-global-fonts
     #:music "fontname" ; nome del file del font senza suffisso né estensione
     #:brace "fontname" ; nome del file del font senza suffisso né estensione
   ))
}
```

Nota: per le categorie `music` e `brace`, specificare il nome del file del font senza suffisso né estensione.

## 3.5 Creazione dell'output MIDI

LilyPond è in grado di produrre dei file conformi allo standard MIDI (Musical Instrument Digital Interface) e dunque permette di fare un controllo uditivo dell'output musicale (con l'aiuto di un'applicazione o un dispositivo che comprendano il MIDI). L'ascolto dell'output MIDI può aiutare anche a individuare errori, come note inserite in modo erroneo o note senza alterazioni e così via.

I file MIDI non contengono audio (come i file AAC, MP3 o Vorbis), bensì richiedono un software esterno che produca il suono a partire da essi.

### 3.5.1 Notazione supportata nel MIDI

I seguenti elementi della notazione musicale saranno resi nell'output MIDI usando le capacità predefinite di LilyPond:

- Respiri
- Accordi inseriti come nomi degli accordi
- Crescendi e decrescendi su varie note. Il volume viene modificato in modo proporzionato fra i due estremi
- Segni di dinamica da `ppppp` a `fffff`, inclusi `mp`, `mf` e `sf`
- Microtoni ma *non* gli accordi microtonali. È necessario anche un lettore MIDI che supporti la "piegatura" (*bending*) delle altezze.
- Testo vocale

- Altezze
- Ritmi inseriti come durate, inclusi i gruppi irregolari
- Articolazioni 'semplici'; staccato, staccatissimo, accento, marcato e portato
- Cambi di tempo tramite la funzione `\tempo`
- Legature di valore
- Tremoli che *non* sono inseriti con un valore ':[*numero*]'

Alcuni effetti, come panning, bilanciamento, espressione, riverbero e chorus, si gestiscono impostando le proprietà di contesto, vedi Sezione 3.5.8 [Proprietà di contesto per gli effetti MIDI], pagina 535.

Se si usa lo script `articulate`, anche i seguenti elementi della notazione musicale saranno presenti nell'output MIDI:

- Appoggiature. Servono a rubare metà del valore della nota che le segue (senza considerare i punti). Per esempio:

  ```
  \appoggiatura c8 d2.
  ```

  Il c avrà il valore di una semiminima.
- Ornamenti (mordenti, trilli, gruppetti, etc.)
- Rallentando, accelerando, ritardando e a tempo
- Legature di portamento, incluse le legature di frase
- Tenuto

Vedi Sezione 3.5.9 [Miglioramento dell'output MIDI], pagina 536.

### 3.5.2 Notazione non supportata nel MIDI

I seguenti elementi della notazione musicale non possono essere trasferiti nel file MIDI:

- Articolazioni diverse da staccato, staccatissimo, accento, marcato e portato
- Crescendi e decrescendi su una *singola* nota
- Corona
- Basso numerato (continuo)
- Glissandi
- Portamenti indeterminati verso il basso e verso l'alto
- Accordi microtonali
- Ritmi inseriti come note, per esempio lo swing
- Cambi di tempo senza usare `\tempo` (per esempio, inseriti come note)
- Tremoli che *sono* inseriti con un valore ':[*numero*]'

### 3.5.3 Il blocco MIDI

Per creare un file MIDI da un file di input LilyPond, inserire un blocco `\midi`, che può essere vuoto, all'interno di un blocco `\score`;

```
\score {
  ... musica ...
  \layout { }
  \midi { }
}
```

> **Nota:** Un blocco `\score` che, oltre alla musica, contenga soltanto un blocco `\midi` (ovvero *sia privo* del blocco `\layout`), produrrà come output soltanto file MIDI e nessun file visuale.

L'estensione predefinita del file di output (`.midi`) può essere modificata tramite l'opzione `-dmidi-extension` del comando `lilypond`:

```
lilypond -dmidi-extension=mid MyFile.ly
```

Altrimenti, aggiungere la seguente espressione Scheme prima dell'inizio dei blocchi `\book`, `\bookpart` o `\score`. Vedi Sezione 3.1.5 [Struttura del file], pagina 484.

```
#(ly:set-option 'midi-extension "mid")
```

### Vedi anche

Guida alla notazione: Sezione 3.1.5 [Struttura del file], pagina 484, Sezione 3.2.3 [Creazione di metadati per i file di output], pagina 498.

File installati: `scm/midi.scm`.

### Problemi noti e avvertimenti

Sono disponibili quindici canali MIDI e un canale ulteriore (#10) per le percussioni. I righi sono assegnati ai canali in sequenza: in una partitura di più di quindici righi, i righi extra condivideranno (senza sovrascriverlo) lo stesso canale MIDI. Ciò potrebbe essere un problema se i righi in questione sono impostati con proprietà MIDI in conflitto e basate sul canale, come ad esempio diversi strumenti MIDI.

L'uso di un blocco `midi` con notazione polimetrica potrebbe produrre avvisi imprevisti di controllo della battuta. In questo caso occorre spostare `Timing_translator` dal contesto `Score` al contesto `Staff` all'interno del blocco `midi`.

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

## 3.5.4 Gestione delle dinamiche nel MIDI

È possile regolare il volume MIDI complessivo, il volume relativo dei segni di dinamica e il volume relativo dei diversi strumenti.

Le dinamiche vengono trasferite automaticamente sui livelli di volume nella gamma di volume MIDI disponibile, mentre crescendi e decrescendi variano il volume gradualmente tra i loro due estremi. È possibile regolare il volume relativo delle dinamiche e i livelli del volume complessivo dei diversi strumenti.

### Dinamiche nel MIDI

Solo i segni di dinamica compresi tra `ppppp` e `fffff`, inclusi `mp`, `mf` e `sf` hanno dei valori assegnati. Questo valore viene poi applicato al valore della gamma del volume MIDI complessivo per ottenere il volume finale incluso nell'output MIDI per quella particolare dinamica. Le frazioni predefinite vanno da 0.25 per *ppppp* a 0.95 per *fffff*. L'insieme completo di dinamiche e delle loro frazioni associate si trova in `scm/midi.scm`.

## Frammenti di codice selezionati

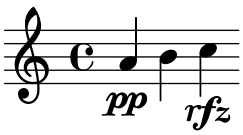
*Creare dinamiche personalizzate nell'output MIDI*

L'esempio seguente mostra come creare un segno di dinamica, non incluso nell'elenco predefinito, e assegnargli un valore specifico così che possa essere usato per cambiare l'output MIDI.

Al segno di dinamica `\rfz` viene assegnato il valore `0.9`.

```
#(define (myDynamics dynamic)
    (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```




File installati: `ly/script-init.ly` `scm/midi.scm`.

Frammenti: Sezione "MIDI" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Dynamic_performer" in *Guida al Funzionamento Interno*.

## Impostazione del volume MIDI

I valori minimo e massimo del volume complessivo delle dinamiche MIDI sono regolati dalle proprietà `midiMinimumVolume` e `midiMaximumVolume` nel livello `Score`. Tali proprietà hanno effetto soltanto all'inizio di una voce e sui segni di dinamica. La frazione corrispondente a ciascun segno di dinamica viene modificata con la seguente formula:

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * frazione
```

Nell'esempio seguente la gamma dinamica del volume MIDI complessivo è limitata alla gamma `0.2` - `0.5`.

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = "flauto"
      ... musica ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinetto"
```

```
      ... musica ...
    }
  >>
  \midi {
    \context {
      \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```

Una semplice equalizzazione degli strumenti MIDI si può ottenere impostando le proprietà `midiMinimumVolume` e `midiMaximumVolume` nel contesto `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "flauto"
    \set Staff.midiMinimumVolume = #0.7
    \set Staff.midiMaximumVolume = #0.9
    ... musica ...
  }
  \midi { }
}
```

In caso di partiture con molti righi e molti strumenti MIDI, i volumi relativi di ogni strumento possono essere impostati individualmente;

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = "flauto"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      ... musica ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinetto"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      ... musica ...
    }
  >>
  \midi { }
}
```

In questo esempio il volume del clarinetto è diminuito in modo proporzionale al volume del flauto.

Se queste proprietà del volume non sono impostate, LilyPond applica comunque “un po’” di equalizzazione a certi strumenti. Vedi `scm/midi.scm`.

File installati: `scm/midi.scm`.

## Vedi anche

Guida alla notazione: Sezione 4.2 [Formattazione della partitura], pagina 550.

Guida al funzionamento interno: Sezione “Dynamic_performer” in *Guida al Funzionamento Interno*.

## Frammenti di codice selezionati

*Modificare l’equalizzazione predefinita degli strumenti MIDI*

L’equalizzatore predefinito degli strumenti MIDI può essere modificato impostando la proprietà `instrumentEqualizer` nel contesto `Score` come una procedura Scheme definita dall’utente che usi il nome dello strumento MIDI come argomento insieme a una coppia di frazioni indicanti i volumi minimi e massimi da applicare a quello specifico strumento.
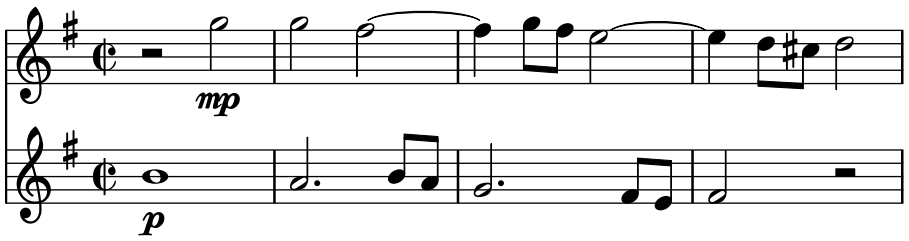
L’esempio seguente imposta i volumi massimo e minimo per il flauto e per il clarintetto.

```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = "flute"
      \new Voice \relative {
        r2 g''\mp g fis~
        4 g8 fis e2~
        4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = "clarinet"
      \new Voice \relative {
        b'1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {  }
}
```




## Problemi noti e avvertimenti

Le modifiche al volume MIDI si verificano soltanto all'inizio di una nota, quindi i crescendi e i decrescendi non possono cambiare il volume di una singola nota.

## Impostazione delle proprietà del blocco MIDI

Il blocco `\midi` può contenere modifiche del contesto, nuove definizioni del contesto o codice che imposti i valori di certe proprietà.

```
\score {
  ... musica ...
  \midi {
    \tempo 4 = 72
  }
}
```

In questo esempio il tempo è impostato su 72 battiti da un quarto per minuto. Il segno di tempo nel blocco `\midi` non appare nella partitura, mentre qualsiasi altra indicazione di `\tempo` specificata nel blocco `\score` sarà trasferita anche nell'output MIDI.

Se all'interno di un blocco `\midi`, il comando `\tempo` imposta le proprietà durante l'interpretazione della musica e nel contesto delle definizioni di output; dunque viene interpretato *come se* fosse una modifica di contesto.

Le definizioni di contesto seguono la stessa sintassi di quelle di un blocco `\layout`;

```
\score {
  ... musica ...
  \midi {
    \context {
      \Voice
      \remove "Dynamic_performer"
    }
  }
}
```

Questo esempio toglie l'effetto delle dinamiche dall'output MIDI. Nota bene: i moduli di traduzione di LilyPond usati per l'audio si chiamano 'performer'.

## Vedi anche

Manuale di apprendimento: Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.3 [Segni di espressione], pagina 123, Sezione 4.2 [Formattazione della partitura], pagina 550.

File installati: `ly/performer-init.ly`.

Frammenti: Sezione "MIDI" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Dynamic_performer" in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Alcuni lettori MIDI non sempre gestiscono correttamente i cambi di tempo.

I cambi di `midiInstrument`, e di altre opzioni MIDI, all'*inizio* di un rigo potrebbero apparire due volte nell'output MIDI.

## 3.5.5 Uso degli strumenti MIDI

Gli strumenti MIDI si impostano tramite la proprietà `midiInstrument` del contesto `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "glockenspiel"
    ... musica ...
  }
  \midi { }
}
```

o

```
\score {
  \new Staff \with {midiInstrument = "cello"} {
    ... musica ...
  }
  \midi { }
}
```

Se il nome dello strumento non corrisponde a nessuno degli strumenti elencati nella sezione 'Strumenti MIDI', verrà usato lo strumento `acoustic grand`. Vedi Sezione A.6 [Strumenti MIDI], pagina 677.

### Vedi anche

Manuale di apprendimento: Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione A.6 [Strumenti MIDI], pagina 677, Sezione 4.2 [Formattazione della partitura], pagina 550.

File installati: `scm/midi.scm`.

### Problemi noti e avvertimenti

Gli strumenti percussivi che sono scritti in un contesto `DrumStaff` verranno inviati, correttamente, al canale MIDI 10, ma alcuni strumenti percussivi con un tono musicale, come xilofono, marimba, vibrafono o timpano, sono trattati come strumenti "normali", quindi la musica di tali strumenti deve essere inserita in un contesto `Staff` (non `DrumStaff`) per ottenere l'output MIDI corretto. L'elenco completo delle voci del set di percussioni del canale 10 `channel 10 drum-kits` si trova in `scm/midi.scm`. Vedi Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

## 3.5.6 Uso delle ripetizioni nel MIDI

Le ripetizioni possono essere applicate all'output MIDI tramite il comando `\unfoldRepeats`.

```
\score {
  \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' }
      }
    }
    \midi { }
  }
```

Per limitare l'effetto di `\unfoldRepeats` all'output MIDI soltanto, senza modificare la partitura stampata, occorre creare *due* blocchi `\score`; uno per il MIDI (con ripetizioni ricopiate) e uno per la partitura scritta (con ripetizioni con volta, tremolo e percentuale):

```
\score {
  ... musica ...
  \layout { }
}
\score {
  \unfoldRepeats {
    ... musica ...
  }
  \midi { }
}
```

Quando si usa più di una voce, ciascuna voce deve avere tutte le ripetizioni ricopiate per generare un output MIDI corretto.

### Vedi anche

Guida alla notazione: Sezione 1.4 [Ripetizioni], pagina 152.

## 3.5.7 Mappatura dei canali MIDI

Quando genera un file MIDI da una partitura, LilyPond assegna automaticamente ogni nota a un canale MIDI, in cui essa dovrà essere suonata quando inviata a un dispositivo MIDI. Un canale MIDI mette a disposizione un certo numero di controlli per selezionare, per esempio, lo strumento da usare per riprodurre le note in quel canale o per chiedere al dispositivo MIDI di applicare vari effetti al suono prodotto nel canale. In ogni momento, ogni controllo di un canale MIDI può avere un solo valore assegnato (che tuttavia può essere modificato, per esempio, per passare a un altro strumento nel mezzo di un brano).

Lo standard MIDI supporta soltanto 16 canali per dispositivo MIDI. Questo limite al numero di canali limita anche il numero dei diversi strumenti che possono essere eseguiti in contemporanea.

LilyPond crea tracce MIDI separate per ogni rigo (o strumento o voce separati, a seconda del valore di `Score.midiChannelMapping`) e anche per ogni contesto lyrics. Non c'è limite al numero di tracce.

Per aggirare il limite del numero di canali MIDI, LilyPond supporta varie modalità per l'assegnazione dei canali MIDI, scelti attraverso la proprietà di contesto `Score.midiChannelMapping`. In ogni caso, se è necessario un numero di canali MIDI superiore al limite, i numeri canale assegnati ripartono da 0, causando possibili assegnazioni erronee degli strumenti di alcune note. Questa proprietà di contesto può essere impostata su uno dei seguenti valori:

`'staff`

> Assegna un canale MIDI separato a ogni rigo della partitura (comportamento predefinito). Tutte le note in tutte le voci di ogni rigo condivideranno lo stesso canale MIDI del rigo che le racchiude, e si trovano tutte nella stessa traccia MIDI.

Il limite dei 16 canali è applicato al numero totale dei contesti staff (rigo) e lyrics (testo vocale), anche se il testo vocale nel MIDI non occupa un canale MIDI.

`'instrument`

Assegna un canale MIDI separato a ogni strumento MIDI diverso specificato nella partitura. Ciò significa che tutte le note suonate con lo stesso strumento MIDI condivideranno lo stesso canale (e traccia) MIDI, anche se le note appartengono a voci o righi diversi.

In questo caso i contesti del testo vocale (lyrics) non contano nel calcolo del limite dei 16 canali MIDI (perché non saranno assegnati a uno strumento MIDI), dunque questa impostazione consente una migliore assegnazione dei canali MIDI quando il numero di righi e contesti lyrics in una partitura è superiore a 16.

`'voice`

Assegna un canale MIDI separato a ogni voce che abbia un nome unico tra le voci del rigo che le racchiude. Alle voci in righi diversi vengono sempre assegnati canali MIDI separati, ma due voci di uno stesso rigo condivideranno lo stesso canale MIDI se hanno lo stesso nome. `midiInstrument` e i vari controlli MIDI per gli effetti, essendo proprietà del contesto del rigo (staff), non possono essere impostati separatamente per ogni voce. La prima voce verrà suonata con lo strumento e gli effetti specificati per il rigo, mentre alle voci con un nome diverso da quello della prima saranno assegnati lo strumento e gli effetti predefiniti.

Nota bene: è possibile assegnare diversi strumenti e/o effetti a varie voci dello stesso rigo spostando `Staff_performer` dal contesto `Staff` al contesto `Voice` e lasciando `midiChannelMapping` sul valore predefinito `'staff` oppure impostandolo su `'instrument`; vedi il frammento in basso.

Per esempio, la mappatura predefinita dei canali MIDI di una partitura può essere modificata per usare l'impostazione `'instrument`:

```
\score {
  ...musica...
  \midi {
    \context {
      \Score
      midiChannelMapping = #'instrument
    }
  }
}
```

## Frammenti di codice selezionati
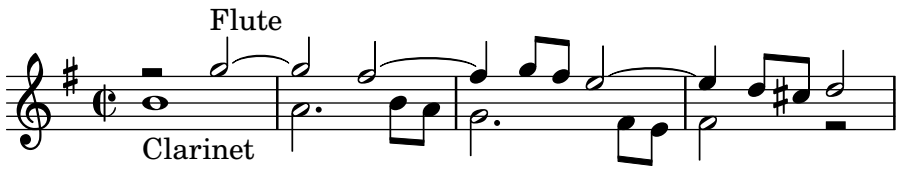
*Impostare l'output MIDI su un canale per voce*

Nella creazione del file di output MIDI, il comportamento predefinito prevede che ogni rigo sia assegnato a un canale MIDI, con tutte le voci del rigo amalgamate in un canale. Ciò diminuisce il rischio di esaurire i canali MIDI disponibili, dato che ce ne sono solo 16 per traccia.

Tuttavia, spostando `Staff_performer` nel contesto `Voice`, ogni voce in un rigo può avere il proprio canale MIDI, come è illustrato nell'esempio seguente: sebbene le voci siano sullo stesso rigo, vengono creati due canali MIDI, ciascuno con un diverso strumento MIDI (`midiInstrument`).

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = #"flute"
```

```
        \voiceOne
        \key g \major
        \time 2/2
        r2 g-"Flute" ~
        g fis ~
        fis4 g8 fis e2 ~
        e4 d8 cis d2
      }
      \new Voice \relative c'' {
        \set midiInstrument = #"clarinet"
        \voiceTwo
        b1-"Clarinet"
        a2. b8 a
        g2. fis8 e
        fis2 r
      }
    >>
    \layout { }
    \midi {
      \context {
        \Staff
        \remove "Staff_performer"
      }
      \context {
        \Voice
        \consists "Staff_performer"
      }
      \tempo 2 = 72
    }
  }
```




### 3.5.8 Proprietà di contesto per gli effetti MIDI

Le seguenti proprietà di contesto possono essere usate per applicare vari effetti MIDI alle note suonate sul canale MIDI associato col rigo, strumento MIDI o voce correnti (a seconda del valore della proprietà di contesto `Score.midiChannelMapping` e del contesto in cui si trova `Staff_performer`; vedi Sezione 3.5.7 [Mappatura dei canali MIDI], pagina 533).

La modifica di queste proprietà di contesto avrà effetto su tutte le note suonate sul canale dopo il cambio, tuttavia alcuni effetti potrebbero essere applicati anche a note che stanno già suonando (a seconda di come è stato implementato il dispositivo MIDI).

Sono supportate le seguenti proprietà di contesto:

`Staff.midiPanPosition`

> La posizione del panning regola il modo in cui il suono in un canale MIDI è distribuito tra gli altoparlanti stereo di sinistra e di destra. Questa proprietà di contesto accetta un numero compreso tra -1.0 (`#LEFT`) e 1.0 (`#RIGHT`); il valore -1.0 sposterà tutto il suono sull'altoparlante sinistro (rendendo muto quello destro), il valore 0.0

(`#CENTER`) distribuirà il suono in modo uniforme tra gli altoparlanti sinistro e destro, e il valore 1.0 sposterà tutto il suono sull'altoparlante destro. I valori intermedi tra -1.0 e 1.0 permettono di ottenere distribuzioni miste tra gli altoparlanti sinistro e destro.

`Staff.midiBalance`
: Il bilanciamento stereo di un canale MIDI. In modo simile al panning, questa proprietà di contesto accetta un numero compreso tra -1.0 (`#LEFT`) e 1.0 (`#RIGHT`). Varia il volume relativo inviato ai due altoparlanti stereo senza alterare la distribuzione dei segnali stereo.

`Staff.midiExpression`
: Livello dell'espressione (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Un dispositivo MIDI combina insieme il livello di espressione del canale MIDI con l'attuale livello di dinamica di una voce (stabilito dai comandi `\p` o `\ff`) per ottenere il volume totale di ogni nota nella voce. Il controllo dell'espressione può essere usato, per esempio, per implementare effetti di crescendo o decrescendo su singole note sostenute (non supportato automaticamente da LilyPond).

  La gamma del livello di espressione va da 0.0 (nessuna espressione, ovvero volume zero) a 1.0 (piena espressione).

`Staff.midiReverbLevel`
: Livello di riverbero (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Questa proprietà accetta numeri compresi tra 0.0 (nessun riverbero) e 1.0 (pieno effetto).

`Staff.midiChorusLevel`
: Livello del chorus (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Questa proprietà accetta numeri compresi tra 0.0 (nessun effetto chorus) e 1.0 (pieno effetto).

### Problemi noti e avvertimenti

Dato che i file MIDI non contengono veri dati audio, le modifiche in queste proprietà di contesto si traducono soltanto in richieste di modifica dei controlli del canale MIDI nei file MIDI generati. Se un particolare dispositivo MIDI (come un lettore MIDI) sia in grado di gestire queste richieste dipende interamente dall'implementazione del dispositivo: un dispositivo potrebbe scegliere di ignorare alcune o tutte le richieste. Dipende dall'implementazione del dispositivo MIDI anche il modo in cui un dispositivo MIDI interpreta valori diversi di questi controlli (generalmente lo standard MIDI corregge il comportamento solo alle estremità della gamma di valori disponibile per ogni controllo) e se un cambiamento nel valore di un controllo avrà effetto anche su note che stanno già suonando su quel canale MIDI.

Quando genera file MIDI, LilyPond trasformerà semplicemente le frazioni comprese in ciascuna gamma in valori di una gamma di numeri interi corrispondenti (0-127 per i 7-bit, 0-32767 per i 14 bit per i controlli del canale MIDI che supportano una migliore risoluzione), arrotondando i valori delle frazioni sul numero intero più vicino. I valori interi convertiti sono salvati così come sono nel file MIDI generato. Consultare la documentazione del proprio dispositivo MIDI per sapere come il dispositivo interpreta questi valori.

## 3.5.9 Miglioramento dell'output MIDI

L'output MIDI predefinito fornisce le fuzionalità basilari, ma può essere migliorato impostando gli strumenti MIDI, le proprietà del blocco `\midi` e/o usando lo script `articulate`.

### Lo script `articulate`

Per usare lo script `articulate` aggiungere il relativo comando `\include` all'inizio del file di input:

```
\include "articulate.ly"
```

Lo script crea file MIDI in cui la durata delle note è correttamente "ridimensionata" per rappresentare molte articolazioni e indicazioni di tempo. Tuttavia anche l'output grafico verrà modificato per corrispondere esattamente all'output MIDI.

```
\score {
  \articulate <<
    ... musica ...
  >>
  \midi { }
}
```

Il comando `\articulate` permette di elaborare le abbreviazioni (come i trilli e i gruppetti). L'elenco completo degli elementi supportati si trova nello script stesso. Vedi `ly/articulate.ly`.

### Vedi anche

Manuale di apprendimento: Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 4.2 [Formattazione della partitura], pagina 550.

File installati: `ly/articulate.ly`.

> **Nota:** Lo script `articulate` potrebbe accorciare la durata degli accordi, cosa che potrebbe non essere appropriata per alcuni tipi di strumento, come nel caso della musica per organo. Potrebbe essere ridotta anche la durata di note prive di articolazioni. Dunque, per consentire tutto ciò, limitare l'uso del comando `\articulate` a brevi segmenti di musica o modificare i valori delle variabili definite nello script `articulate` per compensare il comportamento di accorciamento della durata delle note.

## 3.6 Estrazione dell'informazione musicale

Oltre a creare l'output grafico e il MIDI, LilyPond può mostrare l'informazione musicale in forma testuale.

### 3.6.1 Mostrare la notazione LilyPond

La funzione musicale `\displayLilyMusic` permette di mostrare un'espressione musicale nella notazione di LilyPond. Per vedere l'output, di norma si esegue LilyPond da linea di comando. Per esempio,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

mostrerà

```
{ a,4 cis4 e4 fis4 g4 }
```

LilyPond stampa questi messaggi nella console insieme a tutti gli altri messaggi della compilazione. Per separare questi messaggi e salvare i risultati di `\displayLilyMusic`, redirigere l'output su un file:

```
lilypond file.ly >display.txt
```

Nota bene che LilyPond non soltanto mostra in console l'espressione musicale, ma la interpreta anche (infatti l'espressione musicale di `\displayLilyMusic` appare nell'output oltre a essere mostrata in console). Basta inserire `\displayLilyMusic` in file esistenti, senza doverne modificare le note di input, per ottenere informazioni su di essi. Per interpretare e mostrare una sezione musicale nella console ma, allo stesso tempo, toglierla dal file di output, usare il comando `\void`:

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
  c1
}
```

### 3.6.2 Mostrare le espressioni musicali Scheme

Vedi Sezione "Displaying music expressions" in *Estendere*.

### 3.6.3 Salvare eventi musicali in un file

Gli eventi musicali possono essere salvati in file diversi, un file per ogni rigo, includendo un file nella partitura principale:

```
\include "event-listener.ly"
```

Così verranno creati uno o più file `NOMEFILE-NOMERIGO.notes` o `NOMEFILE-unnamed-staff.notes` per ogni rigo. Se ci sono vari righi senza nome, gli eventi di tutti i righi saranno mischiati insieme nello stesso file. I file hanno una struttura di questo tipo:

```
0.000   note     57       4   p-c 2 12
0.000   dynamic  f
0.250   note     62       4   p-c 7 12
0.500   note     66       8   p-c 9 12
0.625   note     69       8   p-c 14 12
0.750   rest     4
0.750   breathe
```

La sintassi prevede una riga delimitata da tabulatori, con due campi fissi su ogni riga seguiti da parametri opzionali.

```
tempo  tipo  ...parametri...
```

Questa informazione può essere letta facilmente da altri programmi, per esempio da uno script python, e può essere molto utile per ricercatori che desiderano fare delle analisi musicali o degli esperimenti di *playback* con LilyPond.

## Problemi noti e avvertimenti

Non tutti gli eventi musicali di lilypond sono supportati da `event-listener.ly`, che vuole essere una semplice "prova di concetto" ben congeniata. Se alcuni eventi che si vorrebbero vedere non sono inclusi, copiare `event-listener.ly` nella propria directory lilypond e modificare il file in modo che produca l'informazione desiderata.

# 4 Gestione dello spazio

La formattazione globale del foglio è determinata da tre fattori: la formattazione della pagina, le interruzioni di linea e la spaziatura. Ciascun fattore influenza l'altro. La scelta della spaziatura determina la densità con cui vengono disposti i sistemi musicali, che a sua volta influenza la scelta delle interruzioni di linea e quindi infine quante pagine occupa un brano.

Generalmente, questa procedura si svolge in quattro stadi. Inizialmente vengono scelte le distanze flessibili ('springs'), in base alle durate. Poi vengono tentate tutte le possibili combinazioni per le interruzioni di linea e viene calcolato un punteggio 'negativo' per ciascuna di esse. Quindi viene calcolata l'altezza di ogni possibile sistema. Infine viene scelta una combinazione di interruzioni di linea e di interruzioni di pagina che assicuri che la spaziatura verticale e quella orizzontale non siano né troppo compresse né troppo allungate.

Due tipi di blocchi possono contenere le impostazioni di formattazione: `\paper {...}` e `\layout {...}`. Il blocco `\paper` contiene le impostazioni di formattazione della pagina da applicare a tutte le partiture di un libro o di una parte, come l'altezza del foglio o se stampare o meno i numeri di pagina, etc. (vedi Sezione 4.1 [Formattazione della pagina], pagina 539). Il blocco `\layout` contiene le impostazioni di formattazione della partitura, come il numero di sistemi da usare o lo spazio tra i gruppi di righi, etc. (vedi Sezione 4.2 [Formattazione della partitura], pagina 550).

## 4.1 Formattazione della pagina

Questa sezione tratta le opzioni di formattazione della pagina per il blocco `\paper`.

### 4.1.1 Il blocco \paper

I blocchi `\paper` possono essere posizionati in tre punti diversi, in modo da formare una gerarchia discendente di blocchi `\paper`:

- All'inizio del file di input, prima di tutti i blocchi `\book`, `\bookpart` e `\score`.
- Dentro un blocco `\book` ma fuori da tutti i blocchi `\bookpart` e `\score` in esso racchiusi.
- Dentro un blocco `\bookpart` ma fuori da tutti i blocchi `\score` in esso racchiusi.

Non è possibile inserire un blocco `\paper` in un blocco `\score`.

I valori dei campi vengono filtrati dall'alto verso il basso attraverso questa gerarchia: i valori impostati nei livelli più alti della gerarchia persistono finché non sono sovrascritti da un valore impostato in un livello più basso.

Vari blocchi `\paper` possono apparire in ognuno di questi livelli, per esempio come parti di vari file inclusi con `\include`. In questo caso, i campi di ciascun livello vengono combinati e i valori riscontrati per ultimi avranno la precedenza in caso di campi duplicati.

Le impostazioni che possono apparire in un blocco `\paper` comprendono:

- la funzione Scheme `set-paper-size`,
- le variabili `\paper` per personalizzare la formattazione della pagina e
- le definizioni di markup usate per personalizzare la formattazione di intestazioni, piè di pagina e titoli.

La funzione `set-paper-size` è trattata nella prossima sezione, Sezione 4.1.2 [Formato carta e ridimensionamento automatico], pagina 540. Le variabili `\paper` che si occupano della formattazione della pagina sono trattate in sezioni successive. Le definizioni di markup relative a intestazioni, piè di pagina e titoli sono trattate in Sezione 3.2.2 [Titoli intestazioni e piè di pagina personalizzati], pagina 493.

La maggior parte delle variabili `\paper` funzionano soltanto in un blocco `\paper`. Le poche che funzionano anche in un blocco `\layout` sono elencate in Sezione 4.2.1 [Il blocco layout], pagina 550.

Se non indicato diversamente, tutte le variabili `\paper` che corrispondono a distanze sulla pagina sono misurate in millimetri, a meno che un'unità di misura diversa non sia specificata dall'utente. Per esempio, la seguente dichiarazione imposta `top-margin` su dieci millimetri:

```
\paper {
  top-margin = 10
}
```

Per impostarla su `0.5` pollici, usare il suffisso di unità `\in` (inch = pollice):

```
\paper {
  top-margin = 0.5\in
}
```

I suffissi di unità disponibili sono `\mm`, `\cm`, `\in` e `\pt`. Queste unità sono semplici valori utili per convertire dai millimetri e sono definite in `ly/paper-defaults-init.ly`. Solo per chiarezza, quando si usano i millimetri, di solito si usa il suffisso `\mm`, anche se non è tecnicamente necessario.

È anche possibile definire i valori di `\paper` con Scheme. L'equivalente Scheme dell'esempio precedente è:

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

### Vedi anche

Guida alla notazione: Sezione 4.1.2 [Formato carta e ridimensionamento automatico], pagina 540, Sezione 3.2.2 [Titoli intestazioni e piè di pagina personalizzati], pagina 493, Sezione 4.2.1 [Il blocco layout], pagina 550.

File installati: `ly/paper-defaults-init.ly`.

## 4.1.2 Formato carta e ridimensionamento automatico

### Impostare il formato carta

'A4' è il valore predefinito quando non viene impostato esplicitamente alcun formato carta. Esistono due funzioni che permettono di cambiare formato:

`set-default-paper-size`

```
#(set-default-paper-size "quarto")
```

che deve sempre trovarsi nel livello superiore, e

`set-paper-size`

```
\paper {
  #(set-paper-size "tabloid")
}
```

che deve sempre trovarsi in un blocco `\paper`.

La funzione `set-default-paper-size`, se usata nel livello superiore, deve precedere qualsiasi blocco `\paper`. `set-default-paper-size` imposta il formato di tutte le pagine, mentre `set-paper-size` imposta il formato soltanto di quelle pagine a cui è applicato il blocco `\paper`. Per esempio, se il blocco `\paper` si trova all'inizio del file, applicherà il formato a tutte le pagine. Se il blocco `\paper` si trova dentro un blocco `\book`, il formato verrà applicato a quel libro soltanto.

Quando si usa la funzione `set-paper-size`, questa deve essere posta *prima* di qualsiasi altra funzione usata nello stesso blocco `\paper`. Vedi [Ridimensionamento automatico al formato carta], pagina 541.

I formati carta sono definiti in `scm/paper.scm`, e sebbene sia possibile aggiungere formati personalizzati in questo file, tali aggiunte verrebbero sovrascritte da successivi aggiornamenti del software. I formati disponibili sono elencati in Sezione A.5 [Formati carta predefiniti], pagina 674.

È tuttavia possibile aggiungere un formato personalizzato nel file di input per poi utilizzarlo con `set-default-paper-size` o `set-paper-size`:

```
#(set! paper-alist (cons '("mio formato" . (cons (* 15 in) (* 3 in))) paper-alist))

\paper {
  #(set-paper-size "mio formato")
}
```

Si può usare qualsiasi unità di misura: `in` (inch, o pollici), `cm` (centimetri) e `mm` (millimetri).

Aggiungendo il simbolo `'landscape` alla funzione del formato, le pagine vengono ruotate di 90 gradi e le linee occupano il maggior spazio a disposizione.

```
#(set-default-paper-size "a6" 'landscape)
```

Appendendo '`landscape`' (orizzontale) al nome del formato, è possibile scambiare le dimensioni della carta *senza* ruotare la stampa (come quando si stampa in formato cartolina o si creano dei file grafici da includere invece di un documento indipendente):

```
#(set-default-paper-size "a6landscape")
```

Quando il formato termina con un esplicito '`landscape`' (orizzontale) o '`portrait`' (verticale), la presenza di un simbolo `'landscape` influisce *solo* sull'orientamento della stampa, non sul formato usato per la formattazione.

### Vedi anche

Guida alla notazione: [Ridimensionamento automatico al formato carta], pagina 541, Sezione A.5 [Formati carta predefiniti], pagina 674.

File installati: `scm/paper.scm`.

## Ridimensionamento automatico al formato carta

Se il formato viene cambiato con una delle funzioni Scheme (`set-default-paper-size` o `set-paper-size`), i valori di diverse variabili `\paper` sono automaticamente ricalcolati in base alla nuova dimensione. Per aggirare il ridimensionamento automatico di una certa variabile, impostare la variabile dopo aver impostato il formato. Fare attenzione al fatto che il ridimensionamento automatico non viene attivato se si impostano le variabili `paper-height` o `paper-width`, anche se `paper-width` può influenzare altri valori (ma questo è un argomento diverso dal ridimensionamento ed è trattato in seguito). Le funzioni `set-default-paper-size` e `set-paper-size` sono descritte in [Impostare il formato carta], pagina 540.

Le dimensioni verticali interessate dal ridimensionamento automatico sono `top-margin` e `bottom-margin` (vedi ⟨undefined⟩ [Variabili fisse della spaziatura verticale di `\paper`], pagina ⟨undefined⟩). Le dimensioni orizzontali interessate dal ridimensionamento automatico sono `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent` e `short-indent` (vedi ⟨undefined⟩ [Variabili della spaziatura orizzontale di `\paper`], pagina ⟨undefined⟩).

I valori predefiniti di queste dimensioni sono impostati in `ly/paper-defaults-init.ly` e salvati in variabili interne chiamate `top-margin-default`, `bottom-margin-default`, etc. Questi valori si riferiscono al formato predefinito `a4`. Per riferimento, nel formato `a4` il valore di `paper-height` è `297\mm` e quello di `paper-width` è `210\mm`.

### Vedi anche

Guida alla notazione: ⟨undefined⟩ [Variabili fisse della spaziatura verticale di `\paper`], pagina ⟨undefined⟩, ⟨undefined⟩ [Variabili della spaziatura orizzontale di `\paper`], pagina ⟨undefined⟩.

File installati: `ly/paper-defaults-init.ly`, `scm/paper.scm`.

## 4.1.3 Variabili \paper della spaziatura verticale fissa

> **Nota:** Alcune dimensioni definite nel blocco `\paper` sono ridimensionate automaticamente in base al formato carta, portando a possibili risultati imprevisti. Vedi [Ridimensionamento automatico al formato carta], pagina 541.

I valori predefiniti (prima del ridimensionamento) sono definiti in `ly/paper-defaults-init.ly`.

`paper-height`
: L'altezza della pagina (valore predefinito: non impostata). Il ridimensionamento automatico di alcune dimensioni verticali non è influenzato da questa.

`top-margin`
: Il margine tra il punto più alto della pagina e il punto più alto dell'area di stampa. Se il formato viene modificato, il valore predefinito di questa dimensione viene ridimensionato di conseguenza.

`bottom-margin`
: Il margine tra il punto più basso dell'area di stampa e il punto più basso della pagina. Se il formato viene modificato, il valore predefinito di questa dimensione viene ridimensionato di conseguenza.

`ragged-bottom`
: Se impostato su vero, i sistemi avranno la loro naturale spaziatura. Non saranno giustificati, ovvero non saranno né compressi né allungati verticalmente per stare nella pagina.

`ragged-last-bottom`
: Se impostato su falso, l'ultima pagina, e l'ultima pagina di ogni sezione creata con un blocco `\bookpart`, saranno giustificate verticalmente come nelle pagine precedenti.

### Vedi anche

Guida alla notazione: [Ridimensionamento automatico al formato carta], pagina 541.

File installati: `ly/paper-defaults-init.ly`.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### Problemi noti e avvertimenti

I titoli (definiti nel blocco `\header`) sono considerati un sistema, dunque `ragged-bottom` e `ragged-last-bottom` aggiungeranno spazio tra i titoli e il primo sistema della partitura.

Formati carta definiti esplicitamente sovrascriveranno qualsiasi impostazione di margine superiore o inferiore definito dall'utente.

## 4.1.4 Variabili \paper della spaziatura verticale flessibile

Nella maggior parte dei casi è preferibile che le distanze verticali tra certi elementi (come margini, titoli, sistemi e partiture separate) siano flessibili, così che siano ben compresse o allungate a

seconda della situazione. Sono disponibili diverse variabili `\paper` (elencate sotto) per regolare il comportamento di allungamento di queste dimensioni.

Nota bene che le variabili `\paper` trattate in questa sezione non regolano la spaziatura dei righi dei singoli sistemi. La spaziatura interna ai sistemi è controllata dalle proprietà del grob, solitamente inserite in un blocco `\score` o `\layout` e non in un blocco `\paper`. Vedi Sezione 4.4.1 [Spaziatura verticale flessibile all'interno dei sistemi], pagina 561.

### Struttura delle liste associative flessibili della spaziatura verticale

Ognuna delle variabili `\paper` flessibili della spaziatura verticale è una lista associativa (alist) composta da quattro *elementi*:

- `basic-distance` – la distanza verticale, misurata in spazi rigo, tra i *punti di riferimento* dei due elementi, quando non si verificano collisioni e non sono attivi allungamenti o compressioni. Il punto di riferimento di un markup (un titolo o un markup di un livello superiore) è il suo punto più alto, mentre quello di un sistema è il centro verticale del più vicino `StaffSymbol`, anche se una linea diversa da un rigo (come un contesto `Lyrics`) si trova nel mezzo. Valori di `basic-distance` inferiori a quelli di `padding` o `minimum-distance` non hanno senso, dato che la distanza risultante non sarà mai inferiore a quella definita per `padding` o `minimum-distance`.
- `minimum-distance` – la minima distanza verticale consentita, misurata in spazi rigo, tra i punti di riferimento dei due elementi, quando la compressione è attiva. Valori di `minimum-distance` inferiori a quelli di `padding` non hanno senso, perché la distanza risultante non sarà mai inferiore a quella definita per `padding`.
- `padding` – la quantità minima di spazio verticale libero tra i profili (*skyline*) dei due elementi, misurata in spazi rigo.
- `stretchability` – una misura, priva di unità di misura, della relativa propensione della dimensione a allungarsi. Se pari a zero, la distanza non si allungherà (a meno che non ci siano delle collisioni). Se positiva, il significato del valore di `stretchability` di una certa dimensione sta nella sua relazione con i valori di `stretchability` delle altre dimensioni. Per esempio, se una dimensione ha un valore di `stretchability` doppio rispetto a un'altra, si allungherà il doppio. I valori devono essere non negativi e finiti. Il valore `+inf.0` causa un errore di programmazione e viene ingorato, ma si può usare `1.0e7` per ottenere un'elasticità allungabile quasi all'infinito. Se non impostata, il valore predefinito equivale a quello di `basic-distance`. Nota che la propensione della dimensione a *comprimersi* non può essere impostata direttamente dall'utente ed è uguale a (`basic-distance` − `minimum-distance`).

Se in una pagina lo spazio verticale non è giustificato (ovvero con `ragged-bottom` impostato su vero) la distanza risultante è data dal valore più grande assegnato a una di queste tre proprietà:

- `basic-distance`,
- `minimum-distance` e
- `padding` con l'aggiunta della più piccola distanza necessaria a eliminare collisioni.

In caso di partiture con molte pagine in cui l'ultima pagina non è giustificata (ovvero con `ragged-last-bottom` impostato su vero), l'ultima pagina usa la stessa spaziatura della pagina precedente, purché ci sia abbastanza spazio.

I metodi specifici per modificare le liste associative (alist) sono trattati in Sezione 5.3.7 [Modifica delle liste associative], pagina 627. L'esempio seguente illustra i due modi in cui queste possono essere modificate. La prima dichiarazione aggiorna un elemento-valore individualmente, mentre la seconda ridefinisce completamente la variabile:

```
\paper {
  system-system-spacing.basic-distance = #8
  score-system-spacing =
```

```
        #'((basic-distance . 12)
           (minimum-distance . 6)
           (padding . 1)
           (stretchability . 12))
    }
```

### Elenco delle variabili \paper flessibili della spaziatura verticale

I nomi di queste variabili hanno il formato *superiore-inferiore*-spacing, dove *superiore* e *inferiore* indicano gli elementi di cui stabilire la distanza. Ogni distanza viene misurata tra i punti di riferimento dei due elementi (vedi la descrizione precedente della struttura della lista associativa). Nei nomi di queste variabili il termine 'markup' si riferisce sia ai *titoli* (bookTitleMarkup o scoreTitleMarkup) sia ai *markup di livello superiore* (vedi Sezione 3.1.5 [Struttura del file], pagina 484). Tutte le distanze sono misurate in spazi rigo.

Le impostazioni predefinite sono definite in ly/paper-defaults-init.ly.

markup-system-spacing
: la distanza tra il testo (titolo o markup di livello superiore) e il sistema che lo segue.

score-markup-spacing
: la distanza tra l'ultimo sistema di un brano e il testo (titolo o markup di livello superiore) che lo segue.

score-system-spacing
: la distanza tra l'ultimo sistema di un brano e il primo sistema del brano successivo, quando non c'è alcun testo (titolo o markup di livello superiore) tra di loro.

system-system-spacing
: la distanza tra due sistemi di uno stesso brano.

markup-markup-spacing
: la distanza tra due testi (titoli o markup di livello superiore).

last-bottom-spacing
: la distanza tra l'ultimo sistema o l'ultimo markup di livello superiore di una pagina e la fine dello spazio stampabile (ovvero la parte superiore del margine inferiore).

top-system-spacing
: la distanza tra l'inizio dello spazio stampabile (ovvero la parte inferiore del margine superiore) e il primo sistema di una pagina, quando non c'è alcun testo (titolo o markup di livello superiore) tra di loro.

top-markup-spacing
: la distanza tra l'inizio dello spazio stampabile (ovvero la parte inferiore del margine superiore) e il primo testo (titolo o markup di livello superiore) di una pagina, quando non c'è alcun sistema tra i due.

### Vedi anche

Guida alla notazione: Sezione 4.4.1 [Spaziatura verticale flessibile all'interno dei sistemi], pagina 561.

File installati: ly/paper-defaults-init.ly.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.1.5 Variabili \paper della spaziatura orizzontale

**Nota:** Alcune dimensioni in `\paper` sono ridimensionate automaticamente in base al formato carta, producendo talvolta un risultato inatteso. Vedi [Ridimensionamento automatico al formato carta], pagina 541.

### Variabili `\paper` per larghezze e margini

I valori predefiniti (prima del ridimensionamento) non elencati qui si trovano in `ly/paper-defaults-init.ly`.

`paper-width`
: Larghezza della pagina (non impostata inizialmente). `paper-width`, pur non avendo effetto sul ridimensionamento automatico di alcune dimensioni orizzontali, influenza la variabile `line-width`. Se sono impostate sia `paper-width` che `line-width`, saranno modificate anche `left-margin` e `right-margin`. Vedi anche `check-consistency`.

`line-width`
: Se specificata in un blocco `\paper` definisce la lunghezza orizzontale disponibile per i righi musicali nei sistemi non indentati. Se non specificata, la larghezza della linea (`line-width`) del foglio è determinata dalla sottrazione `(paper-width − left-margin − right-margin)`. Se `line-width` è specificata nel blocco `\paper` e entrambi i margini, `left-margin` e `right-margin`, non lo sono, questi saranno aggiornati in modo da centrare i sistemi sulla pagina automaticamente. Vedi anche `check-consistency`.

  Le variabili `line-width` per i singoli brani possono essere specificate nel blocco `\layout` di ogni brano, ovvero dentro un blocco `\score`. Questi valori regolano la larghezza delle linee prodotte brano per brano. Se `line-width` non viene specificata per un brano, il suo valore predefinito coincide con la larghezza della linea del foglio, ovvero col valore della variabile `line-width` del blocco `\paper`. Impostando `line-width` per un brano in particolare, ovvero all'interno di un blocco `\score`, i margini del foglio non vengono modificati. Le linee del rigo, la cui lunghezza è determinata dalla variabile `line-width` del brano, sono allineate a sinistra all'interno dello spazio definito dalla variabile `line-width` del foglio. Se le variabili `line-width` del brano e del foglio sono uguali, le linee del rigo si estenderanno esattamente dal margine sinistro al margine destro, ma se quella del brano è maggiore di quella del foglio le linee del rigo andranno oltre il margine destro.

`left-margin`
: Margine tra il bordo sinistro della pagina e l'inizio delle linee del rigo nei sistemi non indentati. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Se `left-margin` non è impostato, e sono impostati sia `line-width` che `right-margin`, allora `left-margin` equivale alla differenza `(paper-width − line-width − right-margin)`. Se è impostata soltanto `line-width`, entrambi i margini sono impostati su `((paper-width − line-width) / 2)`, e di conseguenza i sistemi sono centrati sulla pagina. Vedi anche `check-consistency`.

`right-margin`
: Margine tra il bordo destro della pagina e la fine delle linee del rigo in sistemi giustificati. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Se `right-margin` non è impostato, e sono impostati sia `line-width` che `left-margin`, allora `right-margin` equivale a `(paper-width − line-width − left-margin)`. Se è impostata soltanto `line-width`, entrambi i margini sono impostati su `((paper-width − line-width) / 2)`, e di conseguenza i sistemi sono centrati sulla pagina. Vedi anche `check-consistency`.

`check-consistency`
: Se impostata su vero (valore predefinito), appare un avvertimento nella console se la somma di `left-margin`, `line-width` e `right-margin` non corrisponde esattamente a `paper-width`, e a ciascuna di queste variabili (eccetto `paper-width`) sarà assegnato il suo valore predefinito (ridimensionato in base al formato, se richiesto). Se impostato su falso, qualsiasi incongruenza viene ignorata e i sistemi potrebbero andare oltre il bordo della pagina.

`ragged-right`
: Se impostato su vero, i sistemi non occuperanno interamente la larghezza della linea, ovvero non saranno giustificati. Termineranno invece alla loro lunghezza orizzontale naturale. Valori predefiniti: `#t` (true = vero) per i brani con un unico sistema, e `#f` (false = falso) per i brani con due o più sistemi. Questa variabile può essere impostata anche in un blocco `\layout`.

`ragged-last`
: Se impostata su vero, l'ultimo sistema del brano non occuperà interamente la larghezza della linea, ovvero non sarà giustificato. Terminerà invece alla sua lunghezza orizzontale naturale. Valore predefinito: `#f` (false = falso). Questa variabile può essere impostata anche in un blocco `\layout`.

## Vedi anche

Guida alla notazione: [Ridimensionamento automatico al formato carta], pagina 541.

File installati: `ly/paper-defaults-init.ly`.

## Problemi noti e avvertimenti

Formati carta definiti esplicitamente sovrascriveranno qualsiasi margine destro o sinistro definito dall'utente.

## Variabili \paper per la modalità due pagine per foglio

I valori predefiniti (prima del ridimensionamento) sono definiti in `ly/paper-defaults-init.ly`.

`two-sided`
: Se impostata su vero, vengono usati `inner-margin`, `outer-margin` e `binding-offset` per determinare i margini a seconda che il numero di pagina sia dispari o pari. Questa variabile sovrascrive `left-margin` e `right-margin`.

`inner-margin`
: Margine del lato interno di tutte le pagine se queste fanno parte di un libro. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

`outer-margin`
: È il margine del lato esterno di tutte le pagine se queste fanno parte di un libro. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

`binding-offset`
: Quantità di spazio da aggiungere a `inner-margin` per assicurarsi che niente sia nascosto dalla rilegatura. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

### Vedi anche

Guida alla notazione: [Ridimensionamento automatico al formato carta], pagina 541.

File installati: `ly/paper-defaults-init.ly`.

### Variabili \paper per spostamenti e indentazioni

I valori predefiniti (prima del ridimensionamento) non elencati qui sono visibili in `ly/paper-defaults-init.ly`.

`horizontal-shift`
: È la quantità di spazio di cui spostare a destra tutti i sistemi (inclusi i titoli e i separatori dei sistemi). Valore predefinito: `0.0\mm`.

`indent`
: Livello di indentazione del primo sistema di un brano. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Lo spazio compreso in `line-width` disponibile per il primo sistema è ridotto di questa quantità. `indent` può essere specificato anche in blocchi `\layout` per impostare l'indentazione brano per brano.

`short-indent`
: Livello di indentazione di tutti i sistemi di un brano eccetto il primo sistema. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Lo spazio compreso in `line-width` disponibile per tutti i sistemi tranne il primo è ridotto di questa quantità. `short-indent` può essere specificato anche in blocchi `\layout` per impostare le brevi indentazioni brano per brano.

### Vedi anche

Guida alla notazione: [Ridimensionamento automatico al formato carta], pagina 541.

File installati: `ly/paper-defaults-init.ly`.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.1.6 Altre variabili di \paper

### Variabili di \paper per l'interruzione di linea

`max-systems-per-page`
: Il numero massimo di sistemi che saranno disposti in una pagina. Attualmente ciò è supportanto soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

`min-systems-per-page`
: Il numero minimo di sistemi che saranno disposti in una pagina. Ciò potrebbe riempire troppo le pagine, se il numero è troppo grande. Attualmente ciò è supportanto soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

`systems-per-page`
: Il numero di sistemi da disporre in una pagina. Attualmente ciò è supportanto soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

`system-count`
: Il numero di sistemi da usare per un brano. Valore predefinito: non impostato. Questa variabile può essere impostata anche in un blocco `\layout`.

### Vedi anche

Guida alla notazione: Sezione 4.3.1 [Interruzioni di linea], pagina 554.

### Variabili di \paper per l'interruzione di pagina

I valori predefiniti non elencati qui sono visibili in `ly/paper-defaults-init.ly`

`page-breaking`
: L'algoritmo di interruzione di pagina da usare. Le opzioni sono `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-page-breaking`, `ly:one-line-breaking`, `ly:one-line-auto-height-breaking` e `ly:optimal-breaking` (predefinito).

`page-breaking-system-system-spacing`
: Induce il sistema di interruzione di pagina a credere che `system-system-spacing` sia impostato in modo diverso dal modo in cui è impostato realmente. Per esempio, se `page-breaking-system-system-spacing.padding` è impostato su un valore molto più grande di `system-system-spacing.padding`, il sistema di interruzione di pagina disporrà meno sistemi su ciascuna pagina. Valore predefinito: non impostato.

`page-count`
: Il numero di pagine da usare per un brano, non impostato.

Le variabili seguenti sono attive soltanto quando `page-breaking` è impostato su `ly:page-turn-breaking`. Le interruzioni di pagina vengono quindi scelte per minimizzare il numero di voltate di pagina. Dato che le voltate di pagina si rendono necessarie quando si passa da una pagina con numero dispari a una con numero pari, verrà solitamente favorita una formattazione in cui l'ultima pagina abbia un numero dispari. I punti preferiti per le voltate di pagina possono essere indicati manualmente col comando `\allowPageTurn` o automaticamente includendo l'incisore `Page_turn_engraver` (vedi [Voltata di pagina ottimale], pagina 560).

Se le scelte disponibili per avere voltate di pagina adeguate sono insufficienti, LilyPond potrebbe inserire una pagina vuota all'interno di un brano o tra i brani (se ce ne sono almeno due), oppure potrebbe terminare un brano su una pagina pari. Aumentando i valori delle seguenti tre variabili diminuisce la probabilità che queste entrino in azione.

I valori sono penalità: maggiore il valore, minore la probabilità che si verifichi l'azione associata rispetto alle altre scelte.

`blank-page-penalty`
: La penalità per avere una pagina vuota nel mezzo di un brano. Se il valore di `blank-page-penalty` è grande ed è selezionato `ly:page-turn-breaking`, sarà meno probabile che LilyPond insersica una pagina nel mezzo del brano. Aggiungerà invece maggior spazio tra i sistemi per occupare la pagina vuota e quella successiva. Valore predefinito: 5.

`blank-last-page-penalty`
: La penalità per terminare il brano su una pagina pari. Se il valore di `blank-last-page-penalty` è grande ed è selezionato `ly:page-turn-breaking`, sarà meno probabile che LilyPond produca una partitura in cui l'ultima pagina sia pari. Regolerà invece la spaziatura in modo da usare una pagina in più o in meno. Valore predefinito: 0.

`blank-after-score-page-penalty`
: La penalità per avere una pagina vuota dopo la fine di un brano e prima di quello successivo. Il valore predefinito è inferiore a `blank-page-penalty`, in modo che siano inserite preferibilmente pagine vuote al termine di un brano piuttosto che nel mezzo di un brano. Valore predefinito: 2.

## Vedi anche

Guida alla notazione: Sezione 4.3.2 [Interruzioni di pagina], pagina 557, [Interruzione di pagina ottimale], pagina 559, [Voltata di pagina ottimale], pagina 560, [Interruzione di pagina minimale], pagina 559, [Interruzione di pagina di una pagina], pagina 559, [Interruzione di pagina su una linea], pagina 559, [Interruzione di pagina su una linea con altezza automatica], pagina 559.

File installati: `ly/paper-defaults-init.ly`.

## Variabili di \paper per la numerazione delle pagine

I valori predefiniti non elencati qui sono visibili in `ly/paper-defaults-init.ly`

`auto-first-page-number`
: L'algoritmo di interruzione di pagina si comporta diversamente a seconda che il numero della prima pagina sia dispari o pari. Se questa variabile è impostata su vero, l'algoritmo di interruzione di pagina deciderà se iniziare con un numero dispari o un numero pari. Quindi il numero della prima pagina resterà lo stesso o aumenterà di uno. Valore predefinito: `#f` (falso).

`first-page-number`
: Il valore del numero di pagina della prima pagina.

`print-first-page-number`
: Se impostato su vero, appare il numero di pagina sulla prima pagina.

`print-page-number`
: Se impostato su falso, non apparirano i numeri di pagina.

`page-number-type`
: Il tipo di numero usato per numerare le pagine. Le opzioni possibili sono `roman-lower`, `roman-upper` e `arabic`. Valore predefinito: `'arabic`.

## Vedi anche

File installati: `ly/paper-defaults-init.ly`.

## Problemi noti e avvertimenti

I numeri di pagina dispari sono sempre sulla destra. Se la musica deve iniziare a pagina 1, ci deve essere una pagina vuota sulla seconda di copertina in modo che pagina 1 sia sul lato destro.

## Svariate variabili di \paper

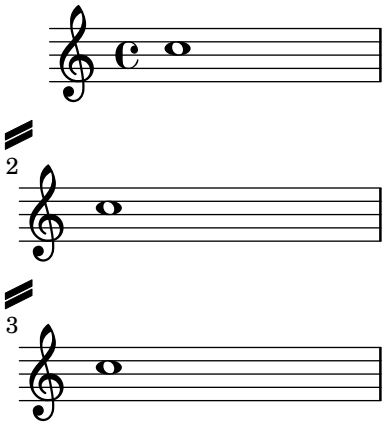
`page-spacing-weight`
: L'importanza della spaziatura (verticale) e della linea (orizzontale) della pagina. Valori più grandi renderanno la spaziatura della pagina più importante. Valore predefinito: `10`.

`print-all-headers`
: Se impostato su vero, appariranno nell'output tutte le intestazioni di ciascun blocco `\score`. Di norma appaiono soltanto le variabili `piece` e `opus`. Valore predefinito: `#f`.

`system-separator-markup`
: Un oggetto markup inserito tra i sistemi, spesso usato per le partiture orchestrali. Valore predefinito: non impostato. È disponibile il comando markup `\slashSeparator`, definito in `ly/titling-init.ly`, che fornisce un ragionevole valore predefinito, per esempio:

```
#(set-default-paper-size "a8")
```

```
\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}
```




### Vedi anche

File installati: `ly/titling-init.ly`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### Problemi noti e avvertimenti

L’intestazione di pagina predefinita pone sulla stessa linea il numero di pagina e il campo `instrument` del blocco `\header`.

## 4.2 Formattazione della partitura

Questa sezione tratta le opzioni di formattazione della partitura disponibili nel blocco `\layout`.

### 4.2.1 Il blocco `\layout`

Mentre il blocco `\paper` contiene le impostazioni relative alla formattazione della pagina per l’intero documento, il blocco `\layout` contiene impostazioni di formattazione specifica di un brano. Per impostare globalmente le opzioni di formattazione del brano, inserirle in un blocco `\layout` del livello superiore. Per impostare le impostazioni di formattazione per un singolo brano, inserirle in un blocco `\layout` racchiuso in un blocco `\score`, dopo la musica. Le impostazioni che possono apparire in un blocco `\layout` includono:

- la funzione Scheme `layout-set-staff-size`,
- le modifiche di contesto nei blocchi `\context` e
- le variabili `\paper` che agiscono sulla formattazione del brano.

La funzione `layout-set-staff-size` è spiegata nella prossima sezione, Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552. Le modifiche di contesto sono trattate in un capitolo

separato; vedi Sezione 5.1.4 [Modifica dei componenti aggiuntivi di un contesto], pagina 600, e Sezione 5.1.5 [Modifica delle impostazioni predefinite di un contesto], pagina 602.

Le variabili `\paper` che possono apparire in un blocco `\layout`, con valori predefiniti presi dal blocco `\paper`, sono:

- `line-width`, `ragged-right` e `ragged-last` (vedi [variabili `\paper` per larghezze e margini], pagina 545)
- `indent` e `short-indent` (vedi [Variabili `\paper` per spostamenti e indentazioni], pagina 547)
- `system-count` (vedi [variabili di `\paper` per l'interruzione di linea], pagina 547)

Ecco un esempio di blocco `\layout`:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGrouper.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Si possono inserire molteplici blocchi `\layout` come espressioni di livello superiore. Ciò può essere utile, per esempio, se impostazioni diverse sono salvate in file separati e incluse facoltativamente. Internamente, quando un blocco `\layout` viene elaborato, viene fatta una copia della configurazione del blocco `\layout` corrente, poi qualsiasi modifica apportata nel blocco viene applicata e il risultato viene salvato come la nuova configurazione corrente. Dalla prospettiva dell'utente, i blocchi `\layout` sono combinati, ma in situazioni di conflitto (ovvero quando viene modificata la stessa proprietà in blocchi diversi) hanno precedenza le definizioni più recenti.

Per esempio, se questo blocco

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```

viene posto dopo quello dell'esempio precedente, le sovrascritture di `'padding` e `'color` per `TextScript` sono combinate, mentre quella più recente di `'thickness` per `Glissando` sostituisce (o nasconde) quella precedente.

I blocchi `\layout` possono essere assegnati a delle variabili per poterli riusare successivamente, ma il modo in cui ciò funziona è leggermente ma significativamente diverso rispetto a scriverle davvero.

Se una variabile è definita in questo modo,

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
  }
}
```

manterrà la configurazione attuale di `\layout` con l'aggiunta della sovrascrittura `NoteHead.font-size`, ma questa combinazione *non* è salvata come la nuova configurazione corrente. Attenzione al fatto che la "configurazione corrente" viene letta quando la variabile è definita e non quando viene usata, dunque il contenuto della variabile dipende dalla sua posizione nel file di input.

La variabile può quindi essere usata all'interno di un altro blocco `\layout`, per esempio:

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

Un blocco `\layout` che contiene una variabile, come nell'esempio precedente, *non* copia la configurazione corrente bensì usa il contenuto di `\layoutVariable` come configurazione di base per altre aggiunte. Ciò significa che qualsiasi modifica definita tra la definizione della variabile e il momento in cui essa viene usata è perduta.

Se `layoutVariable` è definita (o inclusa con `\include`) subito prima di essere usata, il suo contenuto comprende soltanto la configurazione corrente più le sovrascritture definite al suo interno. Quindi nell'esempio precedente che illustra l'uso di `\layoutVariable` il blocco `\layout` finale conterrebbe:

```
TextScript.padding = #1
TextScript.color = #magenta
Glissando.thickness = #1.5
NoteHead.font-size = #4
NoteHead.color = #red
```

più le sovrascritture `indent` e `StaffGrouper`.

Ma se la variabile fosse già stata definita prima del primo blocco `\layout`, la configurazione corrente conterrebbe soltanto

```
NoteHead.font-size = #4 % (scritta nella definizione della variabile)
NoteHead.color = #red % (aggiunta dopo l'uso della variabile)
```

Se ben organizzate, le variabili `\layout` possono essere un valido strumento per strutturare le formattazioni dei sorgenti, e anche per ripristinare la configurazione di `\layout` a uno stato conosciuto.

## Vedi anche

Guida alla notazione: Sezione 5.1.5 [Modifica delle impostazioni predefinite di un contesto], pagina 602.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### 4.2.2 Impostare la dimensione del rigo

La dimensione del rigo (in inglese **staff size**) è 20 punti, che corrispondono a un'altezza del rigo di 7.03mm (un punto è uguale a 100/7227 di un pollice o a 2540/7227 mm). La dimensione del rigo può essere modificata in tre modi:

1. Per impostare la dimensione del rigo globalmente per tutti i brani di un file (o di un blocco `\book`, per essere precisi), usare `set-global-staff-size`:

   ```
   #(set-global-staff-size 14)
   ```

   L'esempio precedente imposta la dimensione del rigo globale a 14pt (4.92mm) e ridimensiona proporzionalmente tutti i tipi di carattere.

2. Per impostare la dimensione del rigo di una singola partitura in un libro, usare `layout-set-staff-size` all'interno del blocco `\layout` di quel brano:

```
\score {
  ...
  \layout {
    #(layout-set-staff-size 14)
  }
}
```

3. Per impostare la dimensione del rigo di un singolo rigo di un sistema, usare il comando `\magnifyStaff`. Per esempio, le partiture di musica da camera incise in modo tradizionale spesso usavano righi per pianoforte di 7mm mentre gli altri righi erano solitamente tra 3/5 e 5/7 più grandi (tra 60% e 71%). Per ottenere la proporzione 5/7, usare:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #5/7
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Se si desidera una dimensione del tipo di carattere (`fontSize`) ben precisa, si può usare la seguente forma:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #(magstep -3)
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Per emulare l'aspetto delle partiture incise coi metodi tradizionali, è meglio evitare di ridurre lo spessore delle linee del rigo.

## Corpo automatico dei tipi di carattere a dimensioni diverse

Il font Emmentaler fornisce l'insieme di glifi musicali *Feta* in otto dimensioni diverse, ciascuna calibrata per una specifica dimensione del rigo. Più piccola è la dimensione del glifo, più "grosso" diventa, per abbinarsi alle linee del rigo più spesse. Le dimensioni dei glifi consigliate sono elencate nella seguente tabella:

| **nome del tipo di carattere** | **altezza del rigo (pt)** | **altezza del rigo (mm)** | **uso** |
|---|---|---|---|
| feta11 | 11.22 | 3.9 | partiture tascabili |
| feta13 | 12.60 | 4.4 | |
| feta14 | 14.14 | 5.0 | |
| feta16 | 15.87 | 5.6 | |
| feta18 | 17.82 | 6.3 | canzonieri |
| feta20 | 20 | 7.0 | parti standard |
| feta23 | 22.45 | 7.9 | |
| feta26 | 25.2 | 8.9 | |

### Vedi anche

Guida alla notazione: Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552, Sezione A.8 [Il font Emmentaler], pagina 679.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

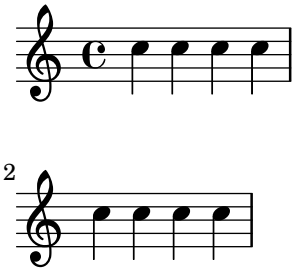
## 4.3 Interruzioni

### 4.3.1 Interruzioni di linea

Le interruzioni di linea di solito sono determinate automaticamente. Sono decise in modo che le linee non sembrino né fitte né troppo spaziate, e che le linee consecutive abbiano una densità simile.
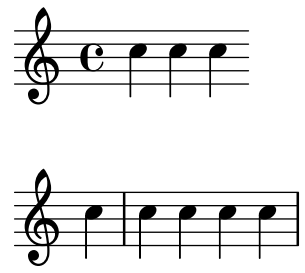
Per forzare manualmente un’interruzione di linea dopo una stanghetta, usare il comando `\break`:

```
\relative c'' {
  c4 c c c | \break
  c4 c c c |
}
```



Per impostazione predefinita, un comando `\break` inserito nel mezzo di una misura viene ignorato (e apparirà un messaggio di avvertimento durante la compilazione del file LilyPond). L’aggiunta di una stanghetta invisibile – ‘`\bar ""`’ – prima del comando `\break` forzerà l’interruzione di linea:

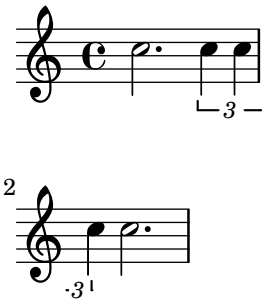
```
\relative c'' {
  c4 c c
  \bar "" \break
  c |
  c4 c c c |
}
```



Un comando `\break` successivo a una stanghetta viene ignorato se la misura precedente termina nel mezzo di una nota (per esempio, quando un gruppo irregolare inizia in una misura e termina in un’altra). In questo caso conviene togliere l’incisore `Forbid_line_break_engraver`

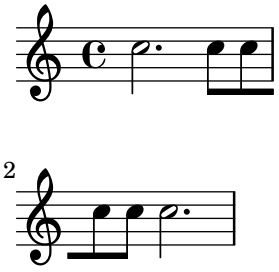
dal contesto `Voice` e usare una costruzione musicale simultanea inserendo il `\break` nel punto giusto della seconda voce:

```
\new Voice \with {
  \remove "Forbid_line_break_engraver"
} \relative {
  <<
    { c''2. \tuplet 3/2 { c4 c c } c2. | }
    { s1 | \break s1 | }
  >>
}
```



Analogalmente, le interruzioni di linea vengono ignorate quando le travature oltrepassano una stanghetta. Tale comportamento predefinito può essere modificato col comando `\override Beam.breakable = ##t`:

```
\relative c'' {
  \override Beam.breakable = ##t
  c2. c8[ c | \break
  c8 c] c2. |
}
```



Il comando `\noBreak` impedisce un'interruzione di linea sulla stanghetta in cui è inserito.

In una partitura, l'interruzione di linea automatica è vietata per la musica compresa tra i comandi `\autoLineBreaksOff` e `\autoLineBreaksOn`. Per impedire anche le interruzioni di pagina, usare i comandi `\autoBreaksOff` e `\autoBreaksOn`. Le interruzioni manuali non sono interessate da questi comandi. Nota che bloccare le interruzioni di linea automatiche potrebbe far andare la musica oltre il margine destro se questa non può essere contenuta in una linea.

Le interruzioni di linea automatiche (ma non le interruzioni di pagina) possono essere abilitate per singole stanghette usando `\once \autoLineBreaksOn` all'inizio di una stanghetta. Ciò identifica un'interruzione di linea permessa, invece che forzata.

Le impostazioni fondamentali che influenzano la spaziatura della linea sono `indent` e `line-width`, impostate nel blocco `\layout`: regolano l'indentazione della prima linea e la lunghezza delle linee.

Se `ragged-right` è impostato su vero nel blocco `\layout`, allora i sistemi terminano alla loro naturale lunghezza orizzontale, invece di essere allungati orizzontalmente per riempire l'intera linea. Ciò è utile per brevi frammenti e per verificare quanto è stretta la spaziatura naturale.

L'opzione `ragged-last` è simile a `ragged-right`, ma agisce soltanto sull'ultima linea del brano.

```
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

Per inserire interruzioni di linea a intervalli regolari usare `\break` separato da pause spaziatrici e ripetuto con `\repeat`. Per esempio, per interrompere le seguenti 28 misure (considerando un tempo di 4/4) esattamente ogni 4 misure, usare:

```
<<
  \repeat unfold 7 {
    s1 \noBreak s1 \noBreak
    s1 \noBreak s1 \break
  }
  { la vera musica... }
>>
```

## Comandi predefiniti

`\break`, `\noBreak`, `\autoBreaksOff`, `\autoBreaksOn`, `\autoLineBreaksOff`, `\autoLineBreaksOn`.

## Frammenti di codice selezionati

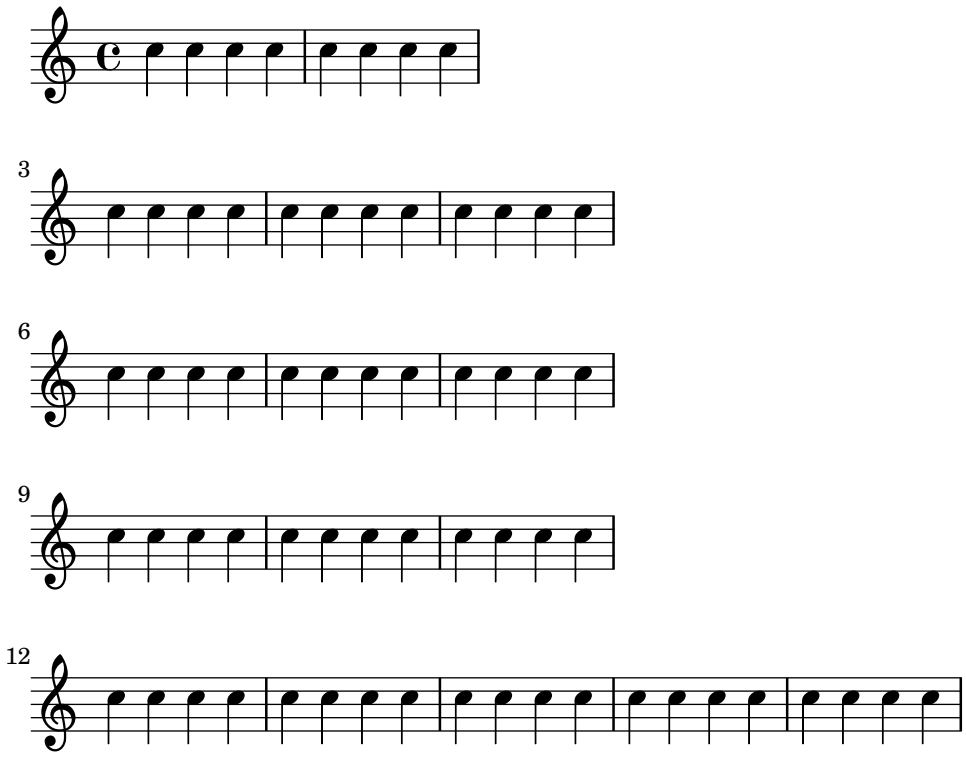
*Usare una voce apposita per le interruzioni*

Spesso è più facile gestire l'informazione sulle interruzioni di linea e di pagina tenendola separata dalla musica grazie a un'ulteriore voce che contenga solo pause spaziatrici e i comandi `\break`, `\pageBreak` e altre informazioni di formattazione.

Questo modello diventa utile specialmente quando si modifica `line-break-system-details` e altre utili ma lunghe proprietà di `NonMusicalPaperColumnGrob`.

```
music = \relative c'' { c4 c c c }

\score {
  \new Staff <<
    \new Voice {
      s1 * 2 \break
      s1 * 3 \break
      s1 * 6 \break
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 6 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}
```




## Vedi anche

Guida alla notazione: [Variabili di paper per l'interruzione di linea], pagina 547, Sezione 4.2.1 [Il blocco layout], pagina 550.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "LineBreakEvent" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

L'inserimento dei comandi `\autoLineBreaksOff` o `\autoBreaksOff` prima della musica produrrà dei messaggi di errore. Inserire sempre questi comandi dopo la musica.

### 4.3.2 Interruzioni di pagina

Questa sezione descrive i diversi metodi di interruzione di pagina e spiega come modificarli.

## Interruzione di pagina manuale

L'interruzione di pagina predefinita può essere sovrascritta con i comandi `\pageBreak` o `\noPageBreak`. Questi comandi, analoghi a `\break` e `\noBreak`, devono essere inseriti dopo una stanghetta e forzano o proibiscono l'interruzione di pagina in quel punto. Ovviamente il comando `\pageBreak` forza anche un'interruzione di linea.

I comandi `\pageBreak` e `\noPageBreak` possono essere inseriti anche nel livello superiore, tra le partiture e i testi (nel blocco `markup`) di livello superiore.

In un brano, le interruzioni di pagina automatiche sono vietate per la musica compresa tra i comandi `\autoPageBreaksOff` e `\autoPageBreaksOn`. Le interruzioni di pagina manuali non sono interessate da questi comandi.

Esistono impostazioni analoghe a `ragged-right` e `ragged-last` che hanno lo stesso effetto sulla spaziatura verticale. Se `ragged-bottom` è impostato su `#t`, i sistemi non saranno giustificati verticalmente. Quando `ragged-last-bottom` è impostato su `#t` (valore predefinito), è permesso dello spazio vuoto in fondo all'ultima pagina (o in fondo all'ultima pagina di ciascun `\bookpart`). Vedi ⟨undefined⟩ [Variabili fisse della spaziatura verticale di `\paper`], pagina ⟨undefined⟩.

Le interruzioni di pagina sono calcolate dalla funzione `page-breaking`. LilyPond fornisce vari algoritmi per calcolare le interruzioni di pagina, tra cui `ly:optimal-breaking`, `ly:page-turn-breaking` e `ly:minimal-breaking`. Quello predefinito è `ly:optimal-breaking`, ma il valore può essere modificato nel blocco `\paper`:

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

Quando un libro ha molte partiture e pagine, il problema delle interruzioni di pagina potrebbe essere difficile da risolvere e richiedere lunghi tempi di elaborazione e molta memoria. Per semplificare il processo di interruzione delle pagine, si usano i blocchi `\bookpart` per dividere il libro in varie parti: in questo modo l'interruzione di pagina si verifica separatamente in ciascuna parte. Si possono anche usare algoritmi di interruzione di pagina diversi per le diverse parti del libro.

```
\bookpart {
  \header {
    subtitle = "Prefazione"
  }
  \paper {
     %% In una parte contenente soprattutto testo,
     %% ly:minimal-breaking potrebbe essere preferibile
     page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In questa parte, contenente musica, si usa l'algoritmo di
  %% interruzione di pagina ottimale.
  \header {
    subtitle = "Primo movimento"
  }
  \score { ... }
  ...
}
```

## Comandi predefiniti

`\pageBreak`, `\noPageBreak`, `\autoPageBreaksOn`, `\autoPageBreaksOff`.

## Vedi anche

Guida alla notazione: [Variabili di paper per l'interruzione di pagina], pagina 548.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il prefisso `\once` non funziona con i comandi `\autoPageBreaksOn` e `\autoPageBreaksOff`. Se l'interruzione di pagina automatica è disabilitata e poi viene abilitata per permettere un'interruzione di pagina, deve restare attiva per alcune battute (il numero preciso di battute dipende dalla partitura) prima di essere disattivata, altrimenti la possibilità di interrompere la pagina non verrà considerata.

### Interruzione di pagina ottimale

La funzione `ly:optimal-breaking` è il metodo predefinito di LilyPond per determinare le interruzioni di pagina. Tenta di individuare un'interruzione di pagina che minimizzi la densità e l'allungamento, sia orizzontalmente che verticalmente. Diversamente da `ly:page-turn-breaking`, non prende in considerazione le voltate di pagina.

### Vedi anche

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### Interruzione di pagina minimale

La funzione `ly:minimal-breaking` fa dei calcoli minimi per valutare l'interruzione di pagina: riempie una pagina col maggior numero possibile di sistemi prima di passare a quella successiva. Dunque potrebbe essere preferibile per le partiture con molte pagine, per le quali le altre funzioni di interruzione di pagina potrebbero essere troppo lente o richiedere troppa memoria, o con molto testo. Si abilita con:

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

### Vedi anche

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### Interruzione di pagina di una pagina

La funzione `ly:one-page-breaking` è un algoritmo di interruzione di pagina per casi speciali che regola automaticamente l'altezza della pagina per farci entrare la musica, in modo che stia tutto in una sola pagina. La variabile `paper-height` del blocco paper viene ignorata, ma le altre impostazioni funzionano normalmente. In particolare, la spaziatura tra l'ultimo sistema (o il testo markup di livello superiore) e il piè di pagina può essere personalizzata grazie a `last-bottom-spacing` nel blocco paper. La larghezza della pagina non viene modificata ma può essere impostata con `paper-width` nel blocco paper.

### Problemi noti e avvertimenti

`ly:one-page-breaking` non è attualmente compatibile con `\bookpart`.

### Interruzione di pagina su una linea

La funzione `ly:one-line-breaking` è un algoritmo di interruzione di pagina per uso speciale in quanto pone ogni brano su una sola pagina e una singola linea. Non appaiono né titoli né margini, ma viene mostrato solo il brano musicale.

La larghezza della pagina è regolata in modo che il brano più lungo stia su una sola linea. In particolare, le variabili `paper-width`, `line-width` e `indent` del blocco `\paper` vengono ignorate, sebbene `left-margin` e `right-margin` siano comunque considerate. L'altezza della pagina resta invariata.

### Interruzione di pagina su una linea con altezza automatica

La funzione `ly:one-line-auto-height-breaking` funziona proprio come `ly:one-line-breaking` con la differenza che l'altezza della pagina viene modificata automaticamente per adattarsi all'altezza della musica. Più precisamente, la variabile `paper-height` del blocco `\paper` viene impostata in modo che abbracci l'altezza della partitura più alta e i margini superiore e inferiore (`top-margin` e `bottom-margin`).

Fare attenzione al fatto che l'impostazione `top-system-spacing` avrà effetto sulla posizione verticale della musica. Impostarla su `##f` in un blocco `\paper` per posizionare la musica tra i margini superiore e inferiore.

## Voltata di pagina ottimale

È spesso necessario trovare una configurazione delle interruzioni di pagina in cui ci sia una pausa al termine di ogni due pagine. In questo modo il musicista può voltare la pagina senza perdere le note. La funzione `ly:page-turn-breaking` tenta di trovare un'interruzione di pagina che minimizzi densità e allungamento, ma con l'ulteriore restrizione che le voltate di pagina sono permesse solo in punti specifici.

Ci sono due passi da seguire per usare questa funzione. Prima occorre abilitarla nel blocco `\paper`, come è spiegato in Sezione 4.3.2 [Interruzioni di pagina], pagina 557. Poi bisogna indicare alla funzione dove sono permesse le interruzioni di pagina.

Ci sono due modi per fare il secondo passo. Si può specificare manualmente ogni potenziale voltata di pagina, inserendo `\allowPageTurn` nei punti adatti del file di input.

Oppure, se ciò è troppo noioso, si può aggiungere l'incisore `Page_turn_engraver` a un contesto Staff o Voice. L'incisore `Page_turn_engraver` analizzerà il contesto in cerca di sezioni senza note (non cerca pause, bensì l'assenza di note, in modo che la polifonia su un singolo rigo con pause in una delle parti non confonda `Page_turn_engraver`). Quando trova una sezione senza note abbastanza lunga, `Page_turn_engraver` inserirà il comando `\allowPageTurn` nella stanghetta finale di quella sezione, a meno che non ci sia una stanghetta 'speciale' (come una doppia stanghetta), nel qual caso il comando `\allowPageTurn` sarà inserito nella stanghetta finale "speciale" della sezione.

L'incisore `Page_turn_engraver` legge la proprietà di contesto `minimumPageTurnLength` per determinare quanto deve essere lunga una sezione senza note prima che una voltata di pagina sia considerata. Il valore predefinito di `minimumPageTurnLength` è `(ly:make-moment 1/1)`. Per disabilitare le voltate di pagina, impostarla su un valore "molto grande".

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % voltata di pagina permessa qui
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
  R1 | % voltata di pagina non permessa qui
  a4 b r2 |
  R1*2 | % voltata di pagina permessa qui
  a1
}
```

Quando si usano ripetizioni con finali alternativi, l'incisore `Page_turn_engraver` permetterà una voltata di pagina durante la ripetizione soltanto se c'è abbastanza tempo all'inizio e alla fine della ripetizione per voltare indietro la pagina. Se la ripetizione è troppo breve, si può usare `Page_turn_engraver` per *disabilitare* le voltate impostando un valore appropriato per la proprietà di contesto `minimumRepeatLengthForPageTurn`. In questo caso `Page_turn_engraver` consentirà le voltate soltanto nelle ripetizioni la cui durata sia maggiore del valore specificato.

I comandi per le voltate di pagina (`\pageTurn`, `\noPageTurn` e `\allowPageTurn`), possono essere usati anche nel livello superiore, nei blocchi markup di livello superiore e tra una partitura e l'altra.

## Comandi predefiniti

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

### Vedi anche

Guida alla notazione: [Variabili di paper per l'interruzione di linea], pagina 547.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### Problemi noti e avvertimenti

Usare soltanto un incisore `Page_turn_engraver` per partitura. Se ce n'è più d'uno, interferiranno uno con l'altro.

### Vedi anche

Guida alla notazione: Sezione 4.4 [Spaziatura verticale], pagina 561.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.4 Spaziatura verticale

La spaziatura verticale è regolata da tre elementi: la quantità di spazio disponibile (ovvero il formato e i margini), la quantità di spazio tra i sistemi e la quantità di spazio tra i righi di un sistema.

### 4.4.1 Spaziatura verticale flessibile all'interno dei sistemi

Tre meccanismi distinti regolano la spaziatura verticale flessibile all'interno dei sistemi, uno per ognuna delle seguenti categorie:

- *righi non raggruppati*,
- *righi raggruppati* (righi con un gruppo come `ChoirStaff`, etc.), e
- *linee che non sono righi* (come `Lyrics`, `ChordNames`, etc.).

L'altezza di ogni sistema è determinata in due fasi. Prima vengono spaziati tutti i righi in base alla quantità di spazio disponibile. Poi le linee che non sono righi sono distribuite tra i righi.

Nota che i meccanismi di spaziatura trattati in questa sezione regolano soltanto la spaziatura verticale dei righi e delle linee (che non sono righi) all'interno di singoli sistemi. La spaziatura verticale tra sistemi, partiture, testi e margini separati è regolata dalle variabili `\paper` trattate in ⟨undefined⟩ [Variabili `\paper` della spaziatura verticale flessibile], pagina ⟨undefined⟩.

#### Proprietà della spaziatura dentro un sistema

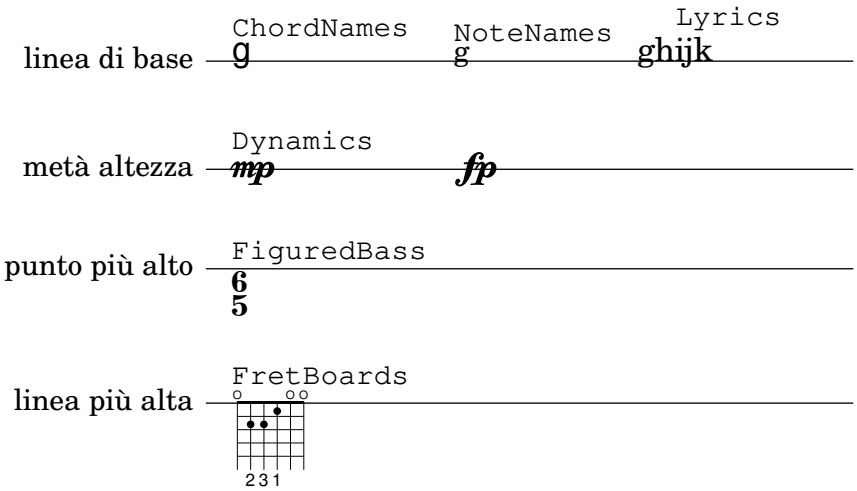
I meccanismi di spaziatura verticale dentro un sistema sono regolati da due gruppi di proprietà dei grob. Il primo gruppo è associato al grob `VerticalAxisGroup`, creato da tutti i righi e tutte le linee che non sono righi. Il secondo gruppo è associato al grob `StaffGrouper`, che può essere creato da gruppi di righi, ma solo se richiamato esplicitamente. Queste proprietà sono descritte una per una alla fine di questa sezione.

Il nome di queste proprietà (con l'eccezione di `staff-affinity`) hanno il formato *`elemento1`*`-`*`elemento2`*`-spacing`, dove *`elemento1`* e *`elemento2`* sono gli elementi di cui determinare la distanza. Nota che *`elemento2`* non è necessariamente sotto *`elemento1`*; per esempio, `nonstaff-relatedstaff-spacing` prenderà le misure verso l'alto a partire dalla linea che non è un rigo (nonstaff) se `staff-affinity` è impostato su `UP`.

Ogni distanza è calcolata tra i *punti di riferimento* dei due elementi. Il punto di riferimento di un rigo è il centro verticale del suo `StaffSymbol` (ovvero la linea centrale se `line-count` (il numero di linee) è dispari; lo spazio centrale se `line-count` è pari). I punti di riferimento per ciascuna linea che non è un rigo sono elencati nella seguente tabella:

| **Linea non-rigo** | **Punto di riferimento** |
|---|---|
| `ChordNames` | linea di base |
| `NoteNames` | linea di base |
| `Lyrics` | linea di base |
| `Dynamics` | metà altezza di 'm' |
| `FiguredBass` | punto più alto |
| `FretBoards` | linea più alta |

Nell'immagine seguente, le linee orizzontali indicano le posizioni di questi punti di riferimento:




Tutte le proprietà di spaziatura verticale del grob (eccetto `staff-affinity`) usano la stessa struttura della lista associativa usata dalle variabili di spaziatura di `\paper` trattate in ⟨undefined⟩ [Variabili `\paper` della spaziatura verticale flessibile], pagina ⟨undefined⟩. Metodi specifici per modificare queste liste sono spiegati in Sezione 5.3.7 [Modifica delle liste associative], pagina 627. Le proprietà dei grob devono essere modificate con un comando `\override` dentro un blocco `\score` o `\layout` e non in un blocco `\paper`.

L'esempio seguente illustra i due modi con cui si possono modificare queste liste associative (alist). La prima dichiarazione trasforma un elemento-valore singolarmente, mentre la seconda ridefinisce completamente la proprietà:

```
\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing =
    #'((basic-distance . 10)
       (minimum-distance . 9)
       (padding . 1)
       (stretchability . 10))
} { ... }
```

Per cambiare le impostazioni di spaziatura globalmente, inserirle in un blocco `\layout`:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
```

```
    }
  }
```

Le impostazioni predefinite delle proprietà di spaziatura verticale dei grob sono elencate in Sezione "VerticalAxisGroup" in *Guida al Funzionamento Interno* e Sezione "StaffGrouper" in *Guida al Funzionamento Interno*. Le modifiche predefinite con `\override` per tipologie specifiche di linee che non sono righi sono elencate nelle descrizioni del relativo contesto in Sezione "Contexts" in *Guida al Funzionamento Interno*.

### Proprietà del grob `VerticalAxisGroup`

Le proprietà di `VerticalAxisGroup` sono solitamente modificate con un `\override` nel livello `Staff` (o equivalente).

`staff-staff-spacing`
: Usata per determinare la distanza tra il rigo corrente e il rigo inferiore nello stesso sistema, anche se tra i due si trovano una o più linee che non sono righi (come `Lyrics`). Non è applicata all'ultimo rigo di un sistema.

  Inizialmente, la proprietà `staff-staff-spacing` di un `VerticalAxisGroup` è una funzione Scheme che applica le proprietà di `StaffGrouper` se il rigo fa parte di un gruppo, o la proprietà `default-staff-staff-spacing` del rigo altrimenti. Questo permette ai righi di essere spaziati diversamente quando sono raggruppati. Per ottenere una spaziatura uniforme indipendentemente dal raggruppamento, questa funzione può essere sostituita da un alist di spaziatura flessibile, usando la forma di override che ridefinisce completamente la variabile, come mostrato prima.

`default-staff-staff-spacing`
: Un alist di spaziatura flessibile che definisce la proprietà `staff-staff-spacing` usata per i righi isolati, a meno che `staff-staff-spacing` non sia stata impostata esplicitamente con un `\override`.

`staff-affinity`
: La direzione del rigo da usare per spaziare la linea che non è un rigo. Le opzioni sono `UP` (su), `DOWN` (giù) e `CENTER` (centro). Se impostata su `CENTER`, la linea fuori dal rigo si troverà in un punto equidistante tra i due righi più vicini su qualunque lato, a meno che delle collisioni o altre costrizioni di spazio non lo impediscano. Linee (che non sono righi) adiacenti dovrebbero avere un valore di `staff-affinity` che non cresce: per esempio, una linea che non è un rigo impostata su `UP` non deve seguire immediatamente una linea impostata su `DOWN`. Linee che non sono righi in cima a un sistema devono usare `DOWN`; quelle in fondo `UP`. Impostando `staff-affinity` per un rigo, questo sarà trattato come una linea che non è un rigo. Impostando `staff-affinity` su `#f`, una linea che non è un rigo sarà trattata come un rigo. Impostando `staff-affinity` su `UP`, `CENTER` o `DOWN`, un rigo verrà spaziato come se fosse una linea che non è un rigo.

`nonstaff-relatedstaff-spacing`
: La distanza fra la linea (che non è un rigo) corrente e il rigo più vicino nella direzione di `staff-affinity`, se non ci sono linee che non sono righi tra le due e `staff-affinity` è impostato su `UP` o `DOWN`. Se `staff-affinity` è impostato su `CENTER`, viene usato `nonstaff-relatedstaff-spacing` per i righi più vicini su *entrambi* i lati, anche se appaiono altre linee tra quella corrente e uno qualsiasi dei righi. Ciò significa che il posizionamento di una linea dipende sia dai righi che dalle linee circostanti. Impostando la proprietà `stretchability` di uno di questi tipi di spaziatura su un piccolo valore, quella spaziatura sarà dominante. Impostando `stretchability` su un grande valore, quella spaziatura avrà poco effetto.

`nonstaff-nonstaff-spacing`
: La distanza fra la linea (che non è un rigo) corrente e quella successiva nella direzione di `staff-affinity`, se entrambe sono sullo stesso lato del rigo in questione e se `staff-affinity` è impostata su `UP` o `DOWN`.

`nonstaff-unrelatedstaff-spacing`
: La distanza fra la linea (che non è un rigo) corrente e il rigo nella direzione opposta rispetto a `staff-affinity`, se non ci sono altre linee tra i due e se `staff-affinity` è impostato su `UP` o `DOWN`. Ciò può servire, per esempio, a imporre un padding minimo tra una linea `Lyrics` e il rigo al quale non appartiene.

### Proprietà del grob `StaffGrouper`

Le proprietà di `StaffGrouper` sono solitamente modificate con un `\override` nel livello `StaffGroup` (o livello equivalente).

`staff-staff-spacing`
: La distanza tra righi consecutivi del gruppo di righi corrente. La proprietà `staff-staff-spacing` del grob `VerticalAxisGroup` di un singolo rigo può essere sovrascritta con varie impostazioni di spaziatura per quel rigo.

`staffgroup-staff-spacing`
: La distanza tra l'ultimo rigo del gruppo di righi corrente e il rigo immediatamente successivo nello stesso sistema, anche se tra i due righi ci sono una o più linee che non sono righi (come `Lyrics`). Non è applicata al rigo inferiore di un sistema. La proprietà `staff-staff-spacing` del grob `VerticalAxisGroup` di un singolo rigo può essere sovrascritta con varie impostazioni di spaziatura per quel rigo.

### Vedi anche

Guida alla notazione: ⟨undefined⟩ [Variabili `\paper` della spaziatura verticale flessibile], pagina ⟨undefined⟩, Sezione 5.3.7 [Modifica delle liste associative], pagina 627.

File installati: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Guida al funzionamento interno: Sezione "Contexts" in *Guida al Funzionamento Interno*, Sezione "VerticalAxisGroup" in *Guida al Funzionamento Interno*, Sezione "StaffGrouper" in *Guida al Funzionamento Interno*.

### Spaziatura dei righi non raggruppati

I *righi* (come `Staff`, `DrumStaff`, `TabStaff`, etc.) sono contesti che possono contenere uno o più contesti voce, ma non possono contenere altri righi.

Le seguenti proprietà influenzano la spaziatura di righi *non raggruppati*:

- Proprietà di `VerticalAxisGroup`:
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

Queste proprietà del grob sono state descritte una a una in precedenza; vedi ⟨undefined⟩ [Within-system spacing properties], pagina ⟨undefined⟩.

Altre proprietà entrano in gioco per i righi che sono parte di un gruppo; vedi ⟨undefined⟩ [Spacing of grouped staves], pagina ⟨undefined⟩.

L'esempio seguente mostra come la proprietà `default-staff-staff-spacing` possa influenzare la spaziatura di righi non raggruppati. Le stesse modifiche applicate a `staff-staff-spacing` avrebbero lo stesso effetto, ma verrebbero applicate anche nel caso in cui i righi siano combinati in uno o più gruppi.

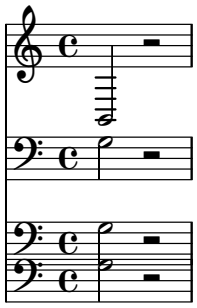
```
\layout {
```

```
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 8)
         (minimum-distance . 7)
         (padding . 1))
  }
}

<<
  % Questa nota molto bassa ha bisogno di più spazio di quanto 'basic-distance
  % possa fornirne, dunque la distanza tra questo rigo e quello successivo
  % è determinato da 'padding.
  \new Staff { b,2 r | }

  % Qui 'basic-distance fornisce abbastanza spazio, e non c'è bisogno
  % di comprimere lo spazio (verso 'minimum-distance) per far spazio
  % per qualcos'altro sulla pagina, dunque la distanza tra questo
  % rigo e quello successivo è determinato da 'basic-distance.
  \new Staff { \clef bass g2 r | }

  % Impostando 'padding su un valore negativo, è possibile far sì che
  % i righi entrino in collisione.  Il più basso valore accettabile per
  % 'basic-distance è 0.
  \new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 3.5)
         (padding . -10))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



## Vedi anche

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

## Spaziatura dei righi raggruppati

Nelle partiture orchestrali e in alte grosse partiture, di norma i righi vengono raggruppati. Lo spazio tra i gruppi è più ampio dello spazio tra i righi dello stesso gruppo.

I *gruppi di righi* (come `StaffGroup`, `ChoirStaff`, etc.) sono contesti che possono contenere uno o più righi simultaneamente.

Le seguenti proprietà influenzano la spaziatura dei righi nei gruppi:

- Proprietà di `VerticalAxisGroup`:
  - `staff-staff-spacing`
- Proprietà di `StaffGrouper`:
  - `staff-staff-spacing`
  - `staffgroup-staff-spacing`

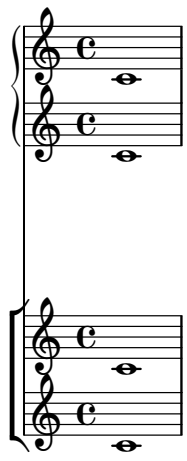
Queste proprietà dei grob sono descritte una a una in una sezione precedente; vedi ⟨undefined⟩ [Within-system spacing properties], pagina ⟨undefined⟩.

L'esempio seguente mostra come le proprietà del grob `StaffGrouper` possano influenzare la spaziatura dei righi raggruppati:

```
\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = #0
    \override StaffGrouper.staff-staff-spacing.basic-distance = #1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper.staffgroup-staff-spacing.basic-distance = #20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>
>>
```



### Vedi anche

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

## Spaziatura delle linee che non sono righi

Le *linee che non sono righi* (come `Lyrics`, `ChordNames`, etc.) sono contesti i cui oggetti della formattazione sono disposti come se fossero su dei righi (ovvero su linee orizzontali all'interno dei sistemi). Precisamente, le linee che non sono righi sono contesti non-rigo che contengono l'incisore Sezione “Axis_group_engraver” in *Guida al Funzionamento Interno*.

Le seguenti proprietà influenzano la spaziatura delle linee che non sono righi:

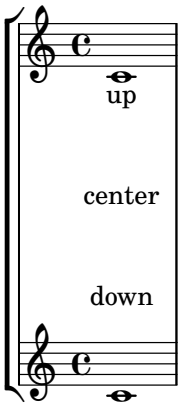
- Proprietà di `VerticalAxisGroup`:
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

Queste proprietà del grob sono descritte una a una in una sezione precedente, vedi ⟨undefined⟩ [Within-system spacing properties], pagina ⟨undefined⟩.

L'esempio seguente mostra come la proprietà `nonstaff-nonstaff-spacing` influenza la spaziatura di linee che non sono un rigo consecutive. Impostando l'elemento `stretchability` su un valore molto alto, il testo vocale riesce a allungarsi molto più del solito:

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup.nonstaff-nonstaff-spacing.stretchability = #1000
  }
}

\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup.staff-staff-spacing = #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>
```




## Vedi anche

File installati: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Contexts” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

### 4.4.2 Posizionamento esplicito di righi e sistemi

Un modo per comprendere i meccanismi di spaziatura verticale appena spiegati è di considerarli come un insieme di impostazioni che regolano la quantità di *padding* verticale tra righi e tra sistemi.

È possibile gestire la spaziatura verticale in un modo diverso usando `NonMusicalPaperColumn.line-break-system-details`. Mentre i meccanismi di spaziatura verticale flessibile specificano il padding verticale, `NonMusicalPaperColumn.line-break-system-details` indica precisamente le posizioni verticali esatte sulla pagina.

`NonMusicalPaperColumn.line-break-system-details` accetta una lista associativa di quattro diverse impostazioni:

- `X-offset`
- `Y-offset`
- `extra-offset`
- `alignment-distances`

Le modifiche del grob con `\override`, incluse quelle per `NonMusicalPaperColumn` come nell’esempio successivo, possono trovarsi in uno di questi tre diversi punti del file di input:

- direttamente in mezzo alle note
- in un blocco `\context`
- nel blocco `\with`

Quando si modifica `NonMusicalPaperColumn`, si usa il solito comando `\override` nei blocchi `\context` e nel blocco `\with`. Invece quando si modifica `NonMusicalPaperColumn` in mezzo alle note, si usa il comando speciale `\overrideProperty`. Ecco alcuni esempi di modifiche di `NonMusicalPaperColumn` col comando speciale `\overrideProperty`:

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 40))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
```
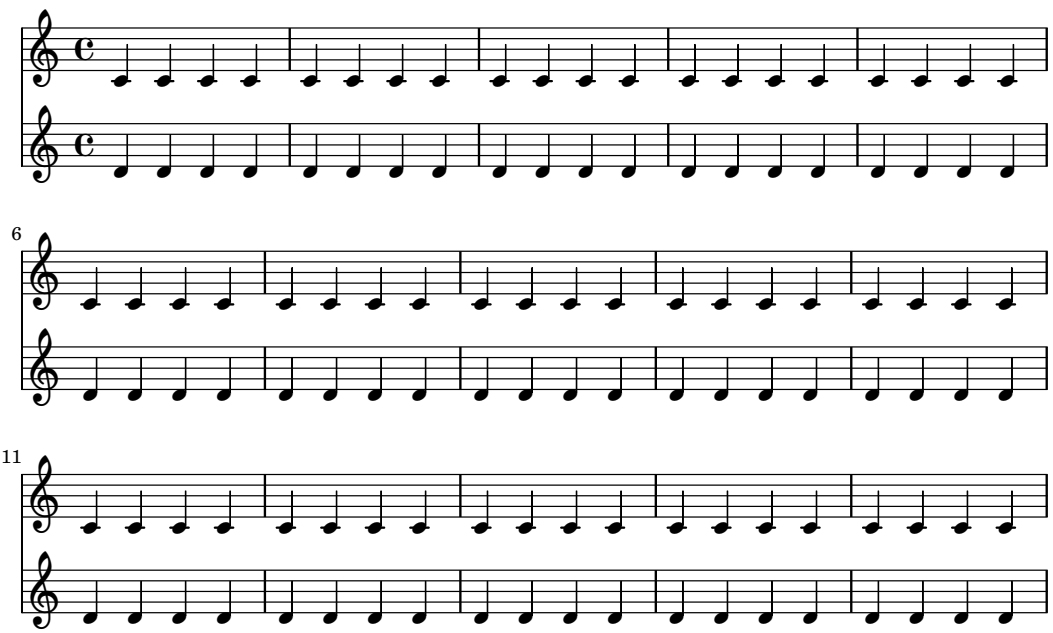
```
  #'((X-offset . 20)
     (Y-offset . 40))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((alignment-distances . (15)))

\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
     (Y-offset . 40)
     (alignment-distances . (15)))
```

Per comprendere come funziona ognuna di queste impostazioni, iniziamo vedendo un esempio che non contiene alcuna modifica.
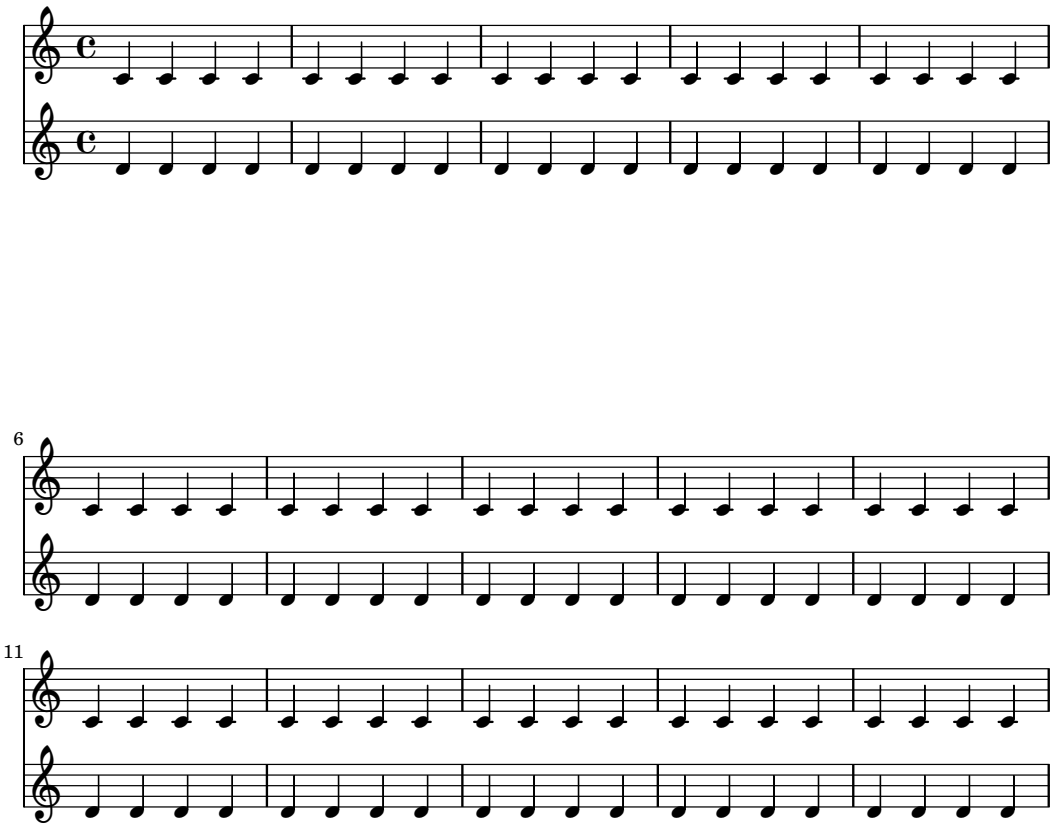
```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```



Questa partitura isola l'informazione sulle interruzioni di linea e di pagina in una voce apposita. Questa tecnica di creare una voce per le interruzioni permette di tenere la formattazione separata dalla musica via via che il nostro esempio diventa più complicato. Vedi anche Sezione 4.3 [Interruzioni], pagina 554.

Usando comandi `\break` espliciti, la musica viene divisa proporzionalmente in cinque misure per linea. La spaziatura verticale è quella predefinita di LilyPond ma il punto di inizio verticale di ogni sistema è impostato esplicitamente con la coppia `Y-offset` dell'attributo `line-break-system-details` del grob `NonMusicalPaperColumn`:

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 0))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 40))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
}
```
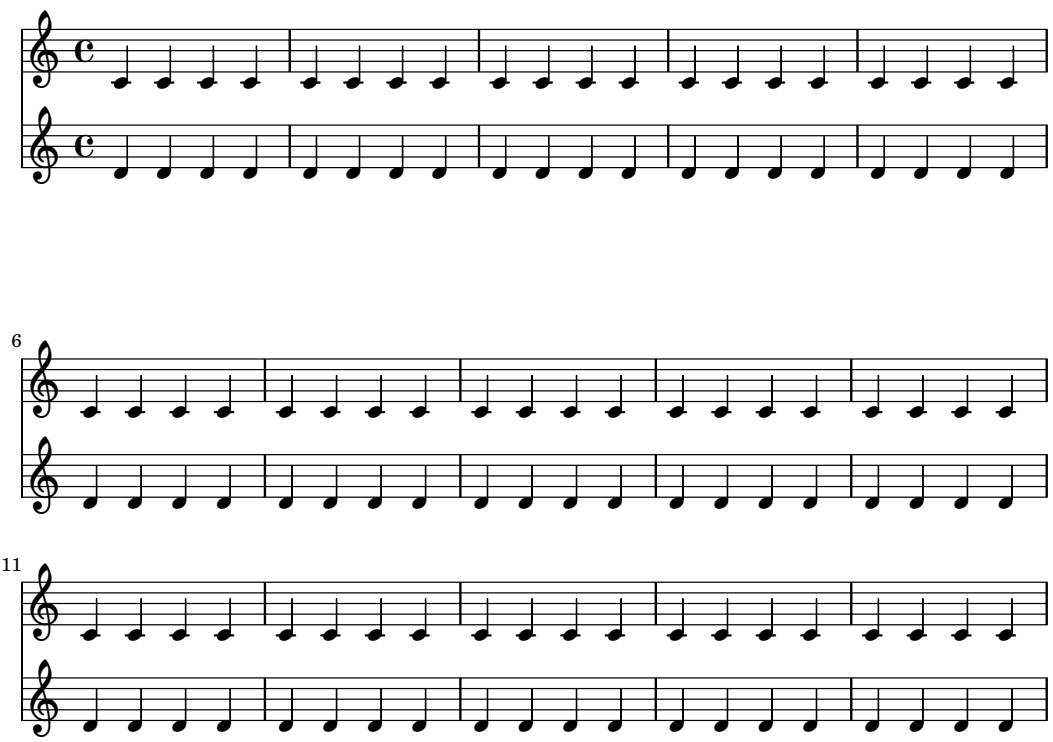


Nota che `line-break-system-details` accetta una lista associativa di molti valori, ma ne abbiamo impostato solo uno in questo esempio. Nota anche che la proprietà `Y-offset` qui determina la posizione verticale esatta sulla pagina in cui ogni nuovo sistema verrà visualizzato.

Oltre al posizionamento assoluto che si ottiene con `Y-offset` e `X-offset`, è possibile anche il posizionamento relativo tramite la proprietà `extra-offset` di `line-break-system-details`. Il posizionamento è relativo alla formattazione predefinita o al posizionamento assoluto determinato dall'uso di `X-offset` e `Y-offset`. La proprietà `extra-offset` accetta una `coppia` di numeri che determinano lo spostamento lungo gli assi X e Y.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((extra-offset . (0 . 10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((extra-offset . (0 . 10)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
    >>
  }
```
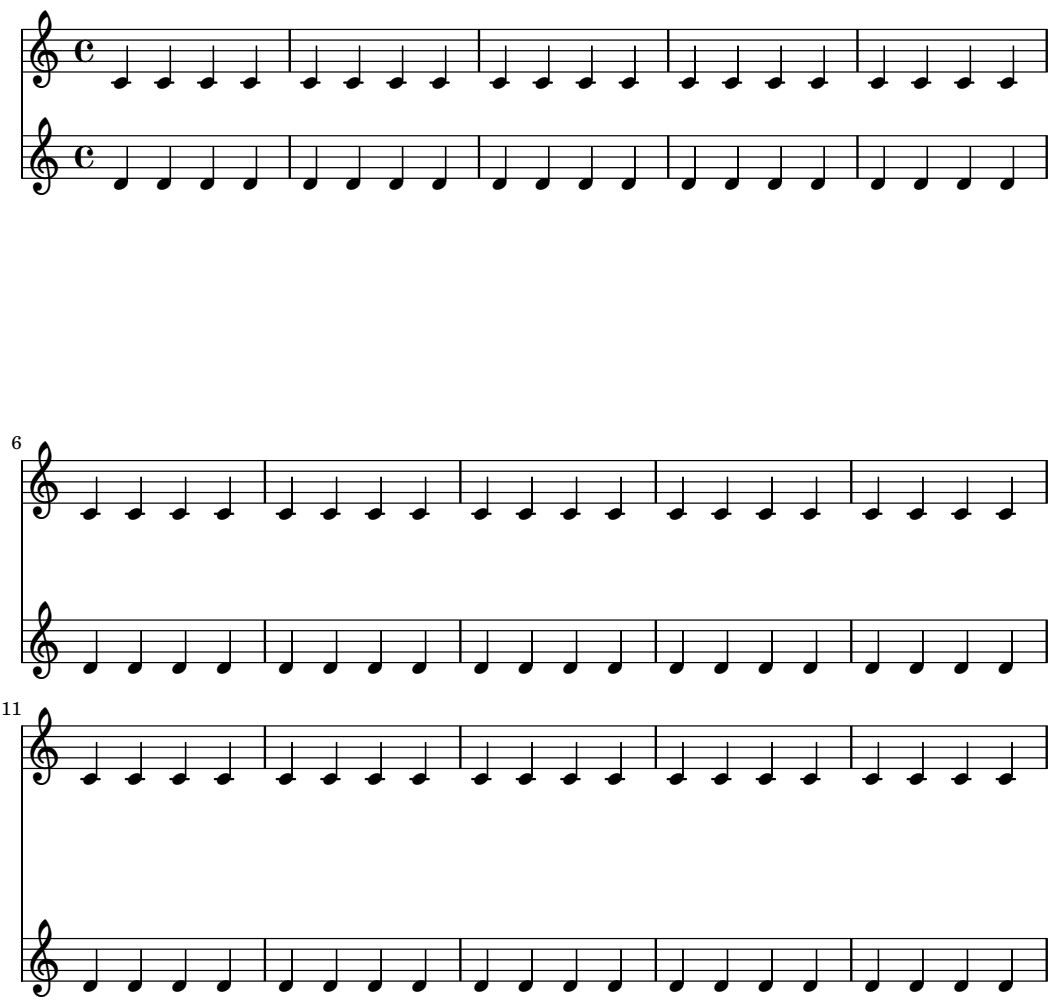
```
}
```



Ora che abbiamo impostato esplicitamente il punto di inizio verticale di ogni sistema, possiamo impostare manualmente anche le distanze verticali tra i righi. Per farlo usiamo la sottoproprietà `alignment-distances` di `line-break-system-details`.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 20)
               (alignment-distances . (10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60)
               (alignment-distances . (15)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 85)
             (alignment-distances . (20)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \repeat unfold 15 { d'4 d' d' d' }
      }
```
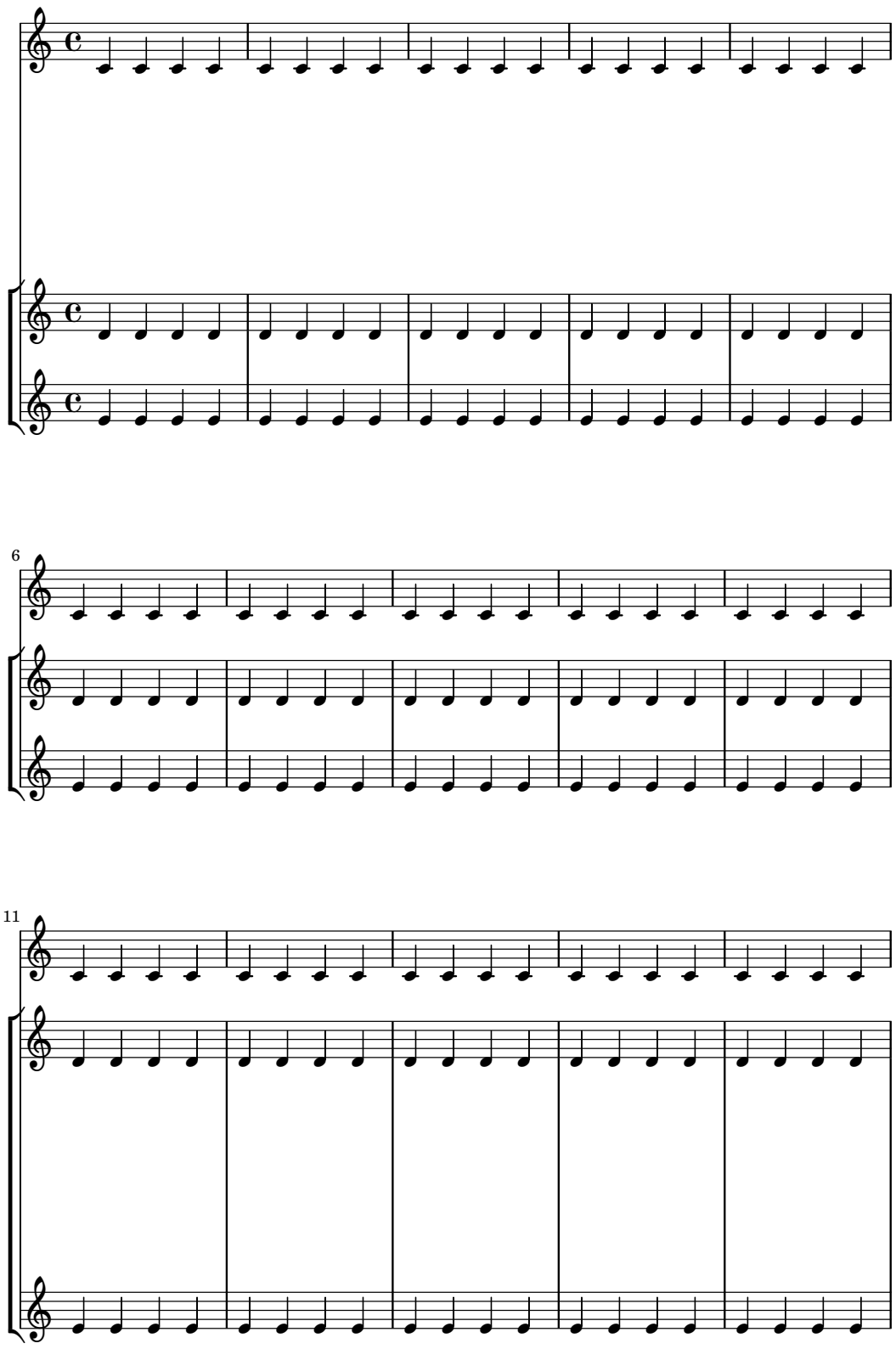
```
    >>
  }
}
```




Nota che qui assegnamo due valori diversi all'attributo `line-break-system-details` del grob `NonMusicalPaperColumn`. Sebbene l'attributo alist `line-break-system-details` accetti molti altri parametri di spaziatura (inclusa, per esempio, una coppia corrispondente di `X-offset`), è sufficiente impostare soltanto le coppie `Y-offset` e `alignment-distances` per regolare il punto di inizio verticale di ogni sistema e ogni rigo. Infine nota che `alignment-distances` specifica il posizionamento verticale dei righi ma non dei gruppi di righi.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 0)
               (alignment-distances . (30 10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
```

```
            #'((Y-offset . 60)
               (alignment-distances . (10 10)))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 100)
               (alignment-distances . (10 30)))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
      \new StaffGroup <<
        \new Staff { \repeat unfold 15 { d'4 d' d' d' } }
        \new Staff { \repeat unfold 15 { e'4 e' e' e' } }
      >>
    >>
  }
}
```




Alcuni punti da considerare:

- Quando si usa `alignment-distances`, il testo vocale e altre linee che non sono righi non contano come rigo.
- Le unità dei numeri assegnati a `X-offset`, `Y-offset`, `extra-offset` e `alignment-distances` sono interpretati come multipli della distanza tra linee del rigo adiacenti. Valori positivi spostano in su righi e testo, valori negativi li spostano in giù.
- Dato che le impostazioni di `NonMusicalPaperColumn.line-break-system-details` illus-

trate qui permettono il posizionamento di righi e sistemi ovunque sulla pagina, è possibile violare i confini del foglio o dei margini o perfino sovrapporre righi e sistemi uno sopra l'altro. Ciò può essere evitato assegnando valori ragionevoli a queste diverse impostazioni.

### Vedi anche

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.4.3 Elusione delle collisioni verticali

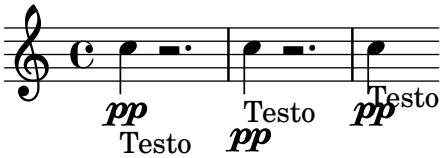
Intuitivamente, ci sono alcuni oggetti della notazione musicale che appartengono al rigo e altri che devono essere disposti fuori dal rigo. Gli oggetti esterni al rigo comprendono i numeri di chiamata, il testo e le dinamiche (d'ora in avanti tutti questi elementi saranno chiamati oggetti esterni al rigo). La regola di LilyPond per il posizionamento verticale degli oggetti esterni al rigo è di disporli il più vicino possibile al rigo ma non così vicino da farli collidere con un altro oggetto.

LilyPond usa la proprietà `outside-staff-priority` per determinare se un grob è un oggetto fuori dal rigo: se `outside-staff-priority` è un numero, il grob è un oggetto esterno al rigo. `outside-staff-priority` indica a LilyPond anche in quale ordine disporre gli oggetti.

LilyPond posiziona prima tutti gli oggetti che non sono esterni al rigo. Quindi ordina gli oggetti esterni al rigo in base al loro valore di `outside-staff-priority` (in ordine crescente). Uno per volta, LilyPond prende gli oggetti esterni al rigo e li dispone in modo che non entrino in collisione con alcun oggetto che sia già stato disposto. Ovvero, se due grob esterni al rigo si contendono lo stesso spazio, quello col valore di `outside-staff-priority` più basso sarà posto più vicino al rigo.

Un elenco delle proprietà esterne al rigo si trova in Sezione "La proprietà outside-staff-priority" in *Manuale di Apprendimento*.

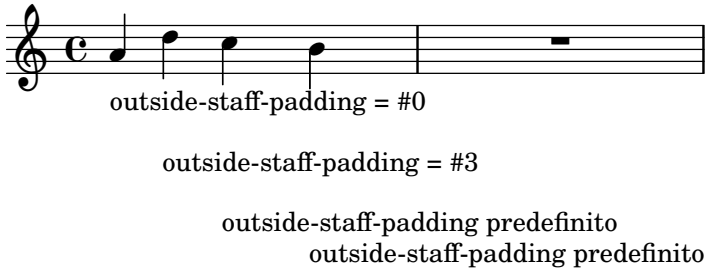
```
\relative c'' {
  c4_"Testo"\pp
  r2.
  \once \override TextScript.outside-staff-priority = #1
  c4_"Testo"\pp % stavolta il testo sarà più vicino al rigo
  r2.
  % impostando outside-staff-priority su un non-numero,
  % disabilitiamo l'elusione automatica delle collisioni
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Testo"\pp % qui entrano in collisione
}
```




Il padding verticale intorno agli oggetti esterni al rigo può essere regolato con `outside-staff-padding`.

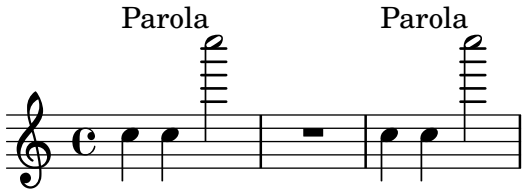
```
\relative {
  \once \override TextScript.outside-staff-padding = #0
  a'4-"outside-staff-padding = #0"
  \once \override TextScript.outside-staff-padding = #3
  d-"outside-staff-padding = #3"
```

```
  c-"outside-staff-padding predefinito"
  b-"outside-staff-padding predefinito"
  R1
}
```




Per impostazione predefinita, gli oggetti esterni al rigo sono disposti in modo da evitare la collisione orizzontale con grob posizionati precedentemente. Ciò può portare a situazioni in cui gli oggetti sono posizionati uno vicino all'altro orizzontalmente. Come è dimostrato nell'esempio successivo, impostando `outside-staff-horizontal-padding` si aumenta la spaziatura orizzontale richiesta e in questo caso si sposta in su il testo per impedire che si avvicini troppo ai tagli addizionali.

```
\relative {
  c''4^"Parola" c c''2
  R1
  \once \override TextScript.outside-staff-horizontal-padding = #1
  c,,4^"Parola" c c''2
}
```




### Vedi anche

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.5 Spaziatura orizzontale
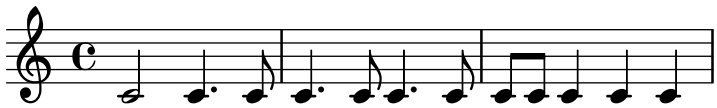
### 4.5.1 Panoramica sulla spaziatura orizzontale

Il motore della spaziatura traduce le differenze delle durate delle note in distanze allungabili ('springs') di diversa lunghezza. Durate più lunghe occupano più spazio, quelle più brevi ne occupano meno. Le durate più brevi occupano una quantità fissa di spazio (regolata da `shortest-duration-space` nell'oggetto Sezione "SpacingSpanner" in *Guida al Funzionamento Interno*). Più lunga è la durata, più spazio occupa: raddoppiando una durata si aggiunge spazio alla nota di una quantità pari al valore di `spacing-increment`.

Per esempio, il brano seguente contiene molte minime, semiminime e crome; la croma (1/8) è seguita da 1 Larghezza della Testa di Nota (LTN). La semiminima (1/4) è seguita da 2 LTN, la minima (1/2) da 3 LTN, etc.

```
\relative c' {
  c2 c4. c8
```

```
  c4. c8 c4. c8
  c8 c c4 c c
}
```



Solitamente, `spacing-increment` è impostato su 1.2 di spazio rigo, che equivale all'incirca alla larghezza della testa di nota, e `shortest-duration-space` è impostato su 2.0, che significa che la nota più breve occupa 2.4 di spazio rigo (2 volte `spacing-increment`) di spazio orizzontale. Questo spazio è calcolato dal margine sinistro del simbolo, dunque le note più brevi sono generalmente seguite da un LTN di spazio.
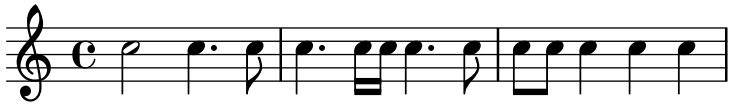
Se si seguisse esattamente la procedura descritta, aggiungendo una sola biscroma (1/32) a un brano che usa solo crome e semicrome, la spaziatura orizzontale dell'intero brano sarebbe troppo larga. Infatti la nota più breve non è più una semicroma ma una biscroma, aggiungendo quindi 1 LTN a ogni nota. Per impedire ciò, la durata più breve per la spaziatura non deriva dalla nota più breve, bensì dalla nota che ricorre più frequentemente nel brano.

La durata più breve più comune viene individuata nel modo seguente. In ogni misura viene determinata la durata più breve e quella più frequente viene scelta come base per la spaziatura, con la condizione che tale durata debba essere sempre uguale o inferiore a una nota di un ottavo.

Tali durate possono anche essere personalizzate. Impostando `common-shortest-duration` in Sezione "SpacingSpanner" in *Guida al Funzionamento Interno*, si imposta la durata di base per la spaziatura. La durata massima per essa (solitamente un ottavo), si imposta con `base-shortest-duration`.

Note ancora più brevi della nota più breve più comune sono seguite da uno spazio proporzionale alla loro durata rispetto a essa. Dunque se aggiungessimo solo alcuni sedicesimi all'esempio precedente, sarebbero seguiti dalla metà di LTN:
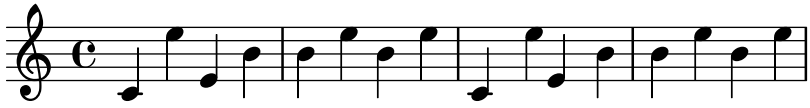
```
\relative { c''2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
```



Come è spiegato nel saggio *Essay on automated music engraving*, le direzioni del gambo influenzano la spaziatura (vedi Sezione "Optical spacing" in *Saggio*) e possono essere aggiustate usando la proprietà `stem-spacing-correction` dell'oggetto Sezione "NoteSpacing" in *Guida al Funzionamento Interno* (che sono generati per ogni contesto Sezione "Voice" in *Guida al Funzionamento Interno*.

L'oggetto `StaffSpacing` (generato nel contesto Sezione "Staff" in *Guida al Funzionamento Interno*) contiene la stessa proprietà per regolare la spaziatura tra gambo e stanghetta.

L'esempio seguente mostra tutto questo: una volta con le impostazioni predefinite e una volta con un aggiustamento esagerato:



LilyPond supporta la notazione proporzionale, vedi ⟨undefined⟩ [Proportional notation], pagina ⟨undefined⟩.

### Vedi anche

Essay on automated music engraving: Sezione “Spaziatura ottica” in *Saggio*.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*, Sezione “NoteSpacing” in *Guida al Funzionamento Interno*, Sezione “StaffSpacing” in *Guida al Funzionamento Interno*, Sezione “NonMusicalPaperColumn” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

Non esiste un modo semplice per modificare manualmente la spaziatura. Per aggirare il problema si può inserire dell’ulteriore spazio in una partitura, regolando il valore di padding di quanto è necessario:

```
\override Score.NonMusicalPaperColumn.padding = #10
```
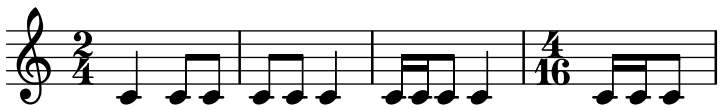
Non esiste alcun trucco per diminuire la quantità di spazio.

## 4.5.2 Nuova spaziatura nel corso di un brano

Nuove sezioni con parametri di spaziatura diversi possono essere iniziati col comando `\newSpacingSection`. Ciò può essere utile per sezioni con nozioni diverse di note ‘lunghe’ e note ‘brevi’. Il comando `\newSpacingSection` crea un nuovo oggetto `SpacingSpanner` in quel momento musicale.
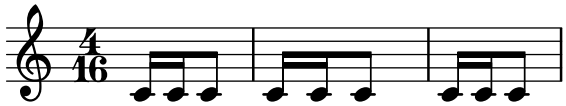
Nell’esempio seguente, il cambio di indicazione di tempo introduce una nuova sezione, e i sedicesimi hanno automaticamente una spaziatura un po’ più larga.

```
\relative c' {
  \time 2/4
  c4 c8 c
  c8 c c4 c16[ c c8] c4
  \newSpacingSection
  \time 4/16
  c16[ c c8]
}
```



Se gli aggiustamenti automatici della spaziatura non producono la spaziatura richiesta, si possono applicare degli `\override` manuali alle sue proprietà. Questi devono essere applicati nello stesso momento musicale del comando `\newSpacingSection` stesso e avranno effetto sulla spaziatura di tutta la musica seguente finché le proprietà non vengono cambiate in una nuova sezione. Per esempio:

```
\relative c' {
  \time 4/16
  c16[ c c8]
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #2
  c16[ c c8]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
  c16[ c c8]
}
```



### Vedi anche
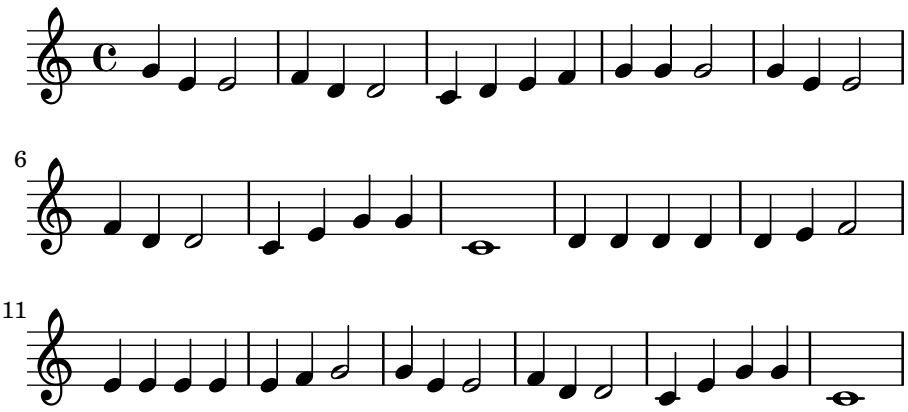
Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*.
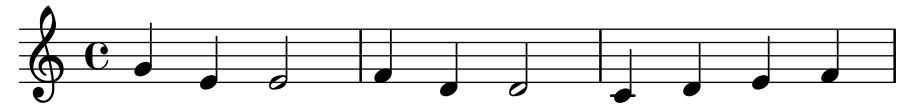
## 4.5.3 Modifica della spaziatura orizzontale
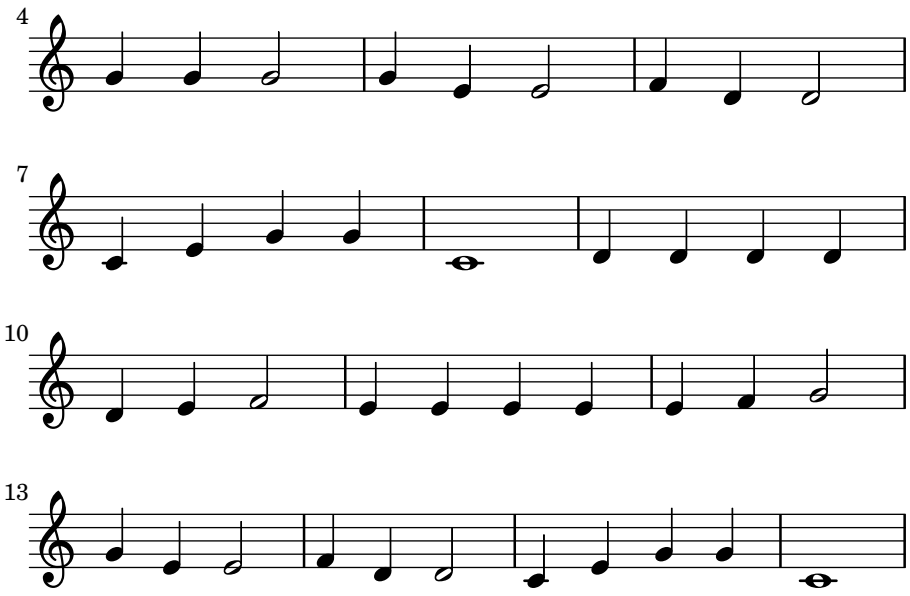
La spaziatura orizzontale può essere modificata tramite la proprietà `base-shortest-duration`. Nel prossimo esempio confrontiamo la stessa musica, prima senza cambiare la proprietà e poi cambiandola. Valori più grandi di `ly:make-moment` produrranno musica più densa. Nota che `ly:make-moment` constituisce una durata, dunque `1 4` è una durata più lunga di `1 16`.

```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
}
```




```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
    }
  }
}
```

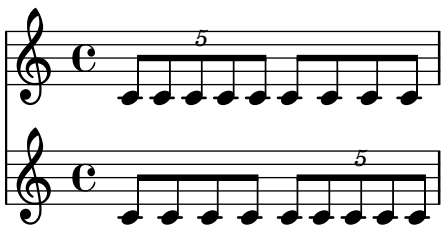




## Allungamento uniforme dei gruppi irregolari

L'impostazione predefinita prevede che la spaziatura nei gruppi irregolari dipenda da vari fattori diversi dalla durata (come alterazioni, cambi di chiave, etc). Per ignorare tali simboli e forzare la spaziatura perché sia uniforme, usare `Score.SpacingSpanner.uniform-stretching`. Questa proprietà può essere modificata soltanto all'inizio di una partitura:

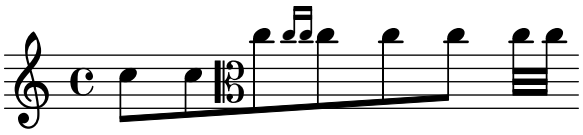
```
\score {
  <<
    \new Staff \relative c' {
      \tuplet 5/4 { c8 c c c c } c8 c c c
    }
    \new Staff \relative c' {
      c8 c c c \tuplet 5/4 { c8 c c c c }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



## Spaziatura rigorosa della nota

Se si imposta `strict-note-spacing`, la spaziatura tra le note non tiene conto di chiavi, stanghette e abbellimenti:

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative {
  c''8[ c \clef alto c \grace { c16 c } c8 c c]  c32[ c] }
```



## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.5.4 Larghezza della linea

Le impostazioni fondamentali che influenzano la spaziatura sono `indent` e `line-width`, impostate nel blocco `\layout`. Regolano l’indentazione della prima linea musicale e la lunghezza delle linee.

Se `ragged-right` è impostato su vero nel blocco `\layout`, i sistemi terminano alla loro naturale lunghezza orizzontale, invece di essere espansi orizzontalmente per riempire tutta la linea. Ciò è utile in caso di brevi frammenti e per verificare quanto è compatta la spaziatura naturale. L’impostazione predefinita è solitamente falso, ma se la partitura ha un solo sistema il valore predefinito è vero.

L’opzione `ragged-last` è simile a `ragged-right`, ma ha effetto soltanto sull’ultima linea del brano. Nessune restrizioni vengono poste su quella linea. Il risultato è simile alla formattazione dei paragrafi di testo. In un paragrafo l’ultima linea occupa la sua naturale lunghezza orizzontale.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.5.5 Notazione proporzionale

LilyPond supporta la notazione proporzionale, un tipo di spaziatura orizzontale in cui ogni nota occupa una quantità di spazio orizzontale esattamente equivalente alla sua durata musicale. Questo tipo di spaziatura proporzionale può essere paragonata alla spaziatura orizzontale su carta quadrettata. Alcune partiture della fine del ventesimo secolo e dell’inizio del ventunesimo usano la notazione proporzionale per chiarire relazioni ritmiche complesse o per agevolare il posizionamento della linea del tempo o di altri elementi grafici direttamente nella partitura.

LilyPond supporta cinque diverse impostazioni per la notazione proporzionale, che possono essere usate insieme o da sole:
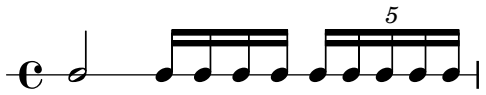
- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove "Separating_line_group_engraver"`
- `\override PaperColumn.used = ##t`

Nell’esempio seguente analizziamo queste cinque diverse impostazioni di notazione proporzionale e valutiamo come esse interagiscono tra loro.

Iniziamo con l’esempio seguente di una misura, che usa la spaziatura classica con la giustificazione del rigo disattivata.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
```

```
    }
  >>
}
```

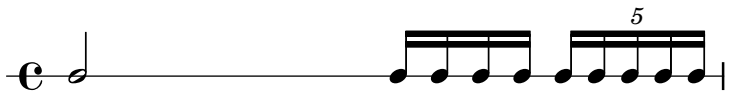



La minima all'inizio della misura occupa uno spazio molto inferiore alla metà dello spazio orizzontale della misura. Ugualmente, i sedicesimi e le quintine di sedicesimi alla fine della battuta insieme occupano molto più spazio della metà dello spazio orizzontale della misura.

Nell'incisione tipografica classica, questa spaziatura è solitamente proprio ciò che si desidera, perché è possibile prendere in prestito dello spazio orizzontale dalla minima e economizzare lo spazio orizzontale complessivo della misura.

D'altra parte, se vogliamo inserire una linea del tempo con tacche o altri elementi grafici sopra o sotto la partitura, abbiamo bisogno della notazione proporzionale. Per attivarla si usa l'impostazione proportionalNotationDuration.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
 \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```




La minima all'inizio della misura e le note più veloci nella seconda metà della misura ora occupano la stessa quantità di spazio orizzontale. Potremmo inserire una linea del tempo con tacche o un'immagine grafica sopra o sotto questo esempio.

L'impostazione `proportionalNotationDuration` è un'impostazione di contesto che si trova in `Score`. Ricordiamo che le impostazioni di contesto possono apparire in tre luoghi del file di input: in un blocco `\with`, in un blocco `\context` o direttamente in mezzo alle note preceduta dal comando `\set`. Come per tutte le impostazioni di contesto, l'utente può scegliere in quale di questi tre luoghi impostare `proportionalNotationDuration`.

L'impostazione `proportionalNotationDuration` prende un solo argomento, che è la durata di riferimento in base alla quale tutta la musica verrà spaziata. La funzione Scheme di LilyPond `make-moment` prende due argomenti: un numeratore e un denominatore che insieme esprimono una qualche frazione di una nota intera. La funzione di chiamata `(ly:make-moment 1/20)` produce quindi una durata di riferimento di un ventesimo di nota. Sono ammessi anche valori come `(ly:make-moment 1/16)`, `(ly:make-moment 1/8)` e `(ly:make-moment 3/97)`.

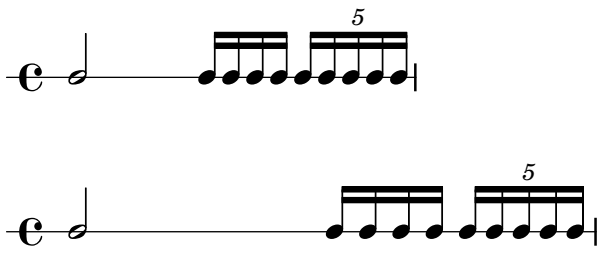
Come scegliamo la durata di riferimento corretta da passare a `proportionalNotationDuration`? Solitamente attraverso un processo di prova e errore, iniziando con una durata vicina alla più veloce (o piccola) durata del brano. Durate di

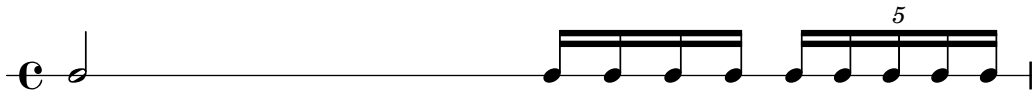
riferimento più piccole determinano una spaziatura della musica più larga; quelle più grandi causano una spaziatura più stretta.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/8)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/16)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/32)
    }
  }
}
```



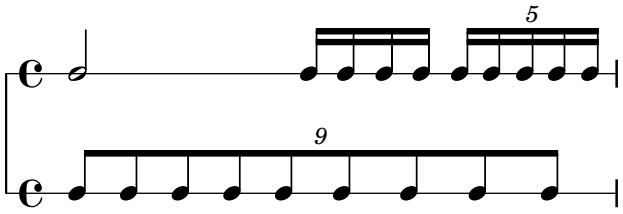



Nota che una durata di riferimento troppo grande – come la nota di un ottavo, sopra – determina una spaziatura della musica troppo stretta e può causare collisioni tra le teste di nota. Fare attenzione anche al fatto che la notazione proporzionale in generale occupa più spazio orizzontale della spaziatura classica. Insomma, la spaziatura proporzionale fornisce chiarezza ritmica al costo dello spazio orizzontale.

Ora vediamo come spaziare in modo ottimale i gruppi irregolari sovrapposti.

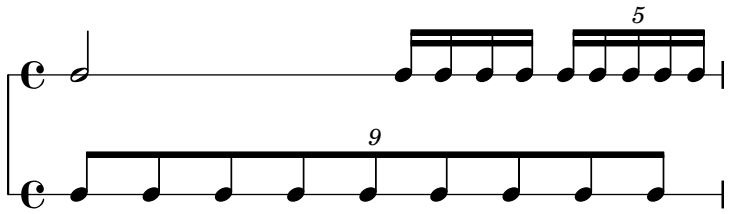
Iniziamo esaminando cosa succede al nostro esempio di partenza, con la spaziatura classica, quando aggiungiamo un secondo rigo con un diverso tipo di gruppo irregolare.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
}
```



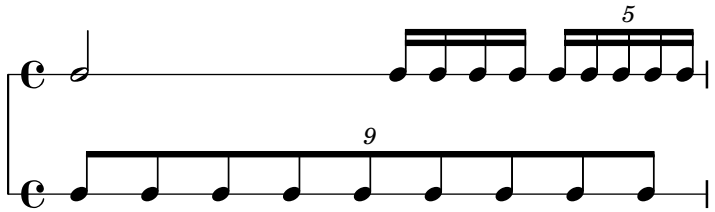

La spaziatura è pessima perché le note del rigo inferiore spaziate a distanze uguali non si allungano in modo uniforme. Le incisioni classiche contengono pochissime terzine complesse e quindi le regole di incisione classica possono generare questo tipo di risultato. Impostando `proportionalNotationDuration` ciò viene corretto.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```




Ma se osserviamo con attenzione possiamo vedere che le note della seconda metà della nonina hanno una spaziatura leggermente più larga delle note della prima parte della nonina. Per assicurare un allungamento uniforme, attiviamo `uniform-stretching`, una proprietà di `SpacingSpanner`.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```




Il nostro esempio di due righi ora ha una spaziatura esatta, le relazioni ritmiche sono visivamente chiare e possiamo includere una linea del tempo con tacche o altro elemento grafico, se lo vogliamo.

Nota che la notazione proporzionale di LilyPond si aspetta che tutte le partiture proporzionali impostino l'attributo 'uniform-stretching di SpacingSpanner su `##t`. Impostare proportionalNotationDuration senza impostare anche l'attributo 'uniform-stretching di SpacingSpanner su `##t` farà sì, per esempio, che le pause invisibili occupino una quantità scorretta di spazio orizzontale.

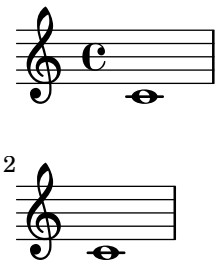
SpacingSpanner è un grob astratto che si trova nel contesto Score. Come per le impostazioni di proportionalNotationDuration, le modifiche `\override` a SpacingSpanner si possono trovare in uno dei tre diversi punti del file di input – nel blocco \with del brano, nel blocco \context del brano o direttamente in mezzo alle note.

Per impostazione predefinita, esiste un solo `SpacingSpanner` per `Score`. Ciò significa che `uniform-stretching` è attivato o disattivato per l'intera partitura. Possiamo tuttavia modificare tale comportamento e attivare diverse funzionalità di spaziatura in punti diversi del brano. Per farlo si usa il comando `\newSpacingSection`. Maggiori informazioni in ⟨undefined⟩ [New spacing section], pagina ⟨undefined⟩.

Ora esaminiamo gli effetti dell'incisore `Separating_line_group_engraver` e vediamo perché le partiture proporzionali solitamente tolgano questo incisore. L'esempio seguente mostra che c'è un piccolo spazio "introduttivo" proprio prima della prima nota di ogni sistema.

```
\paper {
  indent = #0
}

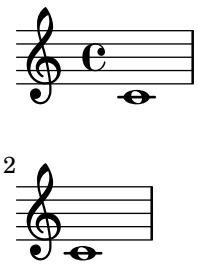
\new Staff {
  c'1
  \break
  c'1
}
```



L'ampiezza di questo spazio introduttivo è la stessa sia dopo un'indicazione di tempo che dopo un'armatura di chiave o una chiave. È l'incisore `Separating_line_group_engraver` a causare questo spazio. Togliendolo lo spazio diventa zero.

```
\paper {
  indent = #0
}

\new Staff \with {
  \remove "Separating_line_group_engraver"
} {
  c'1
  \break
  c'1
}
```



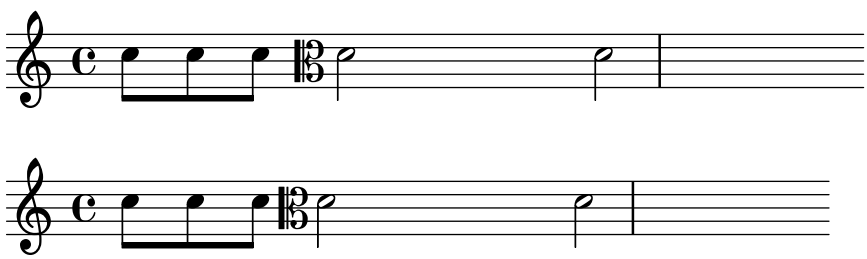
Nella notazione proporzionale gli elementi non musicali come le indicazioni di tempo, le armature di chiave, le chiavi e le alterazioni sono problematiche. Nessuna di queste infatti ha una durata ritmica, ma tutte occupano spazio orizzontale. Questi problemi sono affrontati diversamente dalle varie partiture proporzionali.

È possibile evitare i problemi di spaziatura dovuti alle armature di chiave semplicemente omettendole. Questa è un'opzione valida dato che la maggior parte delle partiture proporzionali sono di musica contemporanea. Lo stesso potrebbe valere per le indicazioni di tempo, specialmente per quelle partiture che includono una linea del tempo o altri elementi grafici. Ma queste partiture sono un'eccezione e la maggior parte delle partiture proporzionali hanno almeno qualche indicazione di tempo. Le chiavi e le alterazioni sono ancora più fondamentali.

Dunque quali strategie adottare per spaziare elementi non musicali nel contesto di musica proporzionale? Una valida opzione è la proprietà `strict-note-spacing` di `SpacingSpanner`. Confrontiamo i seguenti due righi:

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  c''8 8 8 \clef alto d'2 2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 8 8 \clef alto d'2 2
}
```



Entrambi sono proporzionali, ma la spaziatura del primo è troppo larga a causa del cambio di chiave. La spaziatura del secondo resta invece stretta, perché è attivato strict-note-spacing. Attivando strict-note-spacing, la larghezza di indicazioni di tempo, armature di chiave, chiavi e alterazioni non ha alcun ruolo nell'algoritmo di spaziatura.

Oltre alle impostazioni che abbiamo visto, ce ne sono altre che appaiono frequentemente nelle partiture proporzionali, tra cui:

- `\override SpacingSpanner.strict-grace-spacing = ##t`
- `\set tupletFullLength = ##t`
- `\override Beam.breakable = ##t`
- `\override Glissando.breakable = ##t`
- `\override TextSpanner.breakable = ##t`
- `\remove "Forbid_line_break_engraver"` nel contesto `Voice`

Queste impostazioni spaziano in modo conciso gli abbellimenti, estendono le parentesi dei gruppi irregolari per contrassegnare i punti di inizio e di fine del ritmo, e permettono agli elementi che si estendono orizzontalmente di andare oltre i sistemi e le pagine. Consultare le sezioni del manuale per queste impostazioni.

### Vedi anche

Guida alla notazione: ⟨undefined⟩ [New spacing section], pagina ⟨undefined⟩.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

## 4.6 Riduzione del numero di pagine di una partitura

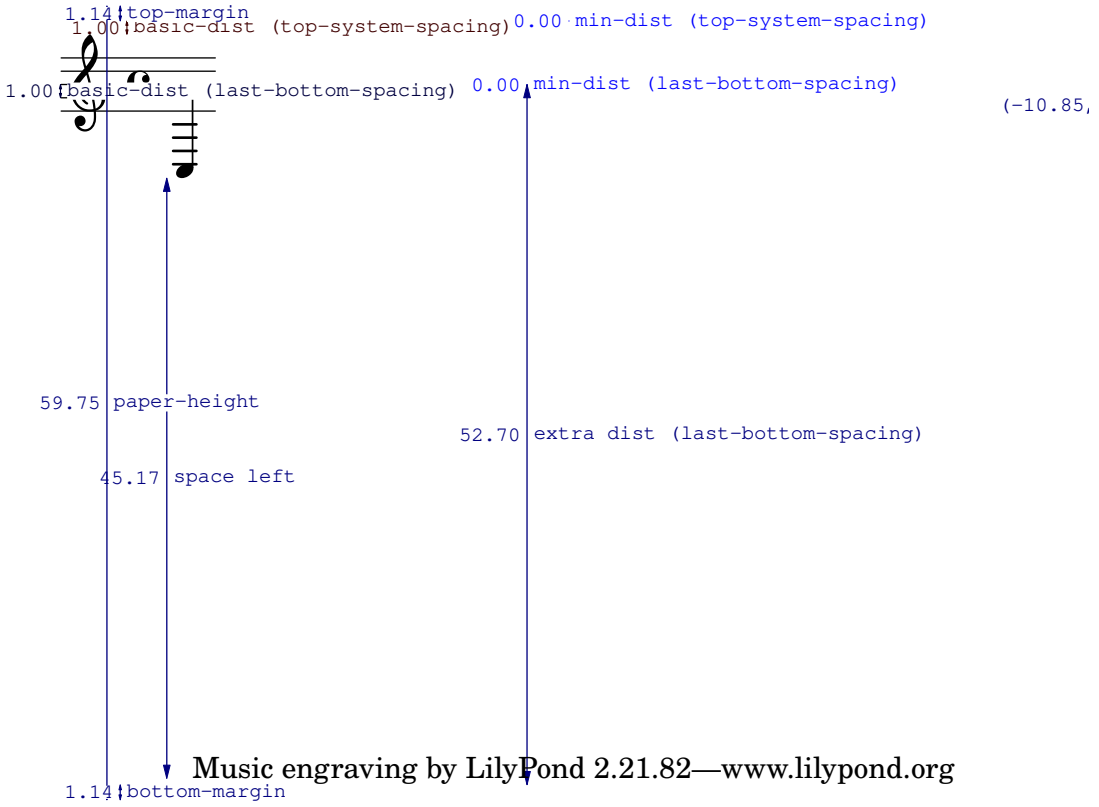
Talvolta può capitare di avere uno o due righi su una seconda (o terza, o quarta. . . ) pagina. Ciò è fastidioso, specialmente se c'è molto spazio nelle pagine precedenti.

Quando si studiano i problemi di formattazione, uno strumento irrinunciabile è `annotate-spacing`, un comando che mostra i valori delle diverse variabili di spaziatura. Maggiori dettagli nella prossima sezione, Sezione 4.6.1 [Visualizzare la spaziatura], pagina 589.

### 4.6.1 Visualizzare la spaziatura

Per visualizzare graficamente le dimensioni delle varibili della formattazione verticale che possono essere modificate per formattare la pagina, impostare `annotate-spacing` nel blocco `\paper`:

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```




Tutte le dimensioni della formattazione sono visualizzate in spazi rigo, indipendentemente dalle unità di misura specificate nei blocchi `\paper` o `\layout`. Nell'esempio qui sopra, l'altezza del foglio (`paper-height`) ha un valore di 59.75 spazi rigo (`staff-space`), e la dimensione del rigo (`staff-size`) è pari a 20 punti (il valore predefinito). Nota che:

1 punto = (25.4/72.27) mm

1 staff-space = (`staff-size`)/4 pts
= (`staff-size`)/4 *
(25.4/72.27) mm

In questo caso, uno `staff-space` è uguale all'incirca a 1.757mm. Dunque i 59.75 `staff-space` di `paper-height` corrispondono a 105 millimetri, pari all'altezza del formato `a6` in orientamento orizzontale. Le coppie ($a$,$b$) sono intervalli, dove $a$ è l'estremo inferiore e $b$ l'estremo superiore dell'intervallo.

## Vedi anche

Guida alla notazione: Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

### 4.6.2 Modificare la spaziatura

L'output di `annotate-spacing` svela le dimensioni verticali molto dettagliatamente. Maggiori informazioni su come modificare i margini e altre variabili di formattazione si trovano in Sezione 4.1 [Formattazione della pagina], pagina 539.

Oltre ai margini, ci sono altre opzioni utili per salvare spazio:

- Forzare i sistemi perché si avvicinino il più possibile (per far entrare più sistemi possibile in una pagina) mentre sono spaziati in modo da non lasciare spazio bianco in fondo alla pagina.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Forzare il numero dei sistemi. Ciò può essere utile in due modi. Il semplice impostare un valore, persino lo stesso valore del numero di sistemi disposti senza modificare la variabile, può far sì che più sistemi riescano a entrare in ogni pagina, perché viene saltato il passaggio di valutazione, dando un valore più adatto per ogni pagina. Inoltre, forzare davvero una riduzione nel numero di sistemi può far risparmiare un'ulteriore pagina. Per esempio, se la formattazione predefinita ha 11 sistemi, la seguente impostazione forzerà la formattazione con 10 sistemi.
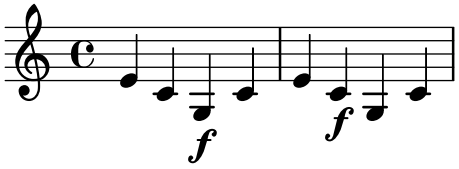
```
\paper {
  system-count = #10
}
```

- Forzare il numero delle pagine. Per esempio, la seguente impostazione forzerà la formattazione in due pagine.

```
\paper {
  page-count = #2
}
```

- Evitare (o ridurre) gli oggetti che aumentano la dimensione verticale di un sistema. Per esempio, le parentesi delle volte per i finali alternativi richiedono ulteriore spazio. Se questi finali si estendono per due sistemi, occupano più spazio che se fossero sullo stesso sistema. Altro esempio: le dinamiche che "spuntano fuori" da un sistema possono essere avvicinate al rigo:
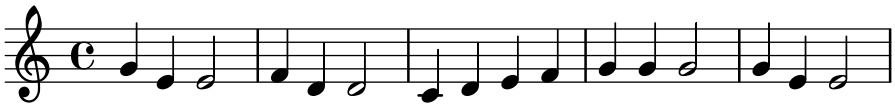
```
\relative e' {
  e4 c g\f c
  e4 c g-\tweak X-offset #-2.7 \f c
}
```




- Modificare la spaziatura orizzontale tramite `SpacingSpanner`. Maggiori informazioni in Sezione 4.5.3 [Modifica della spaziatura orizzontale], pagina 580. L'esempio seguente mostra la spaziatura predefinita:
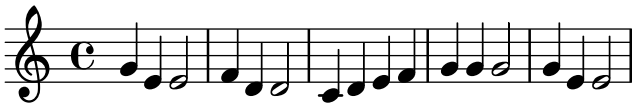
```
\score {
  \relative {
    g'4 e e2 |
```

```
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```



L'esempio successivo modifica `common-shortest-duration` da un valore di `1/4` a uno di `1/2`. La nota di un quarto è la durata più comune e più breve in questo esempio, dunque rendendola più lunga si verifica un effetto "compressione":

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration =
        #(ly:make-moment 1/2)
    }
  }
}
```



La proprietà `common-shortest-duration` non può essere modificata in modo dinamico, quindi deve essere sempre posta in un blocco `\context` così che sia applicata all'intera partitura.

## Vedi anche

Guida alla notazione: Sezione 4.1 [Formattazione della pagina], pagina 539, Sezione 4.5.3 [Modifica della spaziatura orizzontale], pagina 580.

Frammenti: Sezione "Spacing" in *Frammenti di codice*.

# 5 Modifica delle impostazioni predefinite

LilyPond è progettato per fornire la migliore qualità grafica mantenendo le impostazioni predefinite. Tuttavia può capitare che sia necessario modificare la sua formattazione predefinita. La formattazione è regolata da un gran numero di "pulsanti e interruttori" chiamati 'proprietà'. Prima di proseguire si consiglia di leggere una guida introduttiva su come accedere a queste proprietà e modificarle: vedi Sezione "Modifica dell'output" in *Manuale di Apprendimento*, nel Manuale di apprendimento. Questo capitolo tratta lo stesso argomento, ma in uno stile più adatto a un manuale.

La descrizione completa delle proprietà che si possono ritoccare si trova in un documento separato: Sezione "la Guida al funzionamento interno" in *Guida al Funzionamento Interno*. Questo manuale elenca tutte le variabili, le funzioni e le opzioni disponibili in LilyPond. Viene fornito come un documento HTML, disponibile on-line (`http://lilypond.org/doc/stable/Documentation/internals/`), ed è anche incluso nel pacchetto della documentazione di LilyPond.

Internamente, LilyPond usa il linguaggio Scheme (un dialetto di LISP) per fornire l'infrastruttura. Modificare le decisioni di formattazione prevede in effetti l'accesso alle funzioni interne del programma, cosa che richiede l'input Scheme. In un file `.ly` gli elementi Scheme sono introdotti col segno cancelletto `#`.[1]

## 5.1 Contesti di interpretazione

Questa sezione spiega cosa sono i contesti e come modificarli.

### Vedi anche

Manuale di apprendimento: Sezione "Contesti e incisori" in *Manuale di Apprendimento*.

File installati: `ly/engraver-init.ly`, `ly/performer-init.ly`.

Frammenti: Sezione "Contexts and engravers" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "Contexts" in *Guida al Funzionamento Interno*, Sezione "Engravers and Performers" in *Guida al Funzionamento Interno*.

### 5.1.1 Tutto sui contesti

I contesti sono organizzati in modo gerarchico:

### Definizioni di output - gerarchia dei contesti

Questa sezione spiega la rilevanza delle definizioni di output quando si lavora coi contesti. Esempi di vere definizioni di output sono illustrati dopo (vedi [Modifica di tutti i contesti dello stesso tipo], pagina 603).

Sebbene la musica scritta in un file possa riferirsi a un certo tipo o nome di un contesto, i contesti vengono creati soltanto quando la musica viene interpretata. LilyPond interpreta la musica in base a una "definizione di output" e lo fa per le varie definizioni di output, producendo un output diverso per ciascuna. La definizione di output che crea l'output grafico si specifica con `\layout`.

Una definizione di output molto più semplice, usata per produrre l'output Midi, si specifica con `\midi`. Molte altre definizioni di output sono usate da LilyPond internamente, come quando si usa la combinazione automatica delle parti ([Combinazione automatica delle parti], pagina 185) o si creano citazioni musicali ([Citare altre voci], pagina 213).

[1] Sezione "Scheme tutorial" in *Estendere*, contiene una breve guida sull'inserimento di numeri, liste, stringhe e simboli in Scheme.

Le definizioni di output definiscono la relazione tra i contesti e le loro rispettive impostazioni predefinite. Sebbene la maggior parte dei cambiamenti venga fatta solitamente in un blocco `\layout`, le impostazioni relative al Midi avranno effetto solo se inserite in un blocco `\midi`.

Alcune impostazioni interessano vari tipi di output: per esempio, se `autoBeaming` è disattivato in qualche contesto, le travature contano come melismi al fine di abbinare musica e testo vocale, come descritto in [Durate automatiche delle sillabe], pagina 271. Questo abbinamento viene fatto sia per l'output grafico che per il Midi. Se le modifiche fatte a `autoBeaming` in una definizione di contesto di un blocco `\layout` non sono ripetute nel corrispondente blocco `\midi`, il testo e la musica andranno fuori sincrono nel Midi.

### Vedi anche

File installati: `ly/engraver-init.ly`. `ly/performer-init.ly`.

### Score - il padre di tutti i contesti

Questo è il contesto del livello superiore. Nessun altro contesto può contenere un contesto Score. Per impostazione predefinita, il contesto Score gestisce l'amministrazione delle indicazioni di tempo e garantisce che elementi come le chiavi, le indicazioni di tempo e le armature di chiave siano allineate da rigo a rigo.

Un contesto Score viene istanziato implicitamente quando viene elaborato un blocco `\score {...}`.

### Contesti del livello superiore - contenitori di righi

*StaffGroup*

Raggruppa i righi e aggiunge una parentesi quadra sul lato sinistro, che raggruppa insieme i righi. Le stanghette dei righi in esso contenuti sono connessi verticalmente. `StaffGroup` è semplicemente un insieme di righi, con una parentesi quadra all'inizio e le stanghette che li attraversano.

*ChoirStaff*

Identico a `StaffGroup` eccetto che le stanghette dei righi in esso contenuti non sono connesse verticalmente.

*GrandStaff*

Un gruppo di righi, con una parentesi graffa sul lato sinistro, che raggruppa insieme i righi. Le stanghette dei righi in esso contenuti sono connessi verticalmente.

*PianoStaff*

Identico a `GrandStaff`, ma con il supporto ai nomi degli strumenti a sinistra di ogni sistema.

### Contesti del livello intermedio - righi

*Staff*

Gestisce chiavi, stanghette, armature di chiave, alterazioni. Può contenere contesti `Voice`.

*RhythmicStaff*

Come `Staff` ma per le ritmiche. Le altezze vengono ignorate e le note appaiono su una linea. L'output MIDI conserva le altezze inalterate.

*TabStaff*

Contesto per generare l'intavolatura (o tablatura). La forma predefinita è quella dell'intavolatura per chitarra, con sei linee.

*DrumStaff*

Contesto per gli strumenti percussivi. Può contenere `DrumVoice`.

*VaticanaStaff*

Identico a `Staff`, a parte il fatto che è progettato per rappresentare un brano in stile gregoriano.

*MensuralStaff*

Identico a `Staff`, a parte il fatto che è progettato per rappresentare un brano in stile mensurale.

### Contesti del livello inferiore - voci

I contesti al livello della voce inizializzano alcune proprietà e avviano gli incisori adatti. Un contesto del livello inferiore è privo di `defaultchild`. Sebbene sia possibile far sì che accetti/contenga sottocontesti, questi possono essere creati e inseriti solo esplicitamente.

*Voice*

Corrisponde a una voce di un rigo. Questo contesto gestisce la conversione di segni di dinamica, gambi, travature, apici e pedici, legature di portamento e di valore e pause. Deve essere istanziata esplicitamente se si hanno molteplici voci nello stesso rigo.

*VaticanaVoice*

Identico a `Voice`, a parte il fatto che è progettato per rappresentare un brano in stile gregoriano.

*MensuralVoice*

Identico a `Voice`, con delle modifiche per rappresentare un brano in stile mensurale.

*Lyrics*

Corrisponde a una voce con testo vocale. Gestisce la stampa di una singola linea di testo vocale.

*DrumVoice*

Il contesto della voce usato in un rigo per percussioni.

*FiguredBass*

Il contesto in cui sono creati gli oggetti `BassFigure` a partire dall'input inserito in modalità `\figuremode`.

*TabVoice*

Il contesto della voce usato all'interno di un contesto `TabStaff`. Solitamente omesso così che sia creato implicitamente.

*CueVoice*

Un contesto della voce usato per rappresentare note a dimensione ridotta, inteso soprattutto per aggiungere citazioni in corpo più piccolo a un rigo, vedi [Formattazione delle notine], pagina 217. Solitamente omesso così che sia creato implicitamente.

*ChordNames*

Crea e dispone i nomi degli accordi.

## 5.1.2 Creazione e citazione di un contesto

LilyPond crea i contesti del livello inferiore automaticamente se incontra un'espressione musicale prima di un contesto adatto, ma questo approccio di solito funziona soltanto per partiture semplici o frammenti musicali simili a quelli della documentazione. Per partiture più complesse si consiglia di specificare tutti i contesti esplicitamente coi comandi `\new` o `\context`. La sintassi di questi due comandi è molto simile:
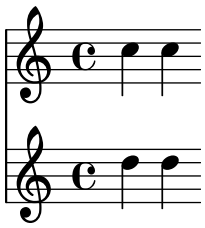
```
[\new | \context] Contesto [ = nome] [espressione-musicale]
```

dove si può usare `\new` o `\context`. *Contesto* è il tipo di contesto che deve essere creato, *nome* è un nome opzionale da assegnare al contesto che si sta creando e *espressione-musicale* è una

singola espressione musicale che deve essere interpretata dagli incisori e dai performer in questo contesto.
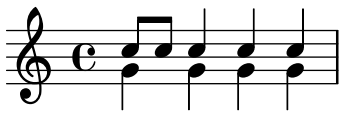
Il prefisso `\new` senza un nome viene usato comunemente per creare partiture con molti righi:

```
<<
  \new Staff \relative {
    % lascia che il contesto Voice sia creato implicitamente
    c''4 c
  }
  \new Staff \relative {
    d''4 d
  }
>>
```



e per mettere varie voci in un rigo:

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    c''8 c c4 c c
  }
  \new Voice \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```
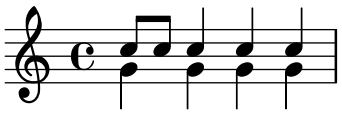


`\new` deve essere sempre usato per specificare contesti senza nome.

La differenza tra `\new` e `\context` sta nell'azione presa:

- `\new` con o senza un nome creerà sempre un contesto del tutto nuovo e distinto, anche se ne esiste già uno con lo stesso nome:

```
\new Staff <<
  \new Voice = "A" \relative {
    \voiceOne
    c''8 c c4 c c
  }
  \new Voice = "A" \relative {
    \voiceTwo
    g'4 g g g
  }
>>
```
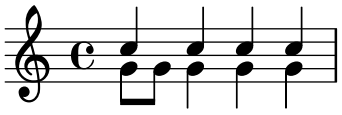


- `\context` seguito da un nome creerà un contesto distinto solo se non esiste già un contesto dello stesso tipo con lo stesso nome nella stessa gerarchia di contesto. Altrimenti sarà preso come riferimento per quel contesto precedentemente creato, e la sua espressione musicale verrà passata a quel contesto per la sua interpretazione.

  Una possibile applicazione dei contesti con nome è la separazione di formattazione della partitura e contenuto musicale. Le seguenti due forme sono entrambe valide:

```
\score {
  <<
    % formattazione della partitura
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>
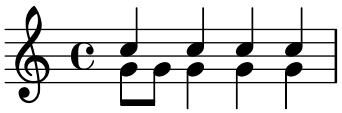
    % contenuto musicale
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



```
\score {
  <<
    % formattazione della partitura
    \new Staff <<
      \context Voice = "one" {
        \voiceOne
      }
      \context Voice = "two" {
        \voiceTwo
      }
    >>

    % contenuto musicale
    \context Voice = "one" {
```

```
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



Altrimenti si possono usare le variabili per ottenere un risultato simile. Vedi Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

- `\context` senza nome corrisponderà al primo di qualsiasi contesto precedentemente creato dello stesso tipo nella stessa gerarchia di contesto, anche uno a cui è stato assegnato un nome, e la sua espressione musicale sarà passata a quel contesto per la sua interpretazione. Questa forma è raramente utile. Tuttavia, si usa `\context` senza nome e senza espressione musicale per impostare il contesto in cui una procedura Scheme specificata con `\applyContext` viene eseguita:

```
\new Staff \relative {
  c'1
  \context Timing
  \applyContext #(lambda (ctx)
                   (newline)
                   (display (ly:context-current-moment ctx)))
  c1
}
```

Un contesto deve essere nominato se deve essere citato successivamente, per esempio quando il testo vocale è associato alla musica:

```
\new Voice = "tenore" musica
...
\new Lyrics \lyricsto "tenore" testo
```

Maggiori informazioni sull'associazione del testo vocale alla musica in [Durate automatiche delle sillabe], pagina 271.

Le proprietà di tutti i contesti di un certo tipo possono essere modificate in un blocco `\layout` (con una sintassi diversa), vedi [Modifica di tutti i contesti dello stesso tipo], pagina 603. Questo costrutto fornisce anche un mezzo per mantenere le istruzioni di formattazione separate dal contenuto musicale. Se occorre modificare un solo contesto, bisogna usare un blocco `\with`, vedi [Modifica di un solo contesto specifico], pagina 605.

## Vedi anche

Manuale di apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

Guida alla notazione: [Modifica di un solo contesto specifico], pagina 605, [Durate automatiche delle sillabe], pagina 271.

### 5.1.3 Conservazione di un contesto

I contesti vengono solitamente terminati nel primo momento musicale in cui non hanno niente da fare. Quindi i contesti `Voice` muoiono appena non contengono eventi; i contesti `Staff` muoiono appena tutti i contesti `Voice` al loro interno non contengono eventi; etc. Ciò può causare difficoltà se contesti precedenti, che sono stati terminati, devono essere richiamati; per esempio, quando si cambia il rigo col comando `\change`, quando si associa il testo a una voce col comando `\lyricsto` o quando si aggiungono ulteriori eventi musicali a un contesto precedente.

C'è un'eccezione a questa regola generale: all'interno di un costrutto `{...}` (musica sequenziale), la nozione di «contesto corrente» scenderà di livello ogni volta che un elemento della sequenza termina in un sottocontesto del precedente contesto corrente. Ciò evita la falsa creazione di contesti impliciti in varie situazioni, ma significa che il primo contesto in cui si scende verrà conservato fino alla fine dell'espressione.

Al contrario, i contesti di un costrutto `<<...>>` (musica simultanea) non sono portati avanti, dunque racchiudendo un comando di creazione di contesto in un'ulteriore coppia di `<<...>>` impedirà che il contesto persista in tutta la sequenza `{...}`.

Un contesto può essere tenuto attivo assicurandosi che abbia qualcosa da fare in ogni momento musicale. I contesti `Staff` sono tenuti attivi assicurandosi che una delle loro voci sia conservata. Un modo per farlo consiste nell'aggiungere pause spaziatrici a una voce in parallelo con la musica vera. Queste devono essere aggiunte a ogni contesto `Voice` da tenere attivo. Se si usano sporadicamente varie voci, è più sicuro tenerle attive invece di tentare di affidarsi alle eccezioni menzionate sopra.

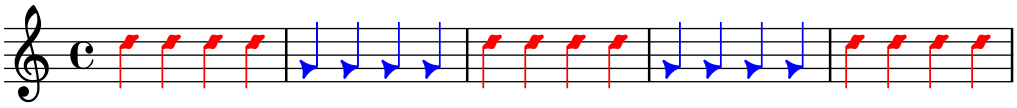
Nell'esempio seguente, sia la voce A che la voce B sono mantenute attive in questo modo per la durata del brano:

```
musicA = \relative { d''4 d d d }
musicB = \relative { g'4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 }  % Tiene la voce "A" attiva per 5 battute
    \new Voice = "B" { s1*5 }  % Tiene la voce "B" attiva per 5 battute
  >>
}

music = {
  \context Voice = "A" {
    \voiceOneStyle
    \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle
    \musicB
  }
  \context Voice = "A" { \musicA }
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

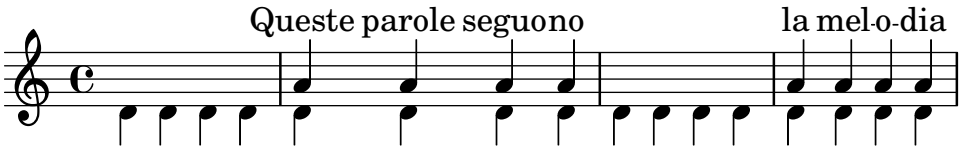
\score {
  \new Staff <<
    \keepVoicesAlive
    \music
```

```
  >>
}
```



L'esempio seguente mostra come scrivere una linea melodica sporadica con testo vocale usando questo approccio. Ovviamente, in una situazione reale la melodia e l'accompagnamento consisterebbero di varie e diverse sezioni.

```
melodia = \relative { a'4 a a a }
accompagnamento = \relative { d'4 d d d }
parole = \lyricmode { Queste parole seguo -- no la mel -- o -- dia }
\score {
  <<
    \new Staff = "musica" {
      <<
        \new Voice = "melodia" {
          \voiceOne
          s1*4  % Tiene la voce "melodia" attiva per 4 battute
        }
        {
          \new Voice = "accompagnamento" {
            \voiceTwo
            \accompagnamento
          }
          <<
            \context Voice = "melodia" { \melodia }
            \context Voice = "accompagnamento" { \accompagnamento }
          >>
          \context Voice = "accompagnamento" { \accompagnamento }
          <<
            \context Voice = "melodia" { \melodia }
            \context Voice = "accompagnamento" { \accompagnamento }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "musica" }
    \lyricsto "melodia" { \parole }
  >>
}
```

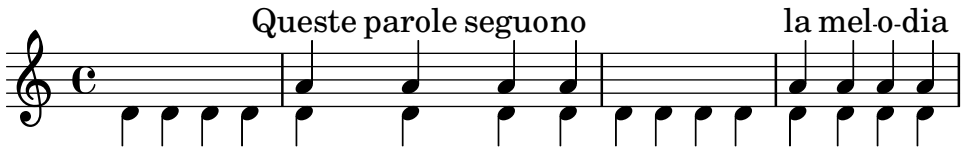



Un modo alternativo, migliore in molte circostanze, è quello di mantenere la linea melodica attiva semplicemente includendo le note spaziatrici in modo che si allineino correttamente con l'accompagnamento:

```
melodia = \relative {
```

```
  s1  % salta una battuta
  a'4 a a a
  s1  % salta una battuta
  a4 a a a
}
accompagnamento = \relative {
  d'4 d d d
  d4 d d d
  d4 d d d
  d4 d d d
}
parole = \lyricmode { Queste parole seguo -- no la mel -- o -- dia }

\score {
  <<
    \new Staff = "musica" {
      <<
        \new Voice = "melodia" {
          \voiceOne
          \melodia
        }
        \new Voice = "accompagnamento" {
          \voiceTwo
          \accompagnamento
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "musica" }
    \lyricsto "melodia" { \parole }
  >>
}
```




### 5.1.4 Modifica dei componenti aggiuntivi di un contesto

I contesti della notazione (come `Score` e `Staff`) non contengono solo le proprietà, ma anche dei componenti aggiuntivi chiamati "engraver" (incisori) che creano gli elementi della notazione. Per esempio, il contesto `Voice` contiene l'incisore `Note_heads_engraver` e il contesto `Staff` contiene l'incisore `Key_engraver`.

Una descrizione completa di ogni componente aggiuntivo si trova in Guida al funzionamento interno ↦ Translation ↦ Engravers. Ogni contesto descritto in Guida al funzionamento interno ↦ Translation ↦ Context. elenca gli incisori usati per quel contesto.
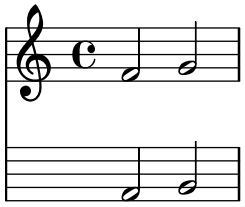
Può essere utile sperimentare questi componenti aggiuntivi. Per farlo si avvia un nuovo contesto con `\new` o `\context` e si modifica:

```
\new contesto \with {
  \consists ...
  \consists ...
  \remove ...
```

```
  \remove ...
  etc.
}
{
  ...musica...
}
```

dove ... deve essere sostituito dal nome dell'incisore. Il comando `\remove` toglie l'incisore, mentre `\consists` lo mantiene. Ecco un semplice esempio che toglie l'incisore dell'indicazione di tempo (`Time_signature_engraver`) e quello della chiave (`Clef_engraver`) da un contesto `Staff`:

```
<<
  \new Staff \relative {
    f'2 g
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } \relative {
    f'2 g2
  }
>>
```
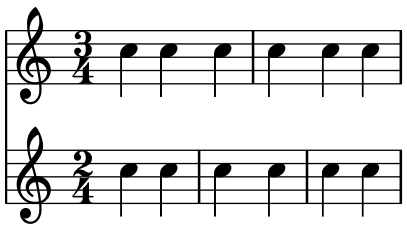


Nel secondo rigo non ci sono indicazioni di tempo né simboli della chiave. Questo è un metodo piuttosto grezzo per far scomparire gli oggetti, dato che ha effetto sull'intero rigo. Questo metodo influenza anche la spaziatura, cosa che potrebbe non essere desiderabile. Metodi più sofisticati per nascondere gli oggetti sono mostrati in Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*.

L'esempio successivo mostra un'applicazione pratica. Le stanghette e le indicazioni di tempo sono normalmente sincronizzate al livello dell'intera partitura. Ciò viene fatto da `Timing_translator` e `Default_bar_line_engraver`. Questo componente aggiuntivo gestisce le indicazioni di tempo, il posizionamento nella misura, etc. Spostando questi incisori dal contesto `Score` al contesto `Staff`, possiamo ottenere una partitura in cui ogni rigo ha la sua indicazione di tempo.

```
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    }
    \relative {
        \time 3/4
        c''4 c c c c c
    }
  \new Staff \with {
```

```
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \relative {
      \time 2/4
      c''4 c c c c
  }
>>
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    }
  }
}
```



### Problemi noti e avvertimenti

L'ordine in cui vengono specificati gli incisori è l'ordine in cui vengono richiamati per compiere la loro elaborazione. Di solito l'ordine in cui gli incisori sono specificati non conta, ma in pochi casi speciali l'ordine è importante, per esempio quando un incisore scrive una proprietà e un altro la legge o quando un incisore crea un grob e un altro lo deve elaborare.

I seguenti ordini sono importanti:

- l'incisore `Bar_engraver` deve essere primo di solito,
- l'incisore `New_fingering_engraver` deve precedere l'incisore `Script_column_engraver`,
- `Timing_translator` deve precedere l'incisore `Bar_number_engraver`.

### Vedi anche

File installati: `ly/engraver-init.ly`.

## 5.1.5 Modifica delle impostazioni predefinite di un contesto

Le proprietà dei contesti e dei grob possono essere modificate con i comandi `\set` e `\override`, come è spiegato in Sezione 5.3 [Modifica delle proprietà], pagina 615. Questi comandi creano eventi musicali, rendendo effettivi i cambiamenti nel momento temporale in cui è elaborata la musica.

Questa sezione spiega invece come cambiare i valori *predefiniti* delle proprietà dei contesti e dei grob nel momento in cui viene creato il contesto. Esistono due modi per farlo. Uno modifica i valori predefiniti in tutti i contesti di un certo tipo, l'altro modifica i valori predefiniti soltanto in una certa istanza di un contesto.

### Modifica di tutti i contesti dello stesso tipo

Le impostazioni di contesto predefinite da usare per l'output grafico nei contesti `Score`, `Staff`, `Voice` e in altri contesti possono essere specificate in un blocco `\context` compreso in un qualsiasi blocco `\layout`.

Le impostazioni per l'output Midi invece devono essere specificate separatamente in blocchi `\midi` (vedi [Definizioni di output - gerarchia dei contesti], pagina 592).
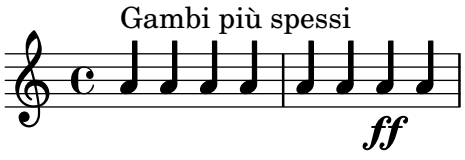
Il blocco `\layout` deve trovarsi all'interno del blocco `\score` al quale si vuole applicare, dopo la musica.

```
\layout {
  \context {
    \Voice
    [impostazioni di contesto per tutti i contesti Voice]
  }
  \context {
    \Staff
    [impostazioni di contesto per tutti i contesti Staff]
  }
}
```

Si possono specificare i seguenti tipi di impostazioni:

- Un comando `\override`, ma col nome del contesto omesso
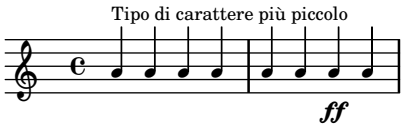
```
\score {
  \relative {
    a'4^"Gambi più spessi" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      \override Stem.thickness = #4.0
    }
  }
}
```




- Impostando direttamente una proprietà di contesto
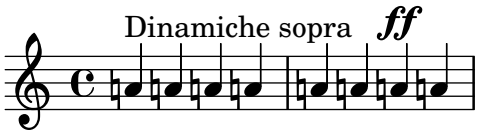
```
\score {
  \relative {
    a'4^"Tipo di carattere più piccolo" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      fontSize = #-4
    }
  }
```

```
}
```




- Un comando predefinito come `\dynamicUp` o un'espressione musicale come `\accidentalStyle dodecaphonic`
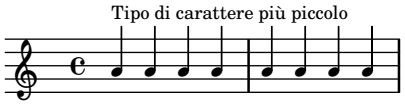
```
\score {
  \relative {
    a'4^"Dinamiche sopra" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}
```




- Una variabile definita dall'utente contenente un blocco `\with`; il blocco `\with` è spiegato approfonditamente in [Modifica di un solo contesto specifico], pagina 605.
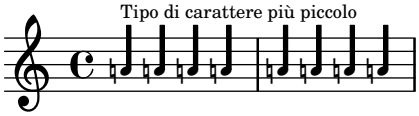
```
StaffDefaults = \with {
  fontSize = #-4
}

\score {
  \new Staff {
    \relative {
      a'4^"Tipo di carattere più piccolo" a a a
      a4 a a a
    }
  }
  \layout {
    \context {
      \Staff
      \StaffDefaults
    }
  }
}
```




I comandi che impostano una proprietà possono essere posti in un blocco `\layout` senza doverli racchiudere in un blocco `\context`. Così facendo si ottiene lo stesso risultato che si otterrebbe se si includessero gli stessi comandi all'inizio di ogni contesto del tipo specificato. Se non è specificato alcun contesto, avranno effetto su *qualsiasi* contesto di basso livello, vedi [Contesti del livello inferiore - voci], pagina 594. La sintassi di un comando di impostazione della proprietà in un blocco `\layout` è identico a quello che si userebbe in mezzo alle note.

```
\score {
  \new Staff {
    \relative {
      a'4^"Tipo di carattere più piccolo" a a a
      a4 a a a
    }
  }
  \layout {
    \accidentalStyle dodecaphonic
    \set fontSize = #-4
    \override Voice.Stem.thickness = #4.0
  }
}
```




## Modifica di un solo contesto specifico

Le proprietà di contesto di una sola istanza di un contesto specifico possono essere cambiate in un blocco `\with`. Tutte le altre istanze di contesto dello stesso tipo manterranno le impostazioni predefinite di LilyPond e saranno modificate da qualsiasi blocco `\layout` che le riguarda. Il blocco `\with` deve essere posto subito dopo il comando `\new` *tipo-contesto*:

```
\new Staff \with { [impostazioni di contesto per questa istanza di contesto soltanto] }
{
  ...
}
```

Altrimenti, se la musica viene inserita usando la forma breve dei comandi che specificano la modalità di inserimento, per esempio `\chords` invece di `\chordmode`, il comando `\with` deve essere posto subito dopo il comando che specifica la modalità:

```
\chords \with { [impostazioni di contesto per questo contesto (implicito) soltanto] }
{
  ...
}
```

perché è il contesto implicito creato da queste forme brevi che deve essere modificato. Vale la stessa considerazione per le altre forme brevi che indicano la modalità di inserimento (`\drums`, `\figures`), vedi Sezione 5.4.1 [Modalità di inserimento], pagina 629.
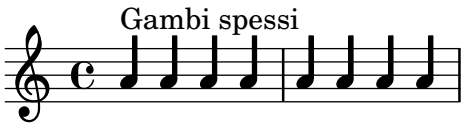
Le 'modifiche di contesto', essendo specificate in blocchi `\with` che si trovano all'interno della musica, avranno effetto su *tutti* gli output (quello grafico *e* il Midi), diversamente da quanto avviene per le modifiche inserite in una definizione di output.

Si possono specificare i seguenti tipi di impostazioni:

- Un comando `\override`, ma col nome di contesto omesso
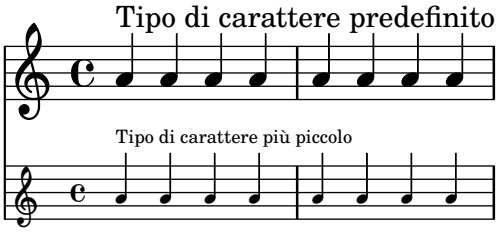
```
\score {
```

```
  \new Staff {
    \new Voice \with { \override Stem.thickness = #4.0 }
    {
      \relative {
        a'4^"Gambi spessi" a a a
        a4 a a a
      }
    }
  }
}
```




- Impostando direttamente una proprietà di contesto
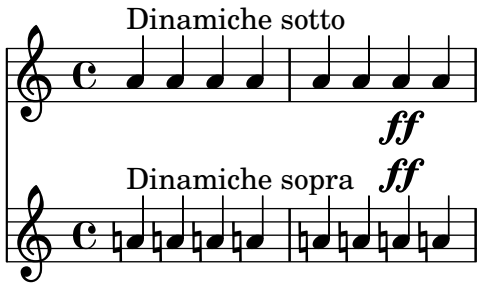
```
\score {
  <<
    \new Staff {
      \relative {
        a'4^"Tipo di carattere predefinito" a a a
        a4 a a a
      }
    }
    \new Staff \with { fontSize = #-4 }
    {
      \relative {
        a'4^"Tipo di carattere più piccolo" a a a
        a4 a a a
      }
    }
  >>
}
```




- Un comando predefinito come `\dynamicUp`

```
\score {
  <<
    \new Staff {
      \new Voice {
        \relative {
          a'4^"Dinamiche sotto" a a a
          a4 a a\ff a
        }
      }
```

```
    }
    \new Staff \with { \accidentalStyle dodecaphonic }
    {
      \new Voice \with { \dynamicUp }
      {
        \relative {
          a'4^"Dinamiche sopra" a a a
          a4 a a\ff a
        }
      }
    }
  >>
}
```




### Vedi anche

Guida alla notazione: Sezione 5.4.1 [Modalità di inserimento], pagina 629,

### Ordine di precedenza

Il valore di una proprietà da applicare in un certo momento viene determinato nel modo seguente:

- se un comando `\override` o `\set` nel flusso dell'input (le note) è attivo viene usato quel valore,
- altrimenti viene usato il valore predefinito preso da una dichiarazione `\with` nella dichiarazione di iniziazione del contesto,
- altrimenti viene usato il valore predefinito preso dal blocco `\context` appropriato più recente nei blocchi `\layout` o `\midi`,
- altrimenti viene usato il valore predefinito in LilyPond.

### Vedi anche

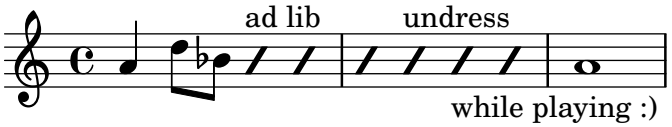
Manuale di apprendimento: Sezione "Modificare le proprietà di contesto" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.1 [Tutto sui contesti], pagina 592, [Contesti del livello inferiore - voci], pagina 594, Sezione 5.3.2 [Il comando set], pagina 615, Sezione 5.3.3 [Il comando override], pagina 617, Sezione 4.2.1 [Il blocco layout], pagina 550.

## 5.1.6 Definizione di nuovi contesti

Contesti specifici, come `Staff` e `Voice`, sono creati a partire da semplici mattoncini. È possibile creare nuovi tipi di contesto con diverse combinazioni di incisori.

Il prossimo esempio mostra come costruire un diverso tipo di contesto `Voice` da zero. Sarà simile a `Voice`, ma stamperà soltanto teste di nota a forma di barra posizionate al centro. Può essere usato per indicare l'improvvisazione nei brani jazz:




Queste impostazioni sono definite in un blocco `\context` compreso in un blocco `\layout`:

```
\layout {
  \context {
    ...
  }
}
```

L'input di esempio che segue sostituisce i ... del frammento precedente.

Per prima cosa occorre definire un nome per il nuovo contesto:

```
\name ImproVoice
```

Essendo simile al contesto `Voice`, ci servono comandi che funzionano in contesti `Voice` esistenti per far sì che continuino a funzionare. Per farlo si assegna al nuovo contesto un alias di `Voice`,

```
\alias Voice
```

Il contesto stamperà note e testi di spiegazione, dunque dobbiamo aggiungere gli incisori che forniscono queste funzionalità, più l'incisore che raggruppa in colonne le note, i gambi e le pause che si trovano nello stesso momento musicale:

```
\consists "Note_heads_engraver"
\consists "Text_engraver"
\consists "Rhythmic_column_engraver"
```

Le teste di nota devono essere poste tutte sulla linea centrale:

```
\consists "Pitch_squash_engraver"
squashedPosition = #0
```

L'incisore `Pitch_squash_engraver` modifica le teste di nota (create dall'incisore `Note_heads_engraver`) e imposta la loro posizione verticale sul valore di `squashedPosition`, in questo caso `0`, la linea centrale.

Le note appaiono come una barra e non hanno gambi:

```
\override NoteHead.style = #'slash
\hide Stem
```

Tutti questi componenti aggiuntivi devono comunicare sotto il controllo del contesto. I meccanismi con cui i contesti comunicano sono stabiliti dichiarando il tipo di contesto con `\type`. In un blocco `\layout`, la maggior parte dei contesti sarà del tipo `Engraver_group`. Alcuni contesti speciali e i contesti nei blocchi `\midi` usano altri tipi di contesto. Copiare e modificare una definizione di contesto esistente comprenderà anche la definizione del tipo. Poiché questo esempio crea una definizione da zero, deve essere specificato in modo esplicito.

```
\type "Engraver_group"
```

Mettendo tutte queste parti insieme otteniamo:

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Pitch_squash_engraver"
  squashedPosition = #0
  \override NoteHead.style = #'slash
  \hide Stem
```

```
    \alias Voice
  }
```

I contesti formano gerarchie. Vogliamo posizionare il contesto `ImproVoice` all'interno del contesto `Staff`, proprio come i normali contesti `Voice`. Cambiamo quindi la definizione di `Staff` col comando `\accepts`,

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Spesso quando si riutilizza una definizione di contesto esistente, il contesto risultante può essere usato in qualsiasi situazione in cui il contesto originale sarebbe stato utile.

```
\layout {
  ...
  \inherit-acceptability a da
}
```

farà sì che i contesti del tipo *a* siano accettati da tutti i contesti che accettano anche *da*. Per esempio, usando

```
\layout {
  ...
  \inherit-acceptability "ImproVoice" "Voice"
}
```

verrà aggiunto un `\accepts` per `ImproVoice` a entrambe le definizioni di `Staff` e `RhythmicStaff`.

L'opposto di `\accepts` è `\denies`, che è talvolta necessario se si riusano definizioni di contesto esistenti.

Sistemando i pezzi necessari in un blocco `\layout` ci porta a:

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \inherit-acceptability "ImproVoice" "Voice"
}
```

Quindi l'output all'inizio di questa parte può essere inserito così:

```
\relative {
  a'4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"undress"
    c c_"while playing :)"
  }
  a1
}
```

Per completare questo esempio, le modifiche che alterano la gerarchia dei contesti devono essere ripetute in un blocco `\midi` per far sì che l'output Midi dipenda dalle stesse relazioni di contesto.

### Vedi anche

Guida al funzionamento interno: Sezione “Note_heads_engraver” in *Guida al Funzionamento Interno*, Sezione “Text_engraver” in *Guida al Funzionamento Interno*, Sezione “Rhythmic_column_engraver” in *Guida al Funzionamento Interno*, Sezione “Pitch_squash_engraver” in *Guida al Funzionamento Interno*.

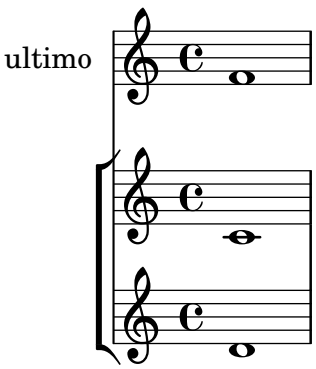
## 5.1.7 Ordine di disposizione dei contesti

Normalmente i contesti in un sistema sono posizionati dall’alto verso il basso nell’ordine in cui sono incontrati nel file di input. Quando i contesti sono annidati, il contesto più esterno includerà i contesti annidati al suo interno come specificato nel file di input solo se i contesti interni sono inclusi nell’elenco “accepts” del contesto più esterno; altrimenti saranno riposizionati sotto il contesto più esterno invece di essere annidati al suo interno.

L’elenco “accepts” di un contesto può essere modificato coi comandi `\accepts` o `\denies`. `\accepts` aggiunge un contesto all’elenco “accepts” mentre `\denies` lo rimuove dall’elenco.

Per esempio, un gruppo di righi racchiusi da una parentesi quadra non si trova solitamente all’interno di un gruppo di righi con stanghette connesse e racchiusi da una graffa, e `GrandStaff` non accetta al suo interno `StaffGroup` per impostazione predefinita.

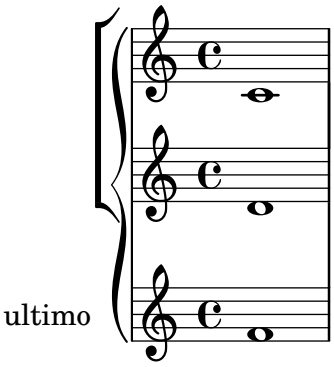
```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = ultimo f'1 }
  >>
}
```




Tuttavia col comando `\accepts` si può aggiungere `StaffGroup` al contesto `GrandStaff`:

```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = ultimo f'1 }
  >>
  \layout {
    \context {
      \GrandStaff
```

```
      \accepts "StaffGroup"
    }
  }
}
```




`\denies` si usa soprattutto quando un nuovo contesto è basato su un altro, ma l'annidamento richiesto differisce. Per esempio, il contesto `VaticanaStaff` è basato sul contesto `Staff`, ma col contesto `VaticanaVoice` che sostituisce il contesto `Voice` nell'elenco "accepts".

Nota bene: un contesto verrà creato implicitamente e senza avviso se si incontra un comando quando non c'è un contesto adatto disponibile per contenerlo.

Nella definizione di un contesto, il tipo di sottocontesto da creare implicitamente viene specificato con `\defaultchild`. Vari eventi musicali richiedono un contesto "di base": quando si incontra tale evento, vengono creati ricorsivamente i sottocontesti finché non si raggiunge un contesto privo di impostazione '`defaultchild`'.

La creazione implicita di contesti può talvolta causare nuovi righi o nuove partiture non attesi. Per evitare questi problemi è consigliabile usare `\new` per creare i contesti esplicitamente.

Talvolta un contesto deve esistere solo per un breve periodo, come nel caso esemplare di un contesto rigo per un ossia. Per ottenere ciò, di solito si introduce la definizione di contesto nel punto appropriato in parallelo con la sezione corrispondente della musica principale. Per impostazione predefinita, il contesto temporaneo verrà posizionato sotto tutti i contesti esistenti. Per riposizionarlo sopra il contesto chiamato "principale", dovrebbe essere definito in questo modo:

```
\new Staff \with { alignAboveContext = "principale" }
```

Una situazione simile si pone quando si posiziona un contesto temporaneo per il testo vocale all'interno di una disposizione multirigo come `ChoirStaff`; per esempio, quando si aggiunge una seconda strofa a una sezione ripetuta. Per impostazione predefinita, il contesto temporaneo per il testo vocale verrà posizionato sotto i righi più bassi. Definendo il contesto temporaneo per il testo vocale con `alignBelowContext`, questo può essere posizionato correttamente sotto i contesti della voce (con nome) che contengono la prima strofa.

Esempi che mostrano questo riposizionamento di contesti temporanei possono essere trovati altrove — vedi Sezione "Annidare le espressioni musicali" in *Manuale di Apprendimento*, Sezione 1.6.2 [Modificare singoli righi], pagina 201, e Sezione 2.1.2 [Tecniche specifiche per il testo vocale], pagina 279.

## Vedi anche

Manuale di apprendimento: Sezione "Annidare le espressioni musicali" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.6.2 [Modificare singoli righi], pagina 201, Sezione 2.1.2 [Tecniche specifiche per il testo vocale], pagina 279.

Utilizzo: Sezione “Appare un rigo in più” in *Uso del Programma*.
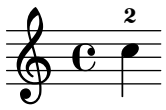
File installati: `ly/engraver-init.ly`.

## 5.2 Come funziona la Guida al funzionamento interno

### 5.2.1 Navigazione nella guida al programma

Supponiamo di voler togliere l’indicazione di diteggiatura nel seguente frammento:

```
c''-2
```



Consultando la documentazione sulle istruzioni di diteggiatura (in [Indicazioni di diteggiatura], pagina 226), si nota:

> **Vedi anche**
>
> Guida al funzionamento interno: Sezione “Fingering” in *Guida al Funzionamento Interno*.

La guida del programmatore è disponibile come documento HTML. È caldamente consigliata la lettura della versione HTML, accessibile online o in locale scaricando la documentazione HTML. Questa sezione sarà molto più difficile da comprendere se si usa il manuale PDF.

Seguire il collegamento a Sezione “Fingering” in *Guida al Funzionamento Interno*. In cima alla pagina si vede

> Fingering objects are created by: Sezione “Fingering_engraver” in *Guida al Funzionamento Interno* and Sezione “New_fingering_engraver” in *Guida al Funzionamento Interno*.

Seguendo i collegamenti correlati nella guida del programma, possiamo seguire il flusso di informazione del programma:

- Sezione “Fingering” in *Guida al Funzionamento Interno*: Sezione “Fingering” in *Guida al Funzionamento Interno* objects are created by: Sezione “Fingering_engraver” in *Guida al Funzionamento Interno*
- Sezione “Fingering_engraver” in *Guida al Funzionamento Interno*: Music types accepted: Sezione “fingering-event” in *Guida al Funzionamento Interno*
- Sezione “fingering-event” in *Guida al Funzionamento Interno*: Music event type `fingering-event` is in Music expressions named Sezione “FingeringEvent” in *Guida al Funzionamento Interno*

Questo percorso va in direzione contraria al flusso dell’informazione nel programma: parte dall’output e termina nell’evento di input. Si può anche partire da un evento di input e leggere seguendo il flusso di informazione arrivando infine all’oggetto di output.

La guida al programma può anche essere sfogliata come un normale documento. Contiene capitoli su `Music definitions` su Sezione “Translation” in *Guida al Funzionamento Interno*, e sul Sezione “Backend” in *Guida al Funzionamento Interno*. Ogni capitolo elenca tutte le definizioni usate e tutte le proprietà che possono essere ritoccate.

### 5.2.2 Interfacce di formattazione

La pagina HTML che abbiamo incontrato nella sezione precedente descrive l’oggetto di formattazione chiamato Sezione “Fingering” in *Guida al Funzionamento Interno*. Tale oggetto è un simbolo interno alla partitura. Ha delle proprietà che contengono numeri (come gli spessori e le

direzioni), ma anche collegamenti a oggetti correlati. Un oggetto di formattazione viene chiamato anche *Grob*, che è un diminutivo di Graphical Object (oggetto grafico). Maggiori dettagli sui Grob si trovano in Sezione "grob-interface" in *Guida al Funzionamento Interno*.

La pagina relativa al grob `Fingering` elenca le definizioni per l'oggetto `Fingering`. Per esempio, la pagina dice

> `padding` (dimension, in staff space):
>
> `0.5`

che significa che il numero verrà tenuto a una distanza di almento 0.5 spazi rigo dalla testa della nota.

Ogni oggetto di formattazione può avere varie funzioni come elemento della notazione o tipografico. Per esempio, l'oggetto Fingering ha i seguenti aspetti

- La sua dimensione è indipendente dalla spaziatura orizzontale, diversamente da legature di portamento o travature.
- È un frammento testuale, solitamente molto breve.
- Questo frammento di testo viene rappresentato con un tipo di carattere, diversamente da legature di portamento o travature.
- Orizzontalmente, il centro del simbolo deve essere allineato al centro della testa della nota.
- Verticalmente, il simbolo viene posto accanto alla nota e sul rigo.
- La posizione verticale viene coordinata anche con altri simboli di apici o pedici.

Ognuno di questi aspetti viene registrato nelle cosiddette interfacce (*interface*), che sono elencate in fondo alla pagina Sezione "Fingering" in *Guida al Funzionamento Interno*:

> This object supports the following interfaces: Sezione "item-interface" in *Guida al Funzionamento Interno*, Sezione "self-alignment-interface" in *Guida al Funzionamento Interno*, Sezione "side-position-interface" in *Guida al Funzionamento Interno*, Sezione "text-interface" in *Guida al Funzionamento Interno*, Sezione "text-script-interface" in *Guida al Funzionamento Interno*, Sezione "font-interface" in *Guida al Funzionamento Interno*, Sezione "finger-interface" in *Guida al Funzionamento Interno*, and Sezione "grob-interface" in *Guida al Funzionamento Interno*.

Facendo clic su uno di questo collegamenti si arriva alla pagina dell'interfaccia del rispettivo oggetto. Ogni interfaccia ha una serie di proprietà. Alcune non sono a disposizione dell'utente ('Internal properties'), ma altre possono essere modificate.

Abbiamo parlato dell'oggetto `Fingering`, ma in realtà non si tratta di niente di troppo complesso. Il file di inizializzazione (vedi Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*) `scm/define-grobs.scm` mostra l'anima di questo "oggetto",

```
(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
```
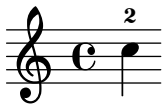
```
font-interface
text-script-interface
text-interface
side-position-interface
self-alignment-interface
item-interface))))))
```

Come si può vedere, l'oggetto `Fingering` non è niente più di un insieme di impostazioni di variabili e la pagina web nella Guida al funzionamento interno è generata direttamente da questa definizione.

### 5.2.3 Determinazione della proprietà del grob

Volevamo cambiare la posizione del **2** in

```
c''-2
```



Dato che il **2** è posizionato verticalmente vicino alla sua nota, dobbiamo fare delle prove con l'interfaccia associata a questo posizionamento, ovvero `side-position-interface`. La pagina relativa a questa interfaccia dice

> `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?)

Sotto questa descrizione, la variabile `padding` viene descritta come
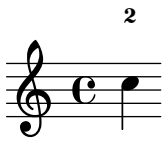
> `padding` (dimension, in staff space)
>
> Add this much extra space between objects that are next to each other.

Aumentando il valore di `padding`, possiamo spostare la diteggiatura lontana dalla testa di nota. Il seguente comando inserisce "tre spazi rigo" di distanza tra la nota e un segno di diteggiatura:

```
\once \override Voice.Fingering.padding = #3
```

L'inserimento di spazio prima che l'oggetto della diteggiatura venga creato produce il seguente risultato:

```
\once \override Voice.Fingering.padding = #3
c''-2
```



In questo caso, il contesto per questa modifica è `Voice`. Come è scritto nella pagina sull'incisore Sezione "Fingering_engraver" in *Guida al Funzionamento Interno*:

> Fingering_engraver is part of contexts: ... Sezione "Voice" in *Guida al Funzionamento Interno*

### 5.2.4 Convenzioni sui nomi

È importante conoscere le convenzioni sui nomi. Ecco una panoramica:

- funzioni Scheme: minuscolo-con-trattini (inclusi i nomi di una-parola)
- funzioni Scheme specifiche di LilyPond: ly:plus-scheme-style
- eventi, classi e proprietà musicali: come-le-funzioni-scheme
- interfacce Grob: stile-scheme
- proprietà del backend: stile-scheme (ma ammette X e Y!)
- contesti (e EspressioniMusicali e grob): Maiuscolo o CamelCase
- proprietà di contesto: minuscoloSeguitoDaCamelCase
- incisori: Maiuscolo_seguito_da_minuscolo_e_con_trattini_bassi

Domande che aspettano una risposta:

- Quali di queste sono convenzioni e quali sono regole?
- Quali sono regole del linguaggio sottostante e quali sono specifiche di LilyPond?

## 5.3 Modifica delle proprietà

### 5.3.1 Panoramica sulla modifica delle proprietà

Ogni contesto è responsabile della creazione di certi tipi di oggetti grafici. Le impostazioni usate per la rappresentazione grafica di questi oggetti sono anch'esse salvate dal contesto. Cambiando queste impostazioni, si può alterare l'aspetto degli oggetti.

Ci sono due tipi diversi di proprietà salvate nei contesti: le proprietà del contesto e quelle del grob. Le proprietà del contesto sono proprietà che si applicano al contesto nel suo complesso e regolano il modo in cui il contesto stesso viene mostrato. Al contrario, le proprietà del grob si applicano a tipi di grob specifici che verranno mostrati nel contesto.

I comandi `\set` e `\unset` vengono usati per cambiare i valori delle proprietà di contesto. I comandi `\override` e `\revert` servono a cambiare i valori delle proprietà dei grob.

#### Vedi anche

Guida al funzionamento interno: Sezione "Backend" in *Guida al Funzionamento Interno*, Sezione "All layout objects" in *Guida al Funzionamento Interno*, Sezione "OverrideProperty" in *Guida al Funzionamento Interno*, Sezione "RevertProperty" in *Guida al Funzionamento Interno*, Sezione "PropertySet" in *Guida al Funzionamento Interno*.

#### Problemi noti e avvertimenti

Il backend non è molto severo nel controllo del tipo delle proprietà di un oggetto. Riferimenti ciclici nei valori Scheme delle proprietà possono causare attese o crash, o entrambi.

### 5.3.2 Il comando `\set`

Ogni contesto ha un insieme di *proprietà*, variabili contenute in quel contesto. Le proprietà di contesto si cambiano col comando `\set`, che ha la seguente sintassi:

```
\set contesto.proprietà = #valore
```

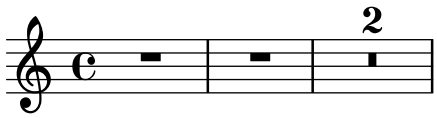
*valore* è un oggetto Scheme, ecco perché deve essere preceduto dal carattere `#`.

Le proprietà dei contesti sono solitamente nominate in `studlyCaps`. Perlopiù regolano la traduzione dalla musica alla notazione, per esempio `localAlterations` (per determinare se stampare le alterazioni), o `measurePosition` (per determinare quando stampare una stanghetta). Le proprietà di contesto possono cambiare nel tempo nel corso dell'interpretazione di un

brano musicale; `measurePosition` ne è un ovvio esempio. Le proprietà di contesto si modificano con `\set`.
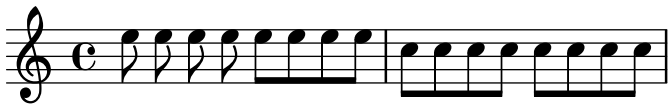
Per esempio, le pause multiple sono combinate in una singola battuta se la proprietà di contesto `skipBars` viene impostata su `#t`:

```
R1*2
\set Score.skipBars = ##t
R1*2
```




Se l'argomento *contesto* non viene specificato, la proprietà sarà impostata nel contesto di base corrente (solitamente `ChordNames`, `Voice`, `TabVoice` o `Lyrics`).
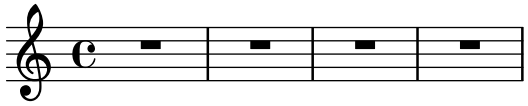
```
\set Score.autoBeaming = ##f
\relative {
  e''8 e e e
  \set autoBeaming = ##t
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```



La modifica viene applicata 'al volo', durante l'interpretazione della musica, in modo che l'impostazione abbia effetto soltanto sul secondo gruppo di note da un ottavo.

Nota che il contesto più inferiore non contiene sempre la proprietà che si desidera cambiare. Per esempio, il tentativo di impostare la proprietà `skipBars` del contesto di base predefinito, in questo caso `Voice`, non produrrà alcun risultato, perché skipBars è una proprietà del contesto `Score`.

```
R1*2
\set skipBars = ##t
R1*2
```



I contesti sono gerarchici, per cui se viene specificato un contesto che ne racchiude altri, per esempio `Staff`, il cambiamento viene applicato anche a tutti i contesti `Voice` del rigo corrente.
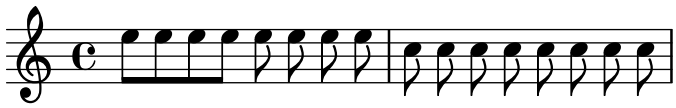
Il comando `\unset`:

`\unset` *`contesto.proprietà`*

viene usato per togliere la definizione di *proprietà* dal *contesto*. Questo comando rimuove la definizione solo se impostata nel *contesto*. Le proprietà impostate nei contesti più esterni non saranno modificate da un `\unset` in un contesto più interno:

```
\set Score.autoBeaming = ##t
```

```
\relative {
  \unset autoBeaming
  e''8 e e e
  \unset Score.autoBeaming
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```
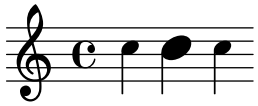


Come in `\set`, non è necessario specificare l'argomento *contesto* per un contesto di fondo, quindi le due dichiarazioni

```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

sono equivalenti se l'attuale contesto di fondo è `Voice`.

Anteponendo `\once` ai comandi `\set` o `\unset`, l'impostazione verrà applicata soltanto a un singolo momento temporale:

```
c''4
\once \set fontSize = #4.7
c''4
c''4
```



Una descrizione completa di tutte le proprietà di contesto disponibili si trova nella Guida al funzionamento interno, vedi Translation ↦ Tunable context properties.

## Vedi anche

Guida al funzionamento interno: Sezione "Tunable context properties" in *Guida al Funzionamento Interno*.

### 5.3.3 Il comando `\override`
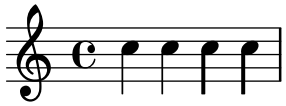
Esiste un tipo particolare di proprietà di contesto: la descrizione del grob. Le descrizioni dei grob vengono nomimate in `StudlyCaps` (iniziando con lettere maiuscole). Contengono le "impostazioni predefinite" per un certo tipo di grob, come una lista associativa. Consultare `scm/define-grobs.scm` per vedere le impostazioni per ogni descrizione di grob. Le descrizioni di grob si modificano con `\override`.

La sintassi del comando `\override` è

```
\override [contesto.]NomeGrob.proprietà = #valore
```
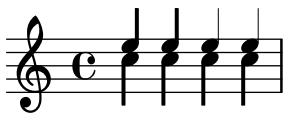
Per esempio, possiamo aumentare lo spessore del gambo di una nota modificando la proprietà `thickness` dell'oggetto `Stem`:

```
c''4 c''
\override Voice.Stem.thickness = #3.0
c''4 c''
```



Se non viene specificato il contesto in un `\override`, viene usato il contesto di base:

```
\override Staff.Stem.thickness = #3.0
<<
  \relative {
    e''4 e
    \override Stem.thickness = #0.5
    e4 e
  } \\
  \relative {
    c''4 c c c
  }
>>
```



Alcune opzioni modificabili vengono chiamate 'sottoproprietà' e stanno dentro le proprietà. Per cambiarle, usare i comandi nella forma

```
\override Stem.details.beamed-lengths = #'(4 4 3)
```

o, per modificare le estremità degli estensori, usare una forma simile alle seguenti

```
\override TextSpanner.bound-details.left.text = "left text"
\override TextSpanner.bound-details.right.text = "right text"
```
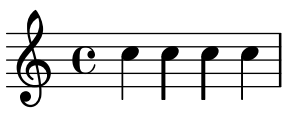
Gli effetti di `\override` possono essere annullati con `\revert`.

La sintassi del comando `\revert` è

```
\revert [contesto.]NomeGrob.proprietà
```

Per esempio,
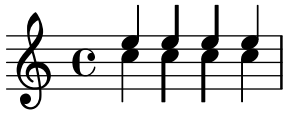
```
\relative {
  c''4
  \override Voice.Stem.thickness = #3.0
  c4 c
  \revert Voice.Stem.thickness
  c4
}
```



Gli effetti di `\override` e `\revert` si applicano a tutti i grob nel contesto interessato da quel momento in poi:
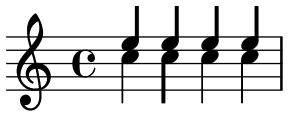
```
<<
  \relative {
    e''4
    \override Staff.Stem.thickness = #3.0
    e4 e e
  } \\
  \relative {
```

```
    c''4 c c
    \revert Staff.Stem.thickness
    c4
  }
>>
```



Si può usare `\once` insieme a `\override` o `\revert` per agire solo sul momento corrente:

```
<<
  \relative c {
    \override Stem.thickness = #3.0
    e''4 e e e
  } \\
  \relative {
    c''4
    \once \override Stem.thickness = #3.0
    c4 c c
  }
>>
```



## Vedi anche

Guida al funzionamento interno: Sezione “Backend” in *Guida al Funzionamento Interno*

### 5.3.4 Il comando `\tweak`

Modificare le proprietà dei grob con `\override` fa sì che i cambiamenti siano applicati a tutti i grob in questione in quel contesto nel momento in cui la modifica viene applicata. Talvolta, tuttavia, si può voler applicare le modifiche a un solo grob invece che a tutti i grob del contesto interessato. Per farlo si usa il comando `\tweak`, che ha la seguente sintassi:

`\tweak [`*`oggetto-formattazione`*`.]`*`proprietà-grob valore`*

*oggetto-formattazione* è opzionale. Il comando `\tweak` viene applicato all’oggetto musicale che segue immediatamente *valore* nel flusso musicale.
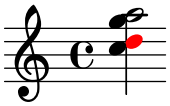
Per un’introduzione alla sintassi e agli usi del comando tweak leggere Sezione “Metodi di modifica” in *Manuale di Apprendimento*.

Quando molti elementi simili sono disposti nello stesso momento musicale, non è possibile usare il comando `\override` per modificarne soltanto uno. È in casi come questi che occorre usare il comando `\tweak`. Ecco alcuni elementi che possono apparire più di una volta nello stesso momento musicale:

- teste di note di un accordo
- segni di articolazione su una singola nota
- legature di valore tra note di un accordo
- parentesi di gruppi irregolari che iniziano nello stesso momento
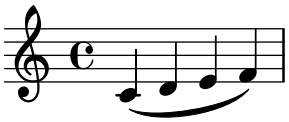
In questo esempio, il colore di una testa di nota e il tipo di testa di un'altra nota sono modificati all'interno di un accordo:

```
< c''
  \tweak color #red
  d''
  g''
  \tweak duration-log #1
  a''
> 4
```
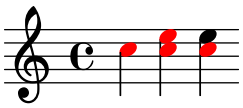


`\tweak` può essere usato per modificare le legature di portamento:

```
\relative { c'-\tweak thickness #5 ( d e f) }
```



Perché il comando `\tweak` funzioni, deve trovarsi proprio accanto all'oggetto al quale deve essere applicato dopo che il file di input è stato convertito in un flusso musicale. Modificare un intero accordo non cambia niente perché il suo evento musicale agisce solo come contenitore, mentre tutti gli oggetti della formattazione sono creati dagli eventi interni all' `EventChord`:
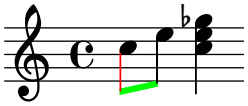
```
\tweak color #red c''4
\tweak color #red <c'' e''>4
<\tweak color #red c'' e''>4
```



Il semplice comando `\tweak` non può essere usato per modificare un oggetto che non sia creato direttamente dall'input. In particolare, non agirà su gambi, travature automatiche o alterazioni, dato che questi sono generati successivamente dagli oggetti della formattazione di `NoteHead` invece che da elementi musicali nel flusso dell'input.

Tali oggetti della formattazione creati indirettamente possono essere tuttavia modificati usando una forma del comando `\tweak` in cui il nome del grob è indicato esplicitamente:
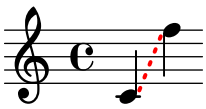
```
\tweak Stem.color #red
\tweak Beam.color #green c''8 e''
<c'' e'' \tweak Accidental.font-size #-3 ges''>4
```



`\tweak` non può essere usato per modificare le chiavi o le indicazioni di tempo, perché queste vengono separate da qualsiasi comando `\tweak` precedente nel flusso dell'input a causa dell'inserimento automatico di ulteriori elementi richiesti per specificare il contesto.

Si possono usare vari comandi `\tweak` prima di un elemento della notazione e tutti avranno effetto su di esso:

```
c'
  -\tweak style #'dashed-line
  -\tweak dash-fraction #0.2
  -\tweak thickness #3
  -\tweak color #red
  \glissando
f''
```



Il flusso musicale generato da una sezione di un file di input, compresi gli elementi inseriti automaticamente, può essere esaminato, vedi Sezione "Displaying music expressions" in *Estendere*. Ciò può essere utile nel determinare cosa può essere modificato da un comando `\tweak` o nel valutare come aggiustare l'input per far sì che un `\tweak` sia applicato.

### Vedi anche

Manuale di apprendimento: Sezione "Metodi di modifica" in *Manuale di Apprendimento*.

Estendere LilyPond: Sezione "Displaying music expressions" in *Estendere*.

### Problemi noti e avvertimenti

Il comando `\tweak` non può essere usato per modificare i punti di controllo di una sola legatura di valore tra tante in un accordo, se non quelli della prima legatura incontrata nel file di input.

## 5.3.5 `\set` vs. `\override`

I comandi `\set` e `\override` manipolano le proprietà associate ai contesti. In entrambi i casi, le proprietà seguono una *gerarchia dei contesti*; proprietà non impostate in un contesto mostreranno i valori del rispettivo contesto padre.

La durata e il valore di una proprietà di contesto sono dinamici e disponibili soltanto mentre la musica viene interpretata (ovvero 'reiterata'). Nel momento della creazione del contesto, le proprietà sono inizializzate a partire dalle definizioni corrispondenti (insieme a altre modifiche) di quel contesto. Qualsiasi modifica successiva viene fatta con dei comandi di impostazione della proprietà presenti nella musica stessa.

Le definizioni degli oggetti grafici (o "grob") sono una categoria *speciale* di proprietà di contesto, perché la loro struttura e uso sono diversi da quelli delle normali proprietà di contesto. Diversamente da quest'ultime, le definizioni dei grob sono suddivise in *proprietà dei grob*.

Inoltre, diversamente dalle normali proprietà di contesto, le definizioni dei grob hanno una loro 'contabilità' interna usata per tenere traccia delle loro individuali proprietà e sottoproprietà. Ciò significa che è possibile definire queste parti in contesti diversi e avere sempre a disposizione la definizione complessiva del grob nel momento della sua creazione, assemblata da tutti i pezzi compresi tra il contesto corrente e i suoi contesti superiori.

Un grob viene solitamente creato da un incisore nel momento in cui l'espressione musicale viene interpretata e riceve le sue proprietà iniziali dalla definizione del grob corrente del contesto dell'incisore. L'incisore (o altre parti del 'backend' di LilyPond) può successivamente cambiare (o aggiungere) le proprietà iniziali del grob, ma ciò non ha effetto sulla definizione di grob del contesto.

Ciò che LilyPond chiama 'proprietà del grob' nel contesto delle modifiche a livello di utente sono in realtà le proprietà della definizione di grob di un *contesto*.

Le definizioni di grob sono manipolate con `\override` e `\revert` e hanno un nome che inizia con una lettera maiuscola (come '`NoteHead`'), mentre le normali proprietà di contesto sono manipolate con `\set` e `\unset` e il loro nome inizia con una lettera minuscola.

I comandi `\tweak` e `\overrideProperty` modificano le proprietà dei grob bypassando completamente tutte le proprietà di contesto. Catturano i grob mentre vengono creati, impostando le loro proprietà per un evento musicale (`\tweak`) o, nel caso di `\overrideProperty`, per una sovrascrittura specifica.

### 5.3.6 Il comando `\offset`

Sebbene sia possibile impostare le proprietà dei grob su nuovi valori coi comandi `\override`, `\tweak` e `\overrideProperty`, è spesso più opportuno modificare tali proprietà in modo relativo a un valore predefinito. A questo fine è stato creato il comando `\offset`.

La sintassi di `\offset` è

`[-]\offset` *`proprietà offset elemento`*

Il comando funziona aggiungendo i contenuti di *offset* all'impostazione predefinita della proprietà *proprietà* del grob indicato da *elemento*.

A seconda di come è formulato il comando, `\offset` può comportarsi come un `\tweak` o come un `\override`. Le variazioni d'uso sono trattate dopo aver considerato le proprietà del grob che possono essere usate con `\offset`.

#### *Proprietà che possono essere spostate con offset*

Molte, ma non tutte, le proprietà dei grob possono essere spostate con offset. Se la *proprietà* non può essere spostata con offset, l'oggetto resterà invariato e apparirà un avviso. In questi casi bisogna usare `\override` o `\tweak` per modificare l'oggetto.

Si potrebbe procedere per prova e errore e basarsi sugli avvisi per capire cosa può essere spostato con offset e cosa no. Tuttavia si può usare un approccio più sistematico.

I seguenti criteri determinano se una proprietà possa essere modificata con `\offset`:

- La proprietà ha un'"impostazione predefinita' nella descrizione del grob. Queste proprietà sono elencate, per ogni grob, in Sezione "All layout objects" in *Guida al Funzionamento Interno* (si trovano anche in `scm/define-grobs.scm`).
- La proprietà accetta un valore numerico. I valori numerici comprendono `number`, una lista di `number`, `number-pair` e `number-pair-list`. Le pagine in Sezione "All layout objects" in *Guida al Funzionamento Interno* elencano il tipo di dati tipici di ciascuna proprietà. È irrilevante che l'impostazione predefinita sia una funzione.
- La proprietà non può essere una 'sottoproprietà', ovvero una proprietà che risiede all'interno di un'altra proprietà.
- Le proprietà impostate su valori infiniti non possono essere spostate con offset, perché non esiste un modo sensato per spostare un infinito positivo o negativo.

Gli esempi seguenti prendono in considerazione varie proprietà grob in relazione ai criteri appena delineati.

- Proprietà che possono essere spostate con offset

  `Hairpin.height`
  : Questa proprietà non è una sottoproprietà e è elencata in Sezione "Hairpin" in *Guida al Funzionamento Interno*. Come valore prende 'dimensione, in spazi rigo' impostata su `0.6666`, ovviamente non un numero infinito.

  `Arpeggio.positions`
  : La pagina Sezione "Arpeggio" in *Guida al Funzionamento Interno* elenca una proprietà `positions` che accetta una 'coppia di numeri'. Il suo valore predefinito è `ly:arpeggio::positions`, una funzione di callback che verrà elaborata

durante la fase di formattazione tipografica per emettere una coppia di numeri per ogni oggetto `Arpeggio`.

- Proprietà che non possono essere spostate con offset

  `Hairpin.color`
  : `color` non è presente in Sezione "Hairpin" in *Guida al Funzionamento Interno*.

  `Hairpin.circled-tip`
  : L'elenco per `Hairpin.circled-tip` in Sezione "Hairpin" in *Guida al Funzionamento Interno* mostra che prende un valore `booleano`. I booleani sono non numerici.

  `Stem.details.lengths`
  : Benché sia elencato in Sezione "Stem" in *Guida al Funzionamento Interno* e il suo valore predefinito sia un elenco di `number`, si tratta di una 'sottoproprietà'. Attualmente non sono supportate le 'proprietà annidate'.

### *offset come override*

Se *elemento* è un nome di grob come `Arpeggio` o `Staff.OttavaBracket`, il risultato è un `\override` del tipo di grob specificato.

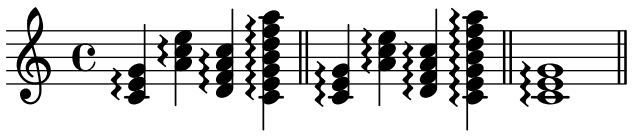
`\offset` *proprietà offset* `[`*contesto*`.]`*NomeGrob*

Notare che il trattino iniziale non è *mai* usato nella forma 'override', proprio come non è mai usato col comando `\override` stesso.

L'esempio seguente usa la forma 'override' per allungare gli arpeggi predefiniti mostrati nella prima misura in modo che coprano del tutto l'esensione degli accordi. Gli arpeggi sono allungati di mezzo spazio rigo verso l'alto e verso il basso. Viene mostrata anche la stessa operazione fatta sul primo accordo con un normale override della proprietà `positions`. Questo metodo non rispecchia affatto il compito di 'allungare di mezzo spazio rigo', perché le estremità devono essere specificate con coordinate assolute invece che relative. Inoltre, sarebbero necessari override individuali per ciascuno degli altri accordi, dato che variano in dimensione e posizione.

```
arpeggioMusic = {
  <c' e' g'>\arpeggio <a' c'' e''>\arpeggio
  <d' f' a' c''>\arpeggio <c' e' g' b' d'' f'' a''>\arpeggio
}

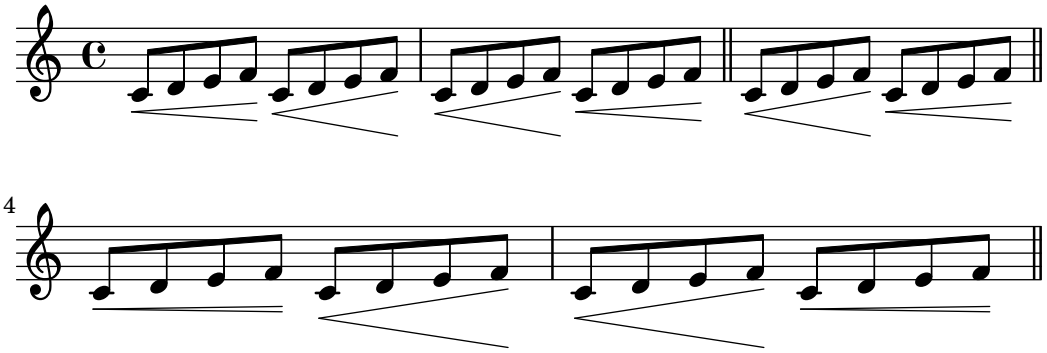
{
  \arpeggioMusic
  \bar "||"
  \offset positions #'(-0.5 . 0.5) Arpeggio
  \arpeggioMusic
  \bar "||"
  \once \override Arpeggio.positions = #'(-3.5 . -0.5)
  <c' e' g'>1\arpeggio
  \bar "||"
}
```



Nel suo uso come 'override', `\offset` può essere preceduto da `\once` o `\temporary` e annullato con `\revert` e la *proprietà* (vedi Sezione "Intermediate substitution functions" in *Estendere*). Ciò deriva dal fatto che `\offset` in realtà crea un `\override` della *proprietà*.

```
music = { c'8\< d' e' f'\! }

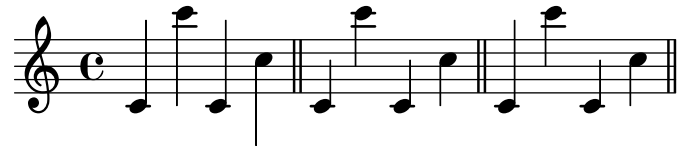
{
  \music
  \offset height 1 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
  \once \offset height 1 Hairpin
  \music \music
  \bar "||"
  \override Hairpin.height = 0.2
  \music
  \temporary \offset height 2 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
}
```



Proprio come `\override`, la forma 'override' di `\offset` può essere usata con `\undo` e `\single`.

```
longStem = \offset length 6 Stem

{
 \longStem c'4 c''' c' c''
 \bar "||"
 \undo \longStem c'4 c''' c' c''
 \bar "||"
 \single \longStem c'4 c''' c' c''
 \bar "||"
}
```



### *offset come tweak*
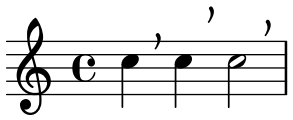
Se *elemento* è un'espressione musicale come `(` o `\arpeggio`, il risultato è la stessa espressione musicale con una modifica di tipo tweak applicata.

`[-]\offset [`*`NomeGrob.`*`]`*`proprietà offset espressione-musicale`*

La sintassi di `\offset` nella sua forma 'tweak' è analoga a quella dello stesso comando `\tweak`, sia nell'ordine sia in presenza o assenza del trattino iniziale.

L'esempio seguente usa la forma 'tweak' per regolare la posizione verticale dell'oggetto `BreathingSign`. Si confronti questo col normale comando `\tweak` che lo segue. La sintassi è equivalente, ma l'output di `\tweak` è meno intuitivo, perché `BreathingSign.Y-offset` è calcolato dalla linea centrale del pentagramma. Al contratio, non è necessatio conoscere come è calcolato `Y-offset` quando si usa `\offset`.

```
{
  c''4
  \breathe
  c''4
  \offset Y-offset 2 \breathe
  c''2
  \tweak Y-offset 3 \breathe
}
```
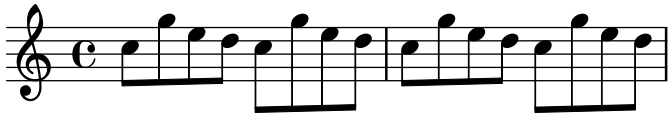


Nell'esempio precedente, gli oggetti modificati sono stati creati direttamente dall'input dell'utente: il comando `\breathe` è un'istruzione esplicita che restituisce un oggetto `BreathingSign`. Dato che l'ambito del comando era non ambiguo, non è stato necessario specificare il nome dell'oggetto. Ma quando un oggetto viene creato *indirettamente*, bisogna includere il nome del grob. Ciò è valido anche per il comando `\tweak`.

Nell'esempio seguente, l'oggetto `Beam` viene abbassato di due spazi rigo applicando `\offset` alla proprietà `positions`.

Il primo impiego di `\offset` richiede la presenza del nome del grob, perché nessun elemento dell'input crea esplicitamente la travatura. Nel secondo impiego la travatura viene creata manualmente con l'espressione musicale `[`; dunque il nome del grob non è necessario. L'esempio mostra anche una scorciatoia: un singolo `numero` viene applicato a entrambi i membri di una coppia di numeri.)

```
{
  c''8 g'' e'' d''
  \offset Beam.positions #'(-2 . -2)
  c''8 g'' e'' d''
  c''8 g'' e'' d''
  c''8-\offset positions #-2 [ g'' e'' d'']
}
```



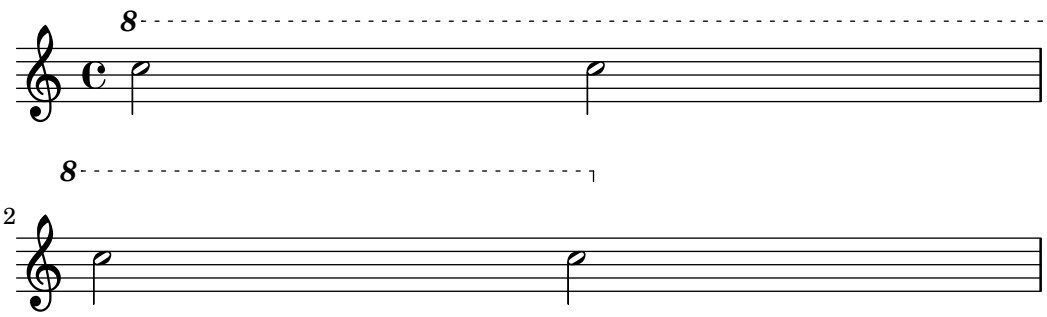
### *offset* con estensori spezzati

È anche possibile modificare in modo indipendente i segmenti di un estensore che va oltre una o più interruzioni di linea. In questo caso, *offset* prende una lista di valori del tipo di dati richiesto dalla proprietà.

Il comando `\offset` usato in questo modo è simile al comando `\alterBroken`, vedi Sezione 5.5.5 [Modifica degli estensori spezzati], pagina 656. Tuttavia, diversamente da `\alterBroken`, i valori assegnati a `\offset` non sono assoluti, ma relativi.

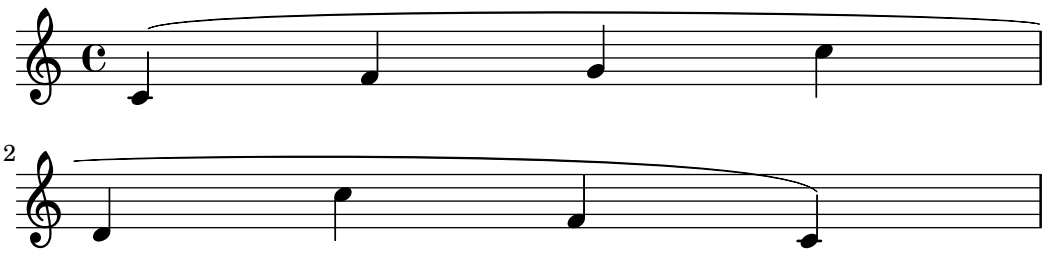
L'esempio seguente sposta l'oggetto 'spezzato' `OttavaBracket` attraverso la sua proprietà `staff-padding`. Dato che la proprietà prende un `numero`, a *offset* viene fornita una lista di `numeri` per rendere conto dei due segmenti creati dall'interruzione di linea. Il pezzo di parentesi dell'ottava sulla prima linea non viene modificato perché viene aggiunto `0` al suo valore predefinito di `staff-padding`. Il segmento sulla seconda linea viene alzato di tre spazi rigo dalla sua altezza predefinita. L'altezza predefinita è `2`, anche se non è necessario saperlo per ottenere la posizione desiderata.

```
{
  \offset staff-padding #'(0 3) Staff.OttavaBracket
  \ottava #1
  c'''2 c'''
  \break
  c'''2 c'''
}
```



L'esempio seguente imita l'effetto del comando `\shape` spostando la proprietà `control-points` dell'oggetto `Slur`. Qui *offset* è una lista di `number-pair-list`, una per ciascun segmento della legatura di portamento. Questo esempio realizza un risultato identico a quello corrispondente in Sezione 5.5.4 [Modifica delle forme], pagina 652.

```
{
  c'4-\offset control-points #'(
              ((0 . 0) (0 . 0) (0 . 0) (0 . 1))
              ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
             ) ( f'4 g' c''
  \break
  d'4 c'' f' c')
}
```



### 5.3.7 Modifica delle liste associative

Alcune proprietà configurabili dall'utente sono rappresentate internamente come *alist* (liste associative), che contengono coppie di *chiavi* e *valori*. La struttura di una lista associativa è:

```
'((chiave1 . valore1)
  (chiave2 . valore2)
  (chiave3 . valore3)
  ...)
```

Se una lista associativa è una proprietà di un grob o una variabile `\paper`, le sue chiavi possono essere modificate individualmente senza influenzare altre chiavi.

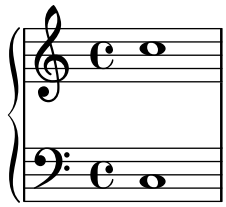
Per esempio, per ridurre lo spazio tra righi adiacenti in un gruppo di righi, si usa la proprietà `staff-staff-spacing` del grob `StaffGrouper`. La proprietà è una lista associativa con quattro chiavi: `basic-distance`, `minimum-distance`, `padding` e `stretchability`. Le impostazioni predefinite per questa proprietà sono elencate nella sezione "Backend" della Guida al funzionamento interno (vedi Sezione "StaffGrouper" in *Guida al Funzionamento Interno*):

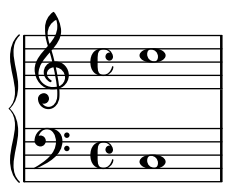
```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

Un modo per avvicinare i righi è ridurre il valore della chiave `basic-distance` (`9`) perché corrisponda al valore di `minimum-distance` (`7`). Per modificare una chiave singola individualmente, usare una *dichiarazione annidata*:

```
% spazio predefinito tra i righi
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>

% spazio ridotto tra i righi
\new PianoStaff \with {
  % questa è la dichiarazione annidata
  \override StaffGrouper.staff-staff-spacing.basic-distance = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```

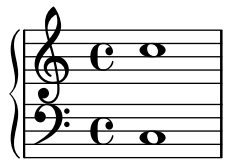




Con una dichiarazione annidata si aggiornerà la chiave specificata (come `basic-distance` nell'esempio precedente) senza modificare alcuna altra chiave già impostata per la stessa proprietà.

Ora immaginiamo di volere che i righi siano più vicini possibile ma senza sovrapporsi. Il modo più semplice per fare ciò è impostare tutte e quattro le chiavi della lista su zero. Tuttavia, non è necessario inserire quattro dichiarazioni annidate, una per ogni chiave. Si può invece ridefinire completamente la proprietà con una sola dichiarazione, attraverso una lista associativa:

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
       (minimum-distance . 0)
       (padding . 0)
       (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



Nota bene che qualsiasi chiave non elencata esplicitamente nella definizione della lista associativa sarà ripristinata al suo valore *predefinito-quando-non-impostato*. Nel caso di `staff-staff-spacing`, qualsiasi chiave-valore non impostata viene ripristinata su zero (eccetto `stretchability`, che prende il valore di `basic-distance` quando non è impostata). Dunque le due seguenti dichiarazioni sono equivalenti:

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
     (minimum-distance . 0)
     (padding . 0)
     (stretchability . 7))
```

Una conseguenza (possibilmente non voluta) di questo approccio è l'eliminazione di eventuali impostazioni predefinite impostate in un file di inizializzazione e caricate ogni volta che un file di input viene compilato. Nell'esempio precedente le impostazioni predefinite per `padding` e `minimum-distance` (definite in `scm/define-grobs.scm`) sono ripristinate ai loro valori predefiniti-quando-non-impostati (zero per entrambe le chiavi). La definizione di una proprietà o di una variabile come una lista associativa (di qualsiasi dimensione) ripristinerà sempre tutte le chiavi-valori ai loro valori predefiniti-quando-non-impostati. A meno che questo non sia il risultato voluto, è più sicuro aggiornare le chiavi-valori individualmente con una dichiarazione annidata.

> **Nota:** Le dichiarazioni annidate non funzionano per le liste associative delle proprietà di contesto (come `beamExceptions`, `keyAlterations`, `timeSignatureSettings`, etc.). Queste proprietà possono essere modificate soltanto ridefinendole completamente come liste associative.

## 5.4 Proprietà e concetti utili

### 5.4.1 Modalità di inserimento

Il modo in cui la notazione contenuta in un file di input è interpretata è determinato dalla modalità di inserimento corrente. In generale, esistono due modi di specificare la modalità: una forma lunga, come `\chordmode`, e una breve, come `\chords`. La forma lunga viene usata tipicamente quando si passa l'input a una variabile o quando si inserisce l'input direttamente in un contesto creato esplicitamente. La forma breve crea implicitamente un contesto del tipo corretto per quell'input e passa l'input a questo direttamente. È utile in casi semplici in cui non è necessario creare esplicitamente il contesto.

#### *Modalità accordo*

Viene attivata col comando `\chordmode` e fa sì che l'input sia interpretato con la sintassi della notazione degli accordi, vedi Sezione 2.7 [Notazione per accordi], pagina 419. La musica in modalità accordo viene elaborata come accordi su un rigo se inserita in un contesto `Staff`, come nomi di accordo se inserita in un contesto `ChordNames` oppure come diagrammi della tastiera se inserita in un contesto `FretBoards`.

La modalità accordo viene attivata anche col comando `\chords`. Anche questo fa sì che l'input che segue sia interpretato con la sintassi della notazione degli accordi, ma in aggiunta crea implicitamente un nuovo contesto `ChordNames` e elabora l'input al suo interno come nomi di accordo, vedi [Stampa dei nomi degli accordi], pagina 424.

#### *Modalità percussioni*

Viene attivata col comando `\drummode` e fa sì che l'input sia interpretato con la sintassi della notazione delle percussioni, vedi [Notazione di base per percussioni], pagina 398. La musica in modalità percussioni è elaborata come note percussive se inserita in un contesto `DrumStaff`.

La modalità percussioni viene attivata anche col comando `\drums`. Anche questo fa sì che l'input che segue sia interpretato con la sintassi della notazione delle percussioni, ma in aggiunta crea implicitamente un nuovo contesto `DrumStaff` e elabora l'input come note percussive, vedi [Notazione di base per percussioni], pagina 398.

#### *Modalità basso continuo*

Viene attivata col comando `\figuremode` e fa sì che l'input che segue sia interpretato con la sintassi del basso continuo, vedi [Inserimento del basso continuo], pagina 433. La musica in modalità basso continuo viene elaborata come basso continuo se inserita in un contesto `FiguredBass` o in un contesto `Staff`.

La modalità basso continuo viene attivata anche col comando `\figures`. Anche questo fa sì che l'input che segue sia interpretato con la sintassi del basso continuo, ma in aggiunta crea implicitamente un nuovo contesto `FiguredBass` e elabora l'input al suo interno come basso figurato, vedi [Introduzione al basso continuo], pagina 433.

#### *Modalità tastiera e intavolatura*

Non esistono speciali modalità di inserimento per i simboli della tastiera e dell'intavolatura (o tablatura).

Per creare diagrammi in intavolatura, inserire note o accordi nella modalità nota e visualizzarli in un contesto `TabStaff`, vedi [Intavolature predefinite], pagina 350.

Per creare diagrammi della tastiera sopra un rigo, inserire le note o gli accordi in modalità nota o in modalità accordo e elaborarli in un contesto `FretBoards`, vedi [Diagrammi dei tasti automatici], pagina 388. Altrimenti i diagrammi possono essere inseriti come testo (markup) sopra le note usando il comando `\fret-diagram`, vedi [Diagrammi dei tasti], pagina 368.

### *Modalità testo vocale*

Viene attivata col comando `\lyricmode` e fa sì che l'input sia interpretato come sillabe del testo vocale con durate opzionali e modificatori del testo associato, vedi Sezione 2.1 [Musica vocale], pagina 267. L'input in modalità testo vocale viene elaborato come sillabe del testo se inserite in un contesto `Lyrics`.

La modalità testo vocale viene attivata anche col comando `\lyrics`. Anche questo fa sì che l'input che segue sia interpretato come sillabe del testo ma in aggiunta crea implicitamente un nuovo contesto `Lyrics` e elabora l'input al suo interno come sillabe.

La modalità testo vocale viene attivata anche col comando `\addlyrics`. Anche questo crea implicitamente un nuovo contesto `Lyrics` e aggiunge anche un comando implicito `\lyricsto` che associa il testo vocale che segue con la musica che lo precede, vedi [Durate automatiche delle sillabe], pagina 271..

### *Modalità testo (markup)*

Viene attivata col comando `\markup` e fa sì che l'input sia interpretato con la sintassi di markup, vedi Sezione A.11 [Comandi per markup], pagina 707.

### *Modalità nota*

Questa è la modalità predefinita e può essere attivata esplicitamente col comando `\notemode`. L'input viene interpretato come altezze, durate, markup, etc. e rappresentato come notazione musicale su un rigo.

Normalmente non è necessario specificare la modalità nota esplicitamente, ma potrebbe essere utile farlo in alcune situazioni, per esempio se si è in modalità testo vocale o accordo o qualsiasi altra modalità e si vuole inserire qualcosa che può essere fatto solo con la sintassi della modalità nota.

## 5.4.2 Direzione e posizionamento

Nella composizione tipografica musicale la direzione e il posizionamento di molti elementi è una questione di gusto. Per esempio, i gambi delle note possono essere rivolti in su o in giù; testi, dinamiche e altri segni espressivi possono essere posti sopra o sotto il rigo; il testo può essere allineato a sinistra, destra o al centro; etc. La maggior parte di queste scelte possono essere lasciate alla decisione di LilyPond, ma in alcuni casi si può voler forzare una direzione o un posizionamento particolari.

### Indicatori di direzione delle articolazioni
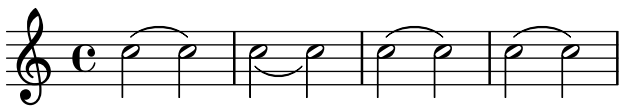
Per impostazione predefinita alcune direzioni sono sempre in su o sempre in giù (es: le dinamiche o la corona), mentre altri elementi possono alternare tra su e giù in base alla direzione del gambo (come le legature di portamento o gli accenti).

L'azione predefinita può essere modificata anteponendo all'articolazione un *indicatore di direzione*. Sono disponbibili tre indicatori di direzione: `^` (ovvero "su"), `_` (ovvero "giù") e `-` (ovvero "usa la direzione predefinita"). L'indicatore di direzione solitamente può essere omesso, nel qual caso viene supposto `-`, ma un indicatore di direzione è **sempre** richiesto prima di

- comandi `\tweak`
- comandi `\markup`
- comandi `\tag`
- stringhe di testo, ovvero `-"stringa"`
- istruzioni di diteggiatura, come `-1`
- scorciatoie delle articolazioni, come `-.`, `->`, `--`

Gli indicatori di direzione hanno effetto soltanto sulla nota vicina:

```
\relative {
  c''2( c)
  c2_( c)
  c2( c)
  c2^( c)
}
```



## La proprietà direction

La posizione della direzione di molti oggetti della formattazione è regolata dalla proprietà `direction`.

Il valore della proprietà `direction` può essere impostata su `1`, ovvero "su" o "sopra", oppure su `-1`, ovvero "giù" o "sotto". I simboli `UP` e `DOWN` possono essere usati al posto di `1` e `-1`. La direzione predefinita può essere specificata impostando `direction` su `0` o `CENTER`. Altrimenti, in molti casi esistono comandi predefiniti per specificare la direzione. Questo hanno la forma
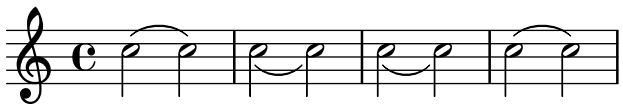
```
\xxxUp, \xxxDown o \xxxNeutral
```

dove `\xxxNeutral` significa "usa la direzione predefinita". Vedi Sezione "Oggetti interni al rigo" in *Manuale di Apprendimento*.

In pochi casi, per esempio l'arpeggio, il valore della proprietà `direction` può specificare se l'oggetto debba essere posizionato a destra o a sinistra dell'oggetto genitore. In questo caso `-1` o `LEFT` significano "a sinistra" e `1` o `RIGHT` significano "a destra". `0` o `CENTER` significano "usa la direzione predefinita".

Queste indicazioni hanno effetto su tutte le note finché non vengono annullate.

```
\relative {
  c''2( c)
  \slurDown
  c2( c)
  c2( c)
  \slurNeutral
  c2( c)
}
```



Nella musica polifonica, generalmente è meglio specificare una voce esplicita invece di cambiare la direzione di un oggetto. Maggiori informazioni in Sezione 1.5.2 [Più voci], pagina 175.

## Vedi anche

Manuale di apprendimento: Sezione "Oggetti interni al rigo" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 1.5.2 [Più voci], pagina 175.

### 5.4.3 Distanze e misurazioni

Le distanze in LilyPond sono di due tipi: assolute e proporzionali.

Le distanze assolute si usano per specificare i margini, le indentazioni e altri dettagli della formattazione di pagina; per impostazione predefinita sono specificate in millimetri. Le distanze possono essere specificate in altre unità di misura appendendo alla quantità `\mm`, `\cm`, `\in` (inch, ovvero pollici) o `\pt` (punti, 1/72.27 di un pollice). Le distanze della formattazione di pagina possono essere specificate anche in unità di misura proporzionali (vedi paragrafo seguente) appendendo `\staff-space` alla quantità. La formattazione di pagina è trattata dettagliatamente in Sezione 4.1 [Formattazione della pagina], pagina 539.

Le distanze proporzionali sono sempre specificate in unità di spazi rigo o, raramente, mezzo spazio rigo. Lo spazio rigo è la distanza tra due linee del rigo adiacenti. Il valore predfinito può essere modificato globalmente impostando la dimensione globale del rigo oppure può essere sovrascritto localmente cambiando la proprietà `staff-space` di `StaffSymbol`. Le distanze proporzionali vengono ridimensionate automaticamente insieme a qualsiasi cambiamento nella dimensione globale del rigo o nella proprietà `staff-space` di `StaffSymbol`, mentre i tipi di carattere vengono ridimensionati automaticamente soltanto con i cambiamenti alla dimensione globale del rigo. Questa permette quindi di variare facilmente la dimensione complessiva di una partitura. I metodi per impostare la dimensione globale del rigo sono descritti in Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

Se soltanto una sezione di una partitura deve essere elaborata su una scala diversa, per esempio una sezione ossia o una nota a piè di pagina, non si può cambiare la dimensione globale del rigo perché ciò avrebbe effetto sull'intera partitura. In questi casi bisogna modificare sia la proprietà `staff-space` di `StaffSymbol` sia la dimensione dei tipi di carattere. Esiste una funzione Scheme, `magstep`, che permette di convertire una modifica della dimensione del tipo di carattere nella modifica equivalente in `staff-space`. Questa funzione è spiegata e esemplificata in Sezione "Lunghezza e spessore degli oggetti" in *Manuale di Apprendimento*.

#### Vedi anche

Manuale di apprendimento: Sezione "Lunghezza e spessore degli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 4.1 [Formattazione della pagina], pagina 539, Sezione 4.2.2 [Impostare la dimensione del rigo], pagina 552.

### 5.4.4 Dimensioni

Le dimensioni di un oggetto grafico specificano le posizioni delle estremità sinistra e destra e di quelle inferiore e superiore del riquadro in cui sono iscritti gli oggetti come distanze dal punto di riferimento degli oggetti in unità di spazi rigo. Queste posizioni sono solitamente scritte come due coppie Scheme. Per esempio, il comando di testo `\with-dimensions` prende tre argomenti: i primi due sono una coppia Scheme che indica le posizioni delle estremità sinistra e destra e un'altra coppia Scheme che indica le posizioni delle estremità inferiore e superiore:

```
\with-dimensions #'(-5 . 10) #'(-3 . 15) arg
```

Questo comando specifica un riquadro per *arg* il cui margine sinistro si trova a -5, il margine destro a 10, il margine inferiore a -3 e quello superiore a 15, tutti misurati a partire dal punto di riferimento degli oggetti in unità di spazi rigo.

#### Vedi anche
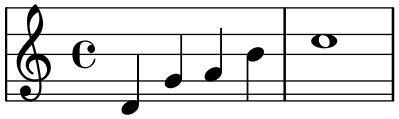
Guida alla notazione: Sezione 5.4.3 [Distanze e misurazioni], pagina 632.

### 5.4.5 Proprietà del simbolo del rigo

La posizione verticale e il numero delle linee del rigo possono essere definiti contemporaneamente. Come mostra l'esempio seguente, le posizioni delle note non sono influenzate dalle posizioni delle linee del rigo.
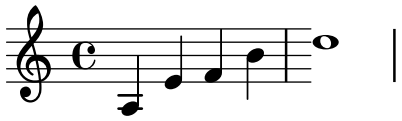
> **Nota:** La proprietà `'line-positions` sovrascrive la proprietà `'line-count`. Il numero di linee del rigo è definito implicitamente dal numero di elementi nella lista di valori per `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol.line-positions = #'(7 3 0 -4 -6 -7)
}
\relative { a4 e' f b | d1 }
```



La larghezza di un rigo può essere modificata. Le unità sono spazi rigo. La spaziatura degli oggetti dentro il rigo non è influenzata da questa impostazione.

```
\new Staff \with {
  \override StaffSymbol.width = #23
}
\relative { a4 e' f b | d1 }
```



### 5.4.6 Estensori

Molti oggetti della notazione musicale si estendono per varie note o addirittura per molte battute. Ne sono un esempio le legature di portamento, le travature, le parentesi dei gruppi irregolari, quelle delle volte delle ripetizioni, i crescendi, i trilli e i glissandi. Tali oggetti in LilyPond si chiamano "spanner", ovvero estensori, e hanno delle proprietà speciali per regolare il loro aspetto e comportamento. Alcune di queste proprietà sono comuni a tutti gli estensori; altre sono limitate a un sottoinsieme di estensori.

Tutti gli estensori supportano l'interfaccia `spanner-interface`. Ma pochi, essenzialmente quelli che disegnano una linea dritta tra due oggetti, supportano anche l'interfaccia `line-spanner-interface`.

#### Uso di `spanner-interface`

Questa interfaccia fornisce due proprietà che si applicano a vari estensori.

#### *La proprietà* `minimum-length`

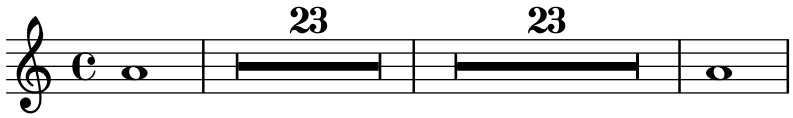
La lunghezza minima dell'estensore è specificata dalla proprietà `minimum-length`. Aumentando questa si aumenta necessariamente anche la spaziatura delle note comprese tra le due estremità. Tuttavia questa modifica non ha effetto su molti estensori, perché la loro lunghezza è determinata da altre considerazioni. Di seguito vengono mostrati alcuni esempi in cui è in funzione.

```
a'~ a'
```
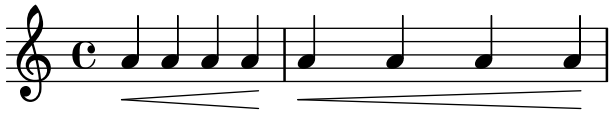
```
a'
% aumenta la lunghezza della legatura di valore
-\tweak minimum-length #5
~ a'
```



```
\relative \compressMMRests {
  a'1
  R1*23
  % aumenta la lunghezza della misura con pausa
  \once \override MultiMeasureRest.minimum-length = #20
  R1*23
  a1
}
```
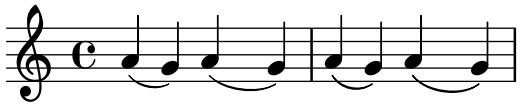



```
\relative {
  a' \< a a a \!
  % aumenta la lunghezza della forcella
  \override Hairpin.minimum-length = #20
  a \< a a a \!
}
```



Questa modifica può essere usata anche per aumentare la lunghezza delle legature di portamento e di frase:

```
\relative {
  a'( g)
  a
  -\tweak minimum-length #5
  ( g)

  a\( g\)
  a
  -\tweak minimum-length #5
  \( g\)
}
```
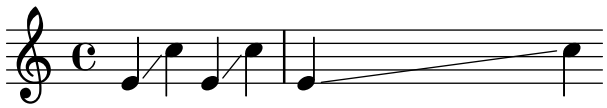


Per alcuni oggetti della formattazione, la proprietà `minimum-length` diventa effettiva solo se viene richiamata esplicitamente la procedura `set-spacing-rods`. Per farlo, la proprietà `springs-and-rods` deve essere impostata su `ly:spanner::set-spacing-rods`. Per esempio, la lunghezza minima di un glissando cambia solo quando si imposta la proprietà `springs-and-rods`:

```
% default
e' \glissando c''

% non funziona da sola
\once \override Glissando.minimum-length = #20
e' \glissando c''
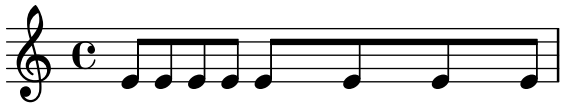
% funziona solo quando entrambe le modifiche sono presenti
\once \override Glissando.minimum-length = #20
\once \override Glissando.springs-and-rods = #ly:spanner::set-spacing-rods
e' \glissando c''
```



Lo stesso vale per l'oggetto `Beam`:

```
% non funziona da sola
\once \override Beam.minimum-length = #20
e'8 e' e' e'
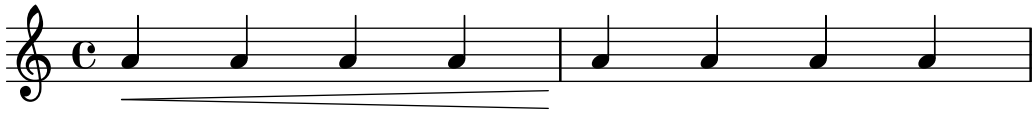
% funziona solo quando entrambe le modifiche sono presenti
\once \override Beam.minimum-length = #20
\once \override Beam.springs-and-rods = #ly:spanner::set-spacing-rods
e'8 e' e' e'
```

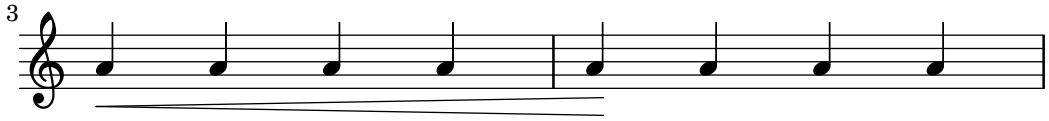


### *La proprietà* `to-barline`

La seconda proprietà utile di `spanner-interface` è `to-barline`. Il valore predefinito (vero) fa sì che le forcelle e gli altri estensori che terminano sulla prima nota di una misura si estendano invece fino alla stanghetta immediatamente precedente. Se impostata su falso, l'estensore andrà oltre la stanghetta e terminerà sulla nota stessa:

```
\relative {
  a' \< a a a a \! a a a \break
  \override Hairpin.to-barline = ##f
  a \< a a a a \! a a a
}
```






Questa proprietà non funziona con tutti gli estensori. Per esempio, impostandola su `#t` non si produce alcun effetto su legature di portamento o di frase, o su altri estensori per i quali terminare sulla stanghetta non avrebbe senso.
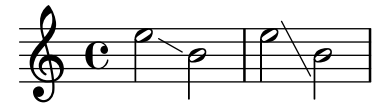
### Uso di `line-spanner-interface`

Gli oggetti che supportano l'interfaccia `line-spanner-interface` comprendono:

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

La routine responsabile del disegno di questi estensori è `ly:line-spanner::print`. Essa determina la posizione esatta delle due estremità e disegna una linea tra di esse, nello stile richiesto. Le posizioni delle due estremità dell'estensore sono calcolate al volo, ma è possibile sovrascrivere le loro coordinate Y. Le proprietà da specificare sono annidate di due livelli nella gerarchia della proprietà, ma la sintassi del comando `\override` è piuttosto semplice:

```
e''2 \glissando b'
\once \override Glissando.bound-details.left.Y = #3
\once \override Glissando.bound-details.right.Y = #-2
e''2 \glissando b'
```

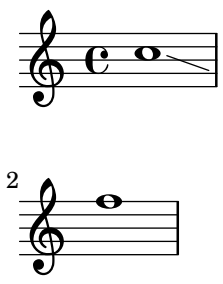


Le unità di misura della proprietà `Y` sono gli spazi rigo (`staff-space`), con la linea centrale del rigo che è il punto zero. Per il glissando, questo è il valore per `Y` alla coordinata X che corrisponde al punto centrale della testa di ogni nota, se si immagina che la linea si estenda fino a là.

Se `Y` non è impostato, il valore viene calcolato dalla posizione verticale del punto di attacco corrispondente dell'estensore.

In caso di un'interruzione di linea, i valori per le estremità sono specificati dalle sottoliste `left-broken` e `right-broken` di `bound-details`. Per esempio:

```
\override Glissando.breakable = ##t
\override Glissando.bound-details.right-broken.Y = #-3
c''1 \glissando \break
f''1
```



Altre proprietà delle sottoliste `left` e `right` della proprietà `bound-details` possono essere modificate nello stesso modo di `Y`:

`Y` Questa imposta la coordinata Y dell'estremità, che si sposta di un certo numero di `staff-space` dalla linea centrale del rigo. Per impostazione predefinita, è il centro dell'oggetto a cui è associato, dunque un glissando punta al centro verticale della testa di nota.

Nel caso di estensori orizzontali, come quelli del testo e del trillo, il suo valore è fisso su 0.
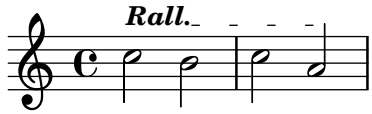
`attach-dir`
Questa determina dove la linea inizia e termina nella direzione X, relativa all'oggetto a cui è associato. Quindi un valore di `-1` (o `LEFT`) fa sì che la linea inizi/termini sul lato sinistro della testa della nota a cui è attaccata.

`X` Questa è la coordinata X assoluta dell'estremità. Viene solitamente calcolata al volo e sovrascriverla produce solitamente un effetto poco utile.

`stencil` Gli estensori della linea possono avere dei simboli all'inizio o alla fine, contenuti in questa sottoproprietà. Questa proprietà è per uso interno; si raccomanda di usare `text` al suo posto.

`text` Questo è un testo markup che viene valutato per produrre lo stampo (stencil, in inglese). Viene usato per mettere *cresc.*, *tr* e altro testo su estensori orizzontali.

```
\override TextSpanner.bound-details.left.text
   = \markup { \small \bold Rall. }
\relative { c''2\startTextSpan b c a\stopTextSpan }
```
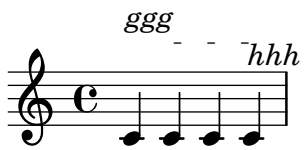



`stencil-align-dir-y`
`stencil-offset`
Se non si imposta una di queste proprietà, lo stampo viene messo semplicemente all'estremità, centrato sulla linea, come definito nelle sottoproprietà `X` e `Y`. Impostando `stencil-align-dir-y` o `stencil-offset` il simbolo si sposterà verticalmente sul margine rispetto all'estremità della linea:

```
\override TextSpanner.bound-details.left.stencil-align-dir-y = #-2
\override TextSpanner.bound-details.right.stencil-align-dir-y = #UP

\override TextSpanner.bound-details.left.text = "ggg"
\override TextSpanner.bound-details.right.text = "hhh"

\relative { c'4^\startTextSpan c c c \stopTextSpan }
```
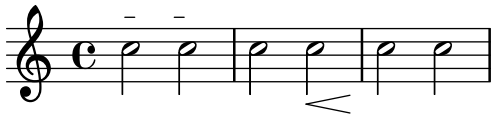



Nota bene che valori negativi spostano il testo in *su*, contrariamente a quanto si potrebbe pensare, perché un valore di `-1` o `DOWN` fa sì che si allinei il margine *inferiore* del testo con la linea dell'estensore. Un valore di `1` o `UP` allinea il margine superiore del testo con la linea dell'estensore.

`arrow` Impostando questa sottoproprietà su `#t` viene generata la punta di una freccia a un'estremità della linea.

`padding` Questa sottoproprietà regola lo spazio tra l'estremità della linea specificata e la fine reale. Senza padding, un glissando inizia e termina nel centro della testa di ogni nota.

La funzione musicale `\endSpanners` termina prematuramente l'estensore che inizia nella nota immediatamente seguente. Viene terminato esattamente dopo una nota o alla stanghetta seguente se `to-barline` è impostato su vero e c'è una stanghetta prima della nota successiva.

```
\relative c'' {
  \endSpanners
  c2 \startTextSpan c2 c2
  \endSpanners
  c2 \< c2 c2
}
```



Quando si usa `\endSpanners` non è necessario chiudere \startTextSpan con \stopTextSpan, né è necessario chiudere le forcelle con `\!`.

### Vedi anche

Guida al funzionamento interno: Sezione "TextSpanner" in *Guida al Funzionamento Interno*, Sezione "Glissando" in *Guida al Funzionamento Interno*, Sezione "VoiceFollower" in *Guida al Funzionamento Interno*, Sezione "TrillSpanner" in *Guida al Funzionamento Interno*, Sezione "line-spanner-interface" in *Guida al Funzionamento Interno*.

## 5.4.7 Visibilità degli oggetti

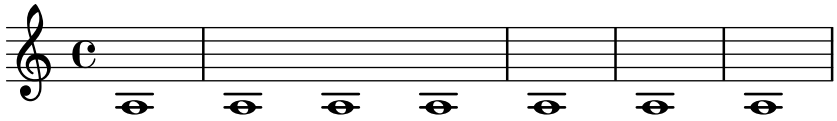
Esistono quattro modi principali per regolare la visibilità degli oggetti della formattazione: si può togliere il loro stampo, possono essere resi trasparenti, possono essere colorati di bianco o si può sovrascrivere la loro proprietà `break-visibility`. Le prime tre sono valide per tutti gli oggetti della formattazione; l'ultimo solo per alcuni, gli oggetti che possono essere *interrotti*, o spezzati, (break, in inglese). Il Manuale di apprendimento contiene un'introduzione a queste quattro tecniche in Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*.

Esistono anche altre tecniche che sono specifiche di alcuni oggetti della formattazione. Queste sono trattate in [Considerazioni speciali], pagina 642.

### Soppressione dello stampo

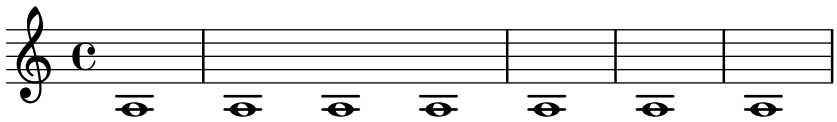
Tutti gli oggetti della formattazione hanno una proprietà `stencil` (in italiano, stampo). Per impostazione predefinita questa è impostata sulla funzione specifica che disegna quell'oggetto. Se questa proprietà viene sovrascritta e impostata su `#f`, non verrà richiamata alcuna funzione e l'oggetto non verrà disegnato. L'azione predefinita può essere ripristinata con `\revert`.

```
a1 a
\override Score.BarLine.stencil = ##f
a a
\revert Score.BarLine.stencil
a a a
```



Questa operazione piuttosto comune ha una scorciatoia, `\omit`:
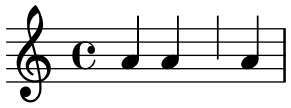
```
a1 a
\omit Score.BarLine
a a
\undo \omit Score.BarLine
a a a
```



## Rendere gli oggetti trasparenti
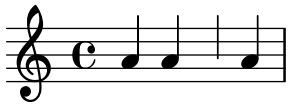
Tutti gli oggetti della formattazione hanno una proprietà `transparent` la cui impostazione predefinita è `#f` (falso). Se impostata su `#t` (vero), l'oggetto occupa sempre lo spazio ma è reso invisibile.

```
a'4 a'
\once \override NoteHead.transparent = ##t
a' a'
```



Questa operazione piuttosto comune ha una scorciatoia, `\hide`:

```
a'4 a'
\once \hide NoteHead
a' a'
```



## Dipingere gli oggetti di bianco

Tutti gli oggetti della formattazione hanno una proprietà del colore il cui valore predefinito è `black` (nero). Se questa viene sovrascritta con `white` (bianco) l'oggetto non sarà più distinguibile dallo sfondo. Tuttavia se l'oggetto attraversa altri oggetti, il colore dei punti di contatto saranno determinati dall'ordine in cui sono disegnati, e ciò potrebbe lasciare un'immagine "spettrale" dell'oggetto bianco, come mostrato qui:

```
\override Staff.Clef.color = #white
a'1
```



Ciò può essere evitato cambiando l'ordine di stampa degli oggetti. Tutti gli oggetti della formattazione hanno una proprietà `layer` (strato) che deve essere impostata su un numero intero. Gli oggetti col valore più basso di `layer` sono disegnati per primi, poi vengono disegnati quelli con valori via via più alti, dunque gli oggetti con valori più alti stanno sopra gli oggetti con

valori più bassi. Per impostazione predefinita alla maggior parte degli oggetti viene assegnato un valore di `layer` pari a `1`, sebbene ad alcuni oggetti, tra cui `StaffSymbol` e `BarLine`, sia assegnato un valore di `0`. L'ordine con cui vengono stampati oggetti con lo stesso valore di `layer` è indeterminato.

Nell'esempio precedente la chiave bianca, con un valore predefinito di `layer` di `1`, viene disegnato dopo le linee del rigo (il cui valore predefinito di `layer` è `0`), dunque sta sopra di esse. Per cambiare tale comportamento, all'oggetto `Clef` deve essere assegnato un valore di `layer` inferiore, per esempio `-1`, in modo che venga disegnato prima:

```
\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1
a'1
```



## Uso di break-visibility

La maggior parte degli oggetti della formattazione sono stampati una volta sola, ma alcuni come le stanghette, le chiavi, le indicazioni di tempo e le armature di chiave possono dover essere stampate due volte quando si verifica un'interruzione di linea – una volta al termine della linea e di nuovo all'inizio della linea successiva. Tali oggetti, chiamati *spezzabili*, (dall'inglese breakable) hanno una proprietà, `break-visibility`, per regolare la loro visibilità nelle tre posizioni in cui potrebbero apparire: all'inizio di una linea, in mezzo a una linea se vengono modificati e al termine di una linea se un cambiamento ha luogo lì.

Per esempio, l'indicazione di tempo viene stampata all'inizio della prima linea, ma da nessuna altra parte a meno che non cambi e allora verrà stampata nel punto in cui il cambiamento ha luogo. Se tale cambiamento si verifica al termine di una linea, la nuova indicazione di tempo verrà stampata all'inizio della linea successiva e un'indicazione di tempo di precauzione verrà stampata anche al termine della linea precedente.

Tale comportamento è regolato dalla proprietà `break-visibility`, che è spiegata in Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*. Questa proprietà prende un vettore di tre valori booleani che, in ordine, determinano se l'oggetto è stampato al termine, in mezzo o all'inizio di una linea. O, per essere più precisi, prima di un'interruzione di linea, dove non c'è un'interruzione di linea oppure dopo un'interruzione di linea.

Altrimenti, queste otto combinazoni possono essere specificate da funzioni predefinite, definite in `scm/output-lib.scm`, dove le ultime tre colonne indicano se gli oggetti della formattazione saranno visibili nelle posizioni mostrate nell'intestazione delle colonne:

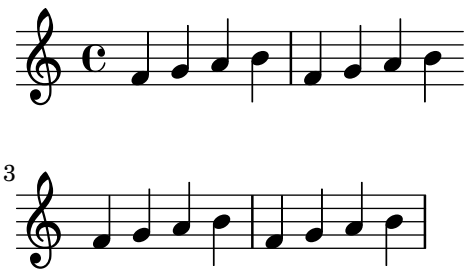
| **Funzione** | **Forma vettoriale** | **Prima di interruzione** | **Senza interruzione** | **Dopo interruzione** |
|---|---|---|---|---|
| `all-visible` | `#(#t #t #t)` | sì | sì | sì |
| `begin-of-line-visible` | `#(#f #f #t)` | no | no | sì |
| `center-visible` | `#(#f #t #f)` | no | sì | no |
| `end-of-line-visible` | `#(#t #f #f)` | sì | no | no |
| `begin-of-line-invisible` | `#(#t #t #f)` | sì | sì | no |
| `center-invisible` | `#(#t #f #t)` | sì | no | sì |
| `end-of-line-invisible` | `#(#f #t #t)` | no | sì | sì |
| `all-invisible` | `#(#f #f #f)` | no | no | no |

Le impostazioni predefinite di `break-visibility` dipendono dall'oggetto della formattazione. La tabella seguente mostra tutti gli oggetti rilevanti che sono influenzati da `break-visibility` e l'impostazione predefinita di questa proprietà:

| **Oggetto formattazione** | **Contesto abituale** | **Impostazione predefinita** |
|---|---|---|
| `BarLine` | `Score` | calcolato |
| `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` |
| `KeyCancellation` | `Staff` | `begin-of-line-invisible` |
| `KeySignature` | `Staff` | `begin-of-line-visible` |
| `ClefModifier` | `Staff` | `begin-of-line-visible` |
| `RehearsalMark` | `Score` | `end-of-line-invisible` |
| `TimeSignature` | `Staff` | `all-visible` |

L'esempio successivo mostra l'uso della forma vettoriale per regolare la visibilità delle stanghette:

```
\relative {
  f'4 g a b
  f4 g a b
  % Toglie la stanghetta al termine della linea corrente
  \once \override Score.BarLine.break-visibility = ##(#f #t #t)
  \break
  f4 g a b
  f4 g a b
}
```




Sebbene tutti i componenti del vettore usati per sovrascrivere `break-visibility` debbano essere presenti, non tutti sono funzionanti con qualsiasi oggetto della formattazione, e alcune combinazioni potrebbero perfino dare errore. Esistono le seguenti limitazioni:
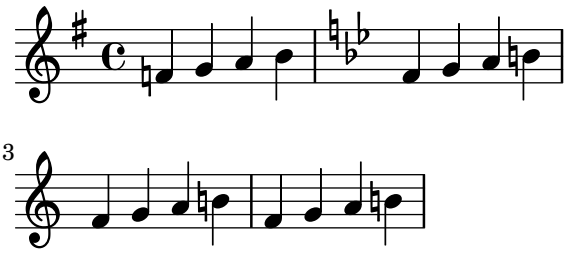
- Non è possibile stampare le stanghette all'inizio di una linea.
- Un numero di battuta non può essere stampato all'inizio della *prima* linea a meno che non sia impostato per essere diverso da `1`.
- Clef – vedi la sezione successiva.
- Le ripetizioni percentuali doppie sono o *tutte stampate* o *tutte soppresse*. Usare `begin-of-line-invisible` per stampare e `all-invisible` per sopprimere.
- Key signature – vedi la sezione successiva.
- ClefModifier – vedi la sezione successiva.

## Considerazioni speciali

### *Visibilità dopo un cambio esplicito*
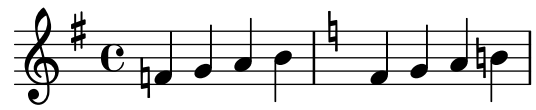
La proprietà `break-visibility` controlla la visibilità delle armature di chiave e dei cambi di chiave solo all'inizio delle linee, ovvero dopo un'interruzione di linea. Non ha effetto sulla visibilità dell'armatura di chiave o della chiave che seguono un cambio esplicito dell'armatura o della chiave all'interno o alla fine di una linea. Nell'esempio seguente l'armatura di chiave che segue il cambio esplicito a Si bemolle maggiore è ancora visibile, anche se `all-invisible` è impostata.
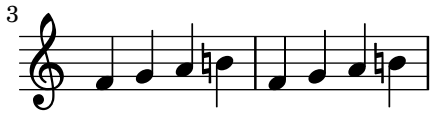
```
\relative {
  \key g \major
  f'4 g a b
  % Tentativo di eliminazione di tutte le armature di chiave
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b
  \break
  f4 g a b
  f4 g a b
}
```



La visibilità di questi cambi espliciti di armatura e di chiave è controllata dalle proprietà `explicitKeySignatureVisibility` e `explicitClefVisibility`. Sono equivalenti alla proprietà `break-visibility` e prendono entrambe come argomento un vettore di tre booleani o le funzioni predefinite elencate prima, proprio come `break-visibility`. Entrambe le proprietà appartengono al contesto Staff, non agli oggetti della formattazione stessi, dunque si impostano col comando `\set`. La loro impostazione predefinita è `all-visible`. Queste proprietà regolano solo la visibilità di armature di chiave e chiavi risultanti da cambi espliciti e non quelli all'inizio delle linee; per eliminare questi ultimi bisogna sempre sovrascrivere `break-visibility` nell'oggetto appropriato.

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```
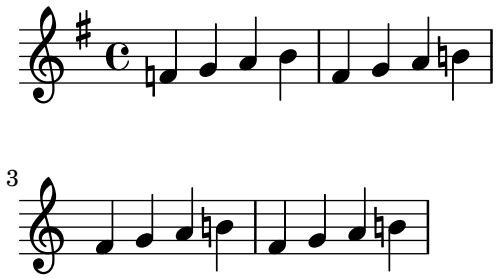




### *Visibilità dei bequadri*

Per eliminare i bequadri stampati su un cambio di chiave esplicito, impostare la proprietà del contesto Staff `printKeyCancellation` su `#f`:
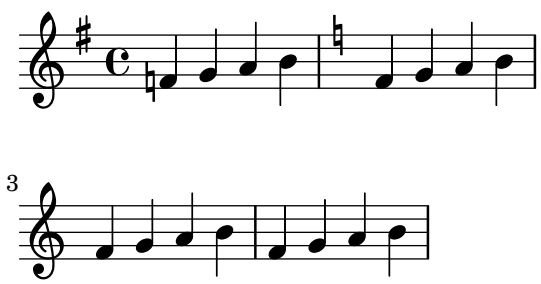
```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Con queste modifiche restano solo le alterazioni accanto alle note per indicare il cambio di armatura.

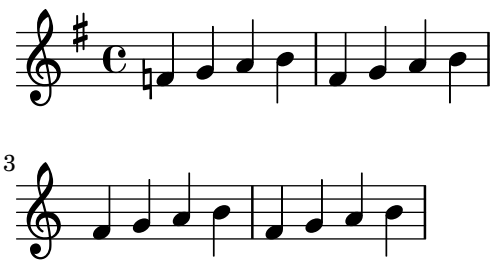
Nota bene che quando si cambia l'armatura su Do maggiore o La minore i bequadri sono l'*unica* indicazione del cambio di chiave. In questo caso impostando `printKeyCancellation` su `#f` non si ottiene alcun effetto:

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Per sopprimere i bequadri anche quando la tonalità passa a Do maggiore o La minore, sovrascrivere la visibilità del grob `KeyCancellation`:

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeyCancellation.break-visibility = #all-invisible
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



### *Battute automatiche*

In un caso particolare, la stampa delle stanghette può essere disattivata impostando la proprietà `automaticBars` nel contesto Score. Se impostata su `#f`, le stanghette non verranno stampate automaticamente; devono invece essere create esplicitamente con un comando `\bar`. Diversamente dal comando predefinito `\cadenzaOn`, le misure vengono comunque contate. La generazione delle battute riprenderà in base a quel conteggio se questa proprietà viene poi impostata su `#t`. Se impostata su `#f`, le interruzioni di linea possono trovarsi solo su comandi `\bar` espliciti.

### *Chiavi trasposte*

Il piccolo simbolo di trasposizione sulle chiavi trasposte è prodotto dall'oggetto di formattazione `ClefModifier`. La sua visibilità è ereditata automaticamente dall'oggetto `Clef`, dunque non è necessario applicare una modifica di `break-visibility` agli oggetti `ClefModifier` per sopprimere i simboli di trasposizione per le chiavi invisibili.

Per cambi di chiave espliciti, la proprietà `explicitClefVisibility` regola sia il simbolo della chiave che qualsiasi simbolo di trasposizione ad esso associato.

### Vedi anche

Manuale di apprendimento: Sezione "Visibilità e colore degli oggetti" in *Manuale di Apprendimento*.

## 5.4.8 Stili della linea
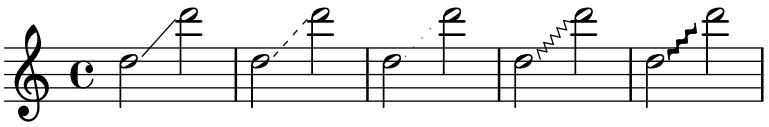
Alcune indicazioni esecutive, come *rallentando*, *accelerando* e *trilli* sono scritte in forma testuale e sono estese per varie misure tramite delle linee, talvolta puntate o ondulate.

Queste usano tutte le stesse routine del glissando per disegnare i testi e le linee, dunque il loro comportamento viene modificato nello stesso modo. Viene fatto con un estensore, e la routine responsabile del disegno degli estensori è `ly:line-spanner::print`. Questa routine determina la posizione esatta dei due *punti di estensione* e disegna una linea tra loro, nello stile richiesto.

Ecco un esempio che mostra i vari stili di linea disponibili, e come impostarli..
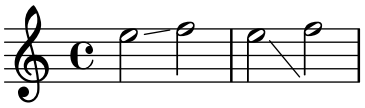
```
\relative {
```

```
  d''2 \glissando d'2
  \once \override Glissando.style = #'dashed-line
  d,2 \glissando d'2
  \override Glissando.style = #'dotted-line
  d,2 \glissando d'2
  \override Glissando.style = #'zigzag
  d,2 \glissando d'2
  \override Glissando.style = #'trill
  d,2 \glissando d'2
}
```



I punti delle estremità dell'estensore sono calcolati al volo per ogni oggetto grafico, ma è possibile sovrascriverli:

```
\relative {
  e''2 \glissando f
  \once \override Glissando.bound-details.right.Y = #-2
  e2 \glissando f
}
```



Il valore di `Y` è impostato su `-2` per l'estremità destra. Quella sinistra può essere aggiustata ugualmente specificando `left` invece di `right`.

Se `Y` non è impostata, il valore è calcolato a partire dalla posizione verticale dei punti di attacco sinistro e destro dell'estensore.

Sono possibili altre modifiche degli estensori; maggiori dettagli in Sezione 5.4.6 [Estensori], pagina 633.

### 5.4.9 Rotazione degli oggetti

Sia gli oggetti della formattazione che il testo compreso in un blocco markup possono essere ruotati di qualsiasi angolo in quasi qualsiasi punto, ma il metodo per farlo cambia.

#### Rotazione degli oggetti della formattazione

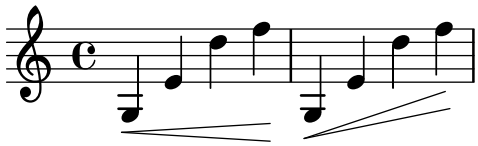
Tutti gli oggetti della formattazione che supportano l'interfaccia `grob-interface` possono essere ruotati impostando la loro proprietà `rotation`. Questa prende come argomento una lista di tre elementi: l'angolo di rotazione in senso antiorario, e le coordinate x e y del punto relativo al punto di riferimento dell'oggetto intorno al quale si deve eseguire la rotazione. L'angolo di rotazione è specificato in gradi e le coordinate in spazi rigo.

L'angolo di rotazione e le coordinate del punto di rotazione devono essere determinate per prova e errore.

Esistono solo poche situazioni in cui la rotazione degli oggetti della formattazione è utile; l'esempio seguente ne mostra uno:

```
g4\< e' d'' f''\!
\override Hairpin.rotation = #'(15 -1 0)
```
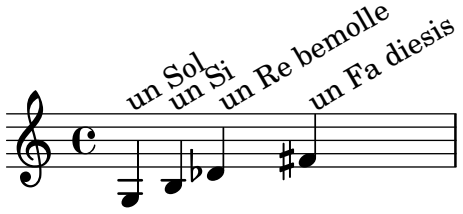
```
g4\< e' d'' f''\!
```



### Rotazione del testo

Tutto il testo in un blocco `\markup` può essere ruotato di qualsiasi angolo facendolo precedere dal comando `\rotate`. Il comando prende due argomenti: l'angolo di rotazione in gradi in senso antiorario e il testo da ruotare. Le estensioni del testo non sono ruotate: prendono il loro valore dagli estremi delle coordinate x e y del testo ruotato. Nell'esempio seguente la proprietà `outside-staff-priority` del testo è impostata su `#f` per disabilitare l'elusione automatica delle collisioni, che spingerebbe parte del testo troppo in alto.

```
\override TextScript.outside-staff-priority = ##f
g4^\markup { \rotate #30 "un Sol" }
b^\markup { \rotate #30 "un Si" }
des'^\markup { \rotate #30 "un Re bemolle" }
fis'^\markup { \rotate #30 "un Fa diesis" }
```




## 5.5 Ritocchi avanzati

Questa sezione discute i vari approcci che permettono di migliorare l'aspetto delle partiture.

### Vedi anche

Manuale di apprendimento: Sezione "Modifica dell'output" in *Manuale di Apprendimento*, Sezione "Altre fonti di informazione" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.2 [Come funziona la Guida al funzionamento interno], pagina 612, Sezione 5.3 [Modifica delle proprietà], pagina 615.

Estendere LilyPond: Sezione "Interfaces for programmers" in *Estendere*.

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione "Tweaks and overrides" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "All layout objects" in *Guida al Funzionamento Interno*.

### 5.5.1 Allineamento degli oggetti

Gli oggetti grafici che supportano l'interfaccia `self-alignment-interface` e/o l'interfaccia `side-position-interface` possono essere allineati a un oggetto disposto precedentemente in vari modi. L'elenco di questi oggetti è consultabile in Sezione "self-alignment-interface" in *Guida al Funzionamento Interno* e Sezione "side-position-interface" in *Guida al Funzionamento Interno*.

Tutti gli oggetti grafici hanno un punto di riferimento, un'estensione orizzontale e una verticale. L'estensione orizzontale è una coppia di numeri che determina gli spostamenti dal punto di

riferimento dei margini sinistro e destro (gli spostamenti a sinistra sono negativi). L'estensione verticale è una coppia di numeri che determina lo spostamento dal punto di riferimento ai margini inferiore e superiore (gli spostamenti in giù sono negativi).

La posizione di un oggetto su un rigo è dato dai valori delle proprietà `X-offset` e `Y-offset`. Il valore di `X-offset` dà lo spostamento dalla coordinata X del punto di riferimento dell'oggetto genitore, mentre il valore di `Y-offset` dà lo spostamento dalla linea centrale del rigo. I valori di `X-offset` e `Y-offset` possono essere impostati direttamente oppure possono essere impostati per essere calcolati dalle procedure in modo da consentire un allineamento con l'oggetto genitore.

> **Nota:** Molti oggetti hanno speciali questioni di posizionamento che fanno sì che qualsiasi impostazione di `X-offset` o `Y-offset` sia ignorata o modificata, anche se l'oggetto supporta l'interfaccia `self-alignment-interface`. Sovrascrivendo le proprietà `X-offset` o `Y-offset` su un valore fisso la rispettiva proprietà `self-alignment` verrà ignorata.

Per esempio, un'alterazione può essere riposizionata verticalmente impostando `Y-offset` ma qualsiasi modifica di `X-offset` non avrà effetto.

I segni di chiamata possono essere allineati con oggetti spezzabili come stanghette, simboli di chiave, simboli di indicazione di tempo e armature di chiave. Ci sono proprietà speciali elencate nell'interfaccia `break-aligned-interface` per posizionare i segni di chiamata su tali oggetti.

## Vedi anche
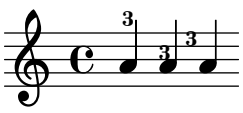
Guida alla notazione: [Uso di break-alignable-interface], pagina 649.

Estendere LilyPond: Sezione "Callback functions" in *Estendere*.

## Impostazione diretta di `X-offset` e `Y-offset`

Si possono assegnare valori numerici alle prietà `X-offset` e `Y-offset` di molti oggetti. L'esempio seguente mostra tre note con la posizione predefinita della diteggiatura e le posizioni con `X-offset` e `Y-offset` modificate.

```
a'-3
a'
-\tweak X-offset #0
-\tweak Y-offset #0
-3
a'
-\tweak X-offset #-1
-\tweak Y-offset #1
-3
```



## Uso di `side-position-interface`

Un oggetto che supporta l'interfaccia `side-position-interface` può essere posizionato accanto al suo oggetto genitore in modo che i margini indicati dei due oggetti si tocchino. L'oggetto può essere posizionato sopra, sotto, a destra o a sinistra dell'oggetto genitore. Quest'ultimo non può essere specificato; è invece determinato dall'ordine degli elementi nel flusso di input. La maggior parte degli oggetti ha la testa di nota associata come oggetto genitore.

I valori delle proprietà `side-axis` e `direction` determinano dove l'oggetto deve essere posizionato, in questo modo:

| `side-axis` **proprietà** | `direction` **proprietà** | **Posizionamento** |
|---|---|---|
| `0` | `-1` | sinistra |
| `0` | `1` | destra |
| `1` | `-1` | sotto |
| `1` | `1` | sopra |

Quando `side-axis` è `0`, `X-offset` deve essere impostato sulla procedura `ly:side-position-interface::x-aligned-side`. Questa procedura restituirà il valore corretto di `X-offset` per porre l'oggetto al lato sinistro o destro dell'oggetto genitore in base al valore di `direction`.

Quando `side-axis` è `1`, `Y-offset` deve essere impostato sulla procedura `ly:side-position-interface::y-aligned-side`. Questa procedura restituirà il valore corretto di `Y-offset` per porre l'oggetto sopra o sotto l'oggetto genitore in base al valore di `direction`.

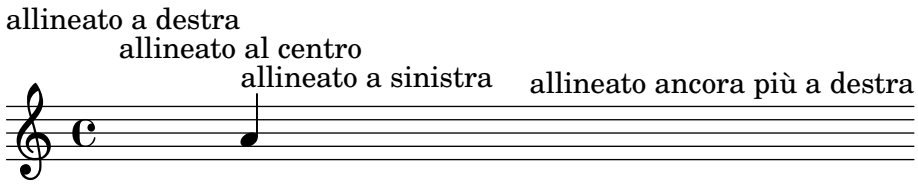
## Uso di `self-alignment-interface`

### *Autoallineamento orizzontale degli oggetti*

L'allineamento orizzontale di un oggetto che supporta l'interfaccia `self-alignment-interface` è regolato dal valore della proprietà `self-alignment-X`, purché la proprietà `X-offset` dell'oggetto sia impostata su `ly:self-alignment-interface::x-aligned-on-self`. A `self-alignment-X` può essere assegnato qualsiasi numero reale, in unità di metà del totale dell'estensione X dell'oggetto. Valori negativi spostano l'oggetto a destra, valori positivi lo spostano a sinistra. Un valore di `0` centra l'oggetto sul punto di riferimento del suo oggetto genitore, un valore di `-1` allinea il margine sinistro dell'oggetto al punto di riferimento del suo genitore, e un valore di `1` allinea il margine destro dell'oggetto al punto di riferimento del suo genitore. I simboli `LEFT`, `CENTER` e `RIGHT` possono essere usati al posto dei valori `-1`, `0` e `1`, rispettivamente.

Normalmente si usa il comando `\override` per modificare il valore di `self-alignment-X`, ma il comando `\tweak` può essere utile per allineare separatamente varie annotazioni su una singola nota:

```
a'
-\tweak self-alignment-X #-1
^"allineato a sinistra"
-\tweak self-alignment-X #0
^"allineato al centro"
-\tweak self-alignment-X #RIGHT
^"allineato a destra"
-\tweak self-alignment-X #-2.5
^"allineato ancora più a destra"
```



### *Autoallineamento verticale degli oggetti*
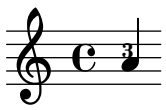
Gli oggetti possono essere allineati verticalmente in un modo analogo al loro allineamento orizzontale se la proprietà `Y-offset` è impostata su `ly:self-alignment-interface::y-aligned-on-self`. Tuttavia altri meccanismi sono spesso implicati nell'allineamento verticale: il valore di `Y-offset` è solo una delle variabili da prendere in considerazione. Ciò potrebbe rendere complicato l'aggiustamento del valore di alcuni oggetti. Le unità sono soltanto la metà dell'estensione verticale dell'oggetto, che è di solito piuttosto piccola, dunque potrebbero essere necessari grossi numeri. Un valore di `-1` allinea il margine inferiore dell'oggetto al punto di riferimento dell'oggetto genitore, un valore di `0` allinea il centro dell'oggetto al punto di riferimento del genitore, e un valore di `1` allinea il margine superiore dell'oggetto al punto di riferimento del genitore. I simboli `DOWN`, `CENTER` e `UP` possono essere usati al posto di `-1`, `0` e `1`, rispettivamente.

### *Autoallineamento degli oggetti in entrambe le direzioni*

Impostando sia `X-offset` che `Y-offset`, un oggetto può essere allineato in entrambe le direzioni simultaneamente.

L'esempio seguente mostra come aggiustare un segno di diteggiatura in modo che stia vicino alla testa di nota.

```
a'
-\tweak self-alignment-X #0.5  % sposta orizzontalmente a sinistra
-\tweak Y-offset #ly:self-alignment-interface::y-aligned-on-self
-\tweak self-alignment-Y #-1  % sposta verticalmente in su
-3  % terzo dito
```
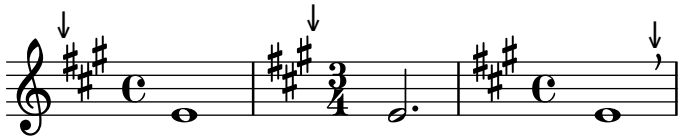


## Uso di `break-alignable-interface`

I segni di chiamata e i numeri di battuta possono essere allineati con oggetti della notazione diversi dalle stanghette. Questi oggetti comprendono `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature` e `time-signature`.

Ogni tipo di oggetto ha il suo punto di riferimento predefinito, al quale i segni di chiamata sono allineati:

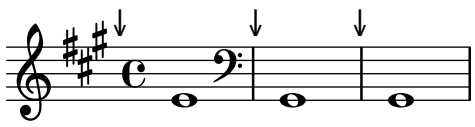
```
% Il segno di chiamata sarà allineato al margine destro della chiave
\override Score.RehearsalMark.break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark "↓"
e'1
% Il segno di chiamata sarà allineato al margine sinistro dell'indicazione di tempo
\override Score.RehearsalMark.break-align-symbols = #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark "↓"
e'2.
% Il segno di chiamata sarà centrato sul segno di respiro
\override Score.RehearsalMark.break-align-symbols = #'(breathing-sign)
\key a \major
```

```
\clef treble
\time 4/4
e'1
\breathe
\mark "↓"
```



Si può specificare un elenco dei possibili oggetti a cui allineare. Se alcuni oggetti sono invisibili in quel punto a causa dell'impostazione di `break-visibility` o di impostazioni esplicite della visibilità di tonalità e chiavi, il segno di chiamata o il numero di battuta viene allineato al primo oggetto dell'elenco che è visibile. Se nessun oggetto dell'elenco è visibile, l'oggetto è allineato alla stanghetta. Se la stanghetta è invisibile, l'oggetto viene allineato al punto in cui si dovrebbe trovare la stanghetta.

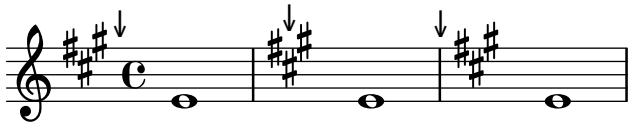
```
% Il segno di chiamata sarà allineato al margine destro dell'armatura di chiave
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
% Il segno di chiamata sarà allineato al margine destro della chiave
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef bass
\mark "↓"
gis,1
% Il segno di chiamata sarà centrato sopra la stanghetta
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
```



L'allineamento del segno di chiamata relativo all'oggetto della notazione può essere modificato, come evidenziato nell'esempio seguente. In una partitura con molteplici righi, questa impostazione deve essere fatta per tutti i righi.

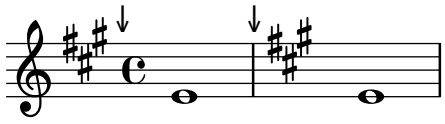
```
% Il segno di chiamata sarà allineato al margine destro dell'armatura di chiave
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
```

```
\mark "↓"
e'1
% Il segno di chiamata sarà centrato sopra l'armatura di chiave
\once \override Score.KeySignature.break-align-anchor-alignment = #CENTER
\mark "↓"
\key a \major
e'1
% Il segno di chiamata sarà allineato al margine sinistro dell'armatura di chiave
\once \override Score.KeySignature.break-align-anchor-alignment = #LEFT
\key a \major
\mark "↓"
e'1
```



Il segno di chiamata può anche essere spostato a destra o a sinistra del margine sinistro di una quantità arbitraria. Le unità sono gli spazi rigo:

```
% Il segno di chiamata sarà allineato al margine sinistro dell'armatura di chiave
% e poi spostato a destra di 3.5 spazi rigo
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark "↓"
e'1
% Il segno di chiamata sarà allineato al margine sinistro dell'armatura di chiave
% e poi spostato a sinistra di 2 spazi rigo
\once \override Score.KeySignature.break-align-anchor = #-2
\key a \major
\mark "↓"
e'1
```



### 5.5.2 Raggruppamento verticale dei grob

I grob VerticalAlignment e VerticalAxisGroup lavorano insieme. VerticalAxisGroup raggruppa insieme diversi grob come Staff, Lyrics, etc. VerticalAlignment quindi allinea verticalmente i diversi grob raggruppati insieme da VerticalAxisGroup. Di solito è presente un solo VerticalAlignment per partitura ma ogni Staff, Lyrics, etc. ha il suo VerticalAxisGroup.

### 5.5.3 Modifica degli stampi

Tutti gli oggetti della formattazione hanno una proprietà `stencil` che fa parte dell'interfaccia `grob-interface`. Per impostazione predefinita, questa proprietà è solitamente impostata su una funzione specifica per l'oggetto per cui è concepita in modo da creare il simbolo che lo rappresenta nell'output. Per esempio, l'impostazione standard della proprietà `stencil` dell'oggetto `MultiMeasureRest` è `ly:multi-measure-rest::print`.
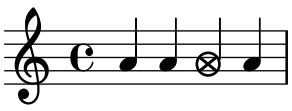
Il simbolo standard di qualsiasi oggetto può essere sostituito modificando la proprietà `stencil` perché faccia riferimento a una procedura diversa e scritta appositamente. Ciò richiede una

profonda conoscenza del funzionamento interno di LilyPond, ma esiste un modo più semplice che spesso è in grado di produrre risultati accettabili.

Si tratta di impostare la proprietà `stencil` sulla procedura che stampa testo – `ly:text-interface::print` – e aggiungere una proprietà `text` all'oggetto impostato per contenere il testo markup che produce il simbolo richiesto. Grazie alla flessibilità di markup, si può ottenere molto – vedi in particolare [Notazione grafica nel blocco markup], pagina 255.

L'esempio seguente illustra questa procedura cambiando il simbolo della testa di nota con una croce iscritta in un cerchio.

```
XinO = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph "noteheads.s2cross"
  }
}
\relative {
  a' a \XinO a a
}
```



Qualsiasi glifo *Feta* usato nel font Emmentaler può essere passato al comando markup `\musicglyph` – vedi Sezione A.8 [Il font Emmentaler], pagina 679.

I file `EPS` e i comandi Postscript possono essere inseriti entrambi nel corpo del testo con i comandi markup `\epsfile` e `\postscript` – vedi Sezione A.11.3 [Graphic], pagina 732.

### Vedi anche

Guida alla notazione: [Notazione grafica nel blocco markup], pagina 255, Sezione 1.8.2 [Formattazione del testo], pagina 246, Sezione A.11 [Comandi per markup], pagina 707, Sezione A.8 [Il font Emmentaler], pagina 679, Sezione A.11.3 [Graphic], pagina 732.

## 5.5.4 Modifica delle forme

### Modifica di legature di valore e di portamento

Le legature, siano esse di valore (`Tie`), di portamento (`Slur`), di frase (`PhrasingSlur`), di laisser-vibrer (`LaissezVibrerTie`) o di valore ripetute (`RepeatTie`), sono tutte disegnate come curve Bézier di terzo ordine. Se la forma della legatura di valore o di frase che viene calcolata automaticamente non è ottimale, può essere modificata manualmente in due modi:

a. specificando di quanto spostare i punti di controllo della curva Bézier calcolata automaticamente, o

b. specificando esplicitamente le posizioni dei quattro punti di controllo richiesti per definire la curva desiderata.

Entrambi i metodi sono spiegati di seguito. Il primo metodo è più adatto se sono necessari solo piccoli aggiustamenti alla curva; il secondo può essere migliore per creare curve che riguardano soltanto una singola nota.
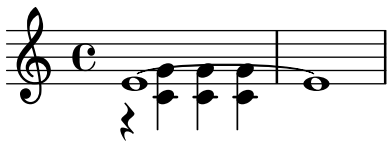
### *Curve Bézier cubiche*

Le curve Bézier di terzo ordine o cubiche sono definite da quattro punti di controllo. Il primo e il quarto punto di controllo sono esattamente i punti di inizio e fine della curva. I due punti intermedi definiscono la forma. Sul web si trovano delle animazioni che mostrano come viene disegnata la curva, ma la seguente descrizione può comunque essere utile. La curva inizia dal primo punto di controllo e si dirige verso il secondo, piegandosi gradualmente per arrivare al terzo e poi al quarto, arrivando lì direttamente dal terzo punto di controllo. La curva è contenuta interamente nel quadrilatero definito dai quattro punti di controllo. Traslazioni, rotazioni e ridimensionamenti dei punti di controllo risultano tutti esattamente nelle stesse operazioni sulla curva.

### *Specificare gli spostamenti dai punti di controllo correnti*

In questo esempio la disposizone automatica della legatura di valore non è ottimale, e `\tieDown` non sarebbe di aiuto.

```
<<
  { e'1~ 1 }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```



Cambiando i punti di controllo della legatura con `\shape` consente di evitare le collisioni.

La sintassi di `\shape` è

`[-]\shape` *`spostamenti elemento`*
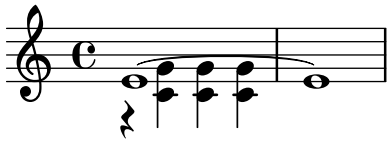
Questo comando riposiziona i punti di controllo di *elemento* di quanto indicato da *spostamenti*. L'argomento *spostamenti* è una lista di coppie di numeri o una lista di tali liste. Ogni elemento di una coppia rappresenta lo spostamento di una delle coordinate di un punto di controllo. Se *elemento* è una stringa, il risultato è `\once\override` per il tipo di grob specificato. Se *elemento* è un'espressione musicale, il risultato è la stessa espressione musicale con un'appropriata modifica.

In altre parole, la funzione `\shape` può comportarsi come un comando `\once\override` o un comando `\tweak` a seconda che l'argomento *elemento* sia il nome di un grob, come "Slur", o un'espressione musicale, come "(". L'argomento *spostamenti* specifica gli sfasamenti dei quattro punti di controllo nella forma di una lista di quattro coppie di valori (dx . dy) in unità di spazi rigo (o una lista di tali liste se la curva ha più di un segmento).

Il trattino iniziale è richiesto se e solo se si usa la forma con `\tweak`.
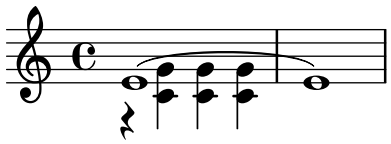
Dunque usando lo stesso esempio precedente e la forma `\once\override` di `\shape`, il seguente comando alzerà la legatura di valore di metà spazio rigo:

```
<<
  {
    \shape #'((0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie
    e'1~ 1
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```
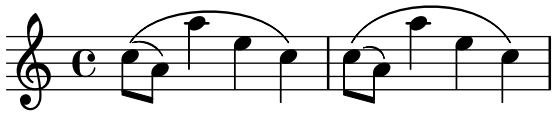


Questo posizionamento è migliore, ma forse dovrebbe essere alzato di più al centro, come viene fatto nel prossimo esempio, dove stavolta si usa la forma alternativa `\tweak`:

```
<<
  {
    e'1-\shape #'((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e'
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```
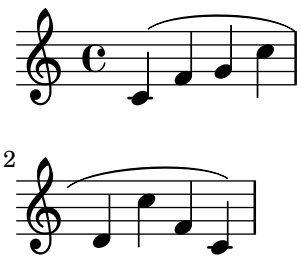


Le modifiche alle posizioni orizzontali dei punti di controllo funzionano nello stesso modo. Si possono ridefinire anche due curve diverse che iniziano nello stesso momento musicale:

```
\relative {
  c''8(\( a) a'4 e c\)
  \shape #'((0.7 . -0.4) (0.5 . -0.4) (0.3 . -0.3) (0 . -0.2)) Slur
  \shape #'((0 . 0) (0 . 0.5) (0 . 0.5) (0 . 0)) PhrasingSlur
  c8(\( a) a'4 e c\)
}
```



La funzione `\shape` può sostituire anche i punti di controllo di curve che si estendono oltre le interruzioni di linea. A ogni pezzo della curva spezzata può essere assegnata la sua lista di spostamenti. Se non sono necessarie delle modifiche a un segmento in particolare, la lista vuota può servire come segnaposto. In questo esempio l'interruzione di linea fa sembrare doppia la legatura di portamento singola:
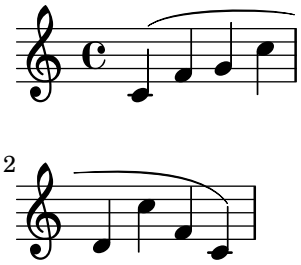
```
\relative {
  c'4( f g c
  \break
  d,4 c' f, c)
}
```




Cambiando le forme delle due metà della legatura di portamento diventa più chiaro che la legatura continua oltre l'interruzione di linea:
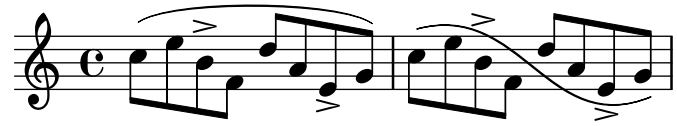
```
% () può essere usato come una scorciatoia per ((0 . 0) (0 . 0) (0 . 0) (0 . 0))
```

```
% nel caso in cui uno dei segmenti non necessiti di modifiche
\relative c' {
  \shape #'(
             (( 0 . 0) (0 . 0) (0 . 0) (0 . 1))
             ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
           ) Slur
  c4( f g c
  \break
  d,4 c' f, c)
}
```



Se è richiesta una curva a forma di S, bisogna sempre definire manualmente i punti di controllo — LilyPond non sceglierà mai tali forme automaticamente.

```
\relative c'' {
  c8( e b-> f d' a e-> g)
  \shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5)) PhrasingSlur
  c8\( e b-> f d' a e-> g\)
}
```



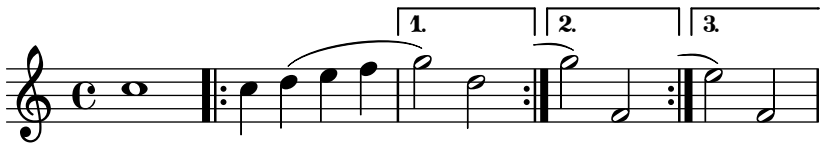
### *Specificare i punti di controllo esplicitamente*

Le coordinate dei punti di controllo della curva di Bézier sono specificate in unità di spazi rigo. La coordinata X è relativa al punto di riferimento della nota a cui la legatura si attacca, mentre la coordinata Y è relativa alla linea centrale del rigo. Le coordinate sono specificate come una lista di quattro coppie di numeri decimali (reali). Un possibile approccio consiste nel valutare prima le coordinate delle due estremità e poi indovinare i due punti intermedi. I valori ottimali vengono quindi trovati per tentativi. Tenere conto del fatto che, in caso di ulteriori modifiche alla musica o alla formattazione, questi valori potrebbero dover essere nuovamente cambiati manualmente.

Una situazione in cui è preferibile specificare i punti di controllo esplicitamente invece di specificarne gli spostamenti è quella in cui devono essere indicati per una singola nota. Il prossimo esempio mostra un modo in cui rappresentare una legatura di portamento che si estende nelle sezioni alternative di una volta.

```
\relative {
  c''1
  \repeat volta 3 { c4 d( e f }
  \alternative {
    { g2) d }
    {
```

```
      g2
      % crea una legatura di portamento e spostala in una nuova posizione
      % il <> è solamente un accordo vuoto che serve a terminare la legatura
      -\tweak control-points #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <> )
      f,
    }
    {
      e'2
      % crea una legatura di portamento e spostala in una nuova posizione
      -\tweak control-points #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <> )
      f,
    }
  }
}
```




## Problemi noti e avvertimenti

Non è possibile modificare le forme di legature di valore e di portamento cambiando la proprietà `control-points` se ci sono molteplici legature nello stesso momento musicale – anche il comando `\tweak` non funzionerà in questo caso. Tuttavia si può sovrascrivere la proprietà `tie-configuration` di `TieColumn` per impostare l'inizio della linea e la direzione come desiderato.

## Vedi anche
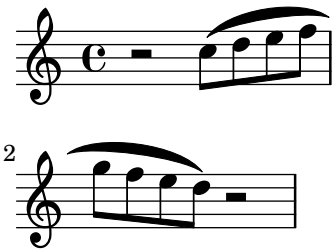
Guida al funzionamento interno: Sezione "TieColumn" in *Guida al Funzionamento Interno*.

### 5.5.5 Modifica degli estensori spezzati

## Uso di `\alterBroken`

Quando un estensore va a capo o si spezza, ciascun pezzo eredita gli attributi dell'estensore originario. Dunque la normale modifica di un estensore spezzato applica le stesse modifiche a ciascuno dei suoi segmenti. Nell'esempio successivo, la sovrascrittura di `thickness` agisce sulla legatura di portamento su entrambi i lati rispetto all'interruzione di linea.

```
\relative c'' {
  r2
  \once\override Slur.thickness = 10
  c8( d e f
  \break
  g8 f e d) r2
}
```



È possibile modificare indipendentemente l'aspetto di singoli pezzi di un estensore spezzato col comando `\alterBroken`. Questo può generare sia un `\override` che un `\tweak` della proprietà di un estensore.

La sintassi di `\alterBroken` è:

`[-]\alterBroken` *`proprietà valori elemento`*
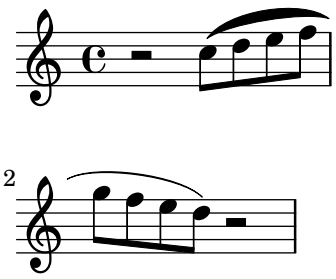
L'argomento *valori* è una lista di valori, uno per ogni pezzo spezzato. Se *elemento* è il nome di un grob come `Slur` o `Staff.PianoPedalBracket`, il risultato è un `\override` del tipo di grob specificato. Se *elemento* è un'espressione musicale come "(" o "[", il risultato è la stessa espressione musicale modificata da un comando `\tweak`.

Il trattino iniziale è richiesto nella forma `\tweak`. Non aggiungerlo se `\alterBroken` viene usato come un `\override`.

Nel suo uso come `\override`, `\alterBroken` può essere preceduto da `\once` o `\temporary` e ripristinato con `\revert` seguito da *proprietà* (vedi Sezione "Intermediate substitution functions" in *Estendere*).

Il codice seguente applica un `\override` indipendente a ciascun segmento della legatura di portamento dell'esempio precedente:
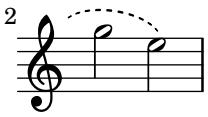
```
\relative c'' {
  r2
  \alterBroken thickness #'(10 1) Slur
  c8( d e f
  \break
  g8 f e d) r2
}
```



Il comando `\alterBroken` può essere usato con qualsiasi oggetto estensore, inclusi `Tie`, `PhrasingSlur`, `Beam` e `TextSpanner`. Per esempio, un editor che stia preparando un'edizione colta potrebbe voler indicare l'assenza di parte di una legatura di frase in un manoscritto tratteggiando solo il segmento che è stato aggiunto. L'esempio seguente mostra come farlo, in questo caso usando la forma `\tweak` del comando:

```
% Si usa per convenienza la lista vuota, perché è l'impostazione
% predefinita di dash-definition, producendo così una curva continua.
\relative {
  c''2-\alterBroken dash-definition #'() ((0 1.0 0.4 0.75))) \(e
  \break
  g2 e\)
}
```






È importante comprendere che `\alterBroken` imposterà ogni pezzo di un estensore spezzato sul valore corrispondente specificato in *valori*. Quando il numero di valori è inferiore a quello dei pezzi, a qualsiasi pezzo ulteriore sarà assegnata una lista vuota. Ciò potrebbe portare a risultati indesiderati se la proprietà di formattazione non è impostata su una lista vuota per impostazione predefinita. In tali casi, a ogni segmento deve essere assegnato un valore appropriato.

### Problemi noti e avvertimenti

Le interruzioni di linea possono verificarsi in punti diversi in seguito a modifiche della formattazione. Le impostazioni scelte per `\alterBroken` potrebbero non essere più adatte per un estensore che non va più a capo o è diviso in più segmenti di prima. Per proteggersi da questa situazione conviene fare un uso esplicito di `\break`.

Il comando `\alterBroken` non funziona nel caso di proprietà dell'estensore a cui si accede prima di line-breaking, come `direction`.

### Vedi anche

Estendere LilyPond: Sezione "Difficult tweaks" in *Estendere*.

## 5.5.6 Contenitori unpure-pure

I contenitori unpure-pure (impuri-puri) sono utili per sovrascrivere i calcoli di spaziatura dell'*asse Y* - in particolare `Y-offset` e `Y-extent` - tramite una funzione Scheme invece che con un numero o una coppia di numeri.

Per alcuni grob, il cui `Y-extent` è basato sulla proprietà `stencil`, sovrascrivere tale proprietà renderà necessario un'ulteriore sovrascrittura di `Y-extent` tramite un contenitore unpure-pure. Quando una funzione sovrascrive `Y-offset` e/o `Y-extent`, si presume che ciò innescherà il calcolo delle interruzioni di linea troppo presto durante la compilazione. Dunque la funzione non viene affatto calcolata (e di solito restituirà un valore di '`0`' o '`'(0 . 0)`') e ciò può causare delle collisioni. Una funzione 'pure' (pura) non avrà effetto su proprietà, oggetti o "suicidi" dei grob e quindi tutto ciò che è collegato al suo asse Y sarà sempre calcolato correttamente.

Attualmente, ci sono circa trenta funzioni che sono già considerate 'pure' e i contenitori unpure-pure sono un modo per impostare le funzioni non ancora presenti in questa lista come 'pure'. La funzione 'pure' viene calcolata *prima* di qualsiasi interruzione di linea e quindi la spaziatura orizzontale può essere regolata "in tempo". La funzione 'unpure' viene invece calcolata *dopo* le interruzioni di linea.

> **Nota:** Dato che è difficile sapere sempre quali funzioni si trovano in questa lista, consigliamo a chi crea funzioni 'pure' di non usare i grob `Beam` o `VerticalAlignment`.

Un contenitore unpure-pure viene costruito nel modo seguente:

`(ly:make-unpure-pure-container f0 f1)`

dove `f0` è una funzione che prende *n* argomenti (*n* `>=` *1*) e il primo argomento deve sempre essere il grob. Questa è la funzione che restituisce il vero risultato. *f1* è la funzione etichettata come 'pure' che prende *n* `+` *2* argomenti. Di nuovo, il primo argomento deve sempre essere il grob, ma il secondo e il terzo sono il punto di partenza, 'start', e quello di arrivo, 'end'.

*start* e *end* sono, per tutti gli intenti e scopi, valori fittizi che contano solo per gli estensori, o `Spanners`, (ovvero `Hairpin` o `Beam`), che possono restituire calcoli diversi dell'altezza in base a una colonna di inizio e di fine.

Il resto sono altri argomenti della prima funzione (che potrebbero essere nessuno se *n* = *1*).
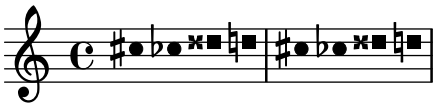
I risultati della seconda funzione sono usati come un'approssimazione del valore necessario che viene poi usato dalla prima funzione per ottenere il valore reale, che viene infine usato per la messa a punto, molto più tardi nel corso del processo di spaziatura.

```
#(define (square-line-circle-space grob)
(let* ((pitch (ly:event-property (ly:grob-property grob 'cause) 'pitch))
       (notename (ly:pitch-notename pitch)))
 (if (= 0 (modulo notename 2))
     (make-circle-stencil 0.5 0.0 #t)
     (make-filled-box-stencil '(0 . 1.0)
                              '(-0.5 . 0.5)))))

squareLineCircleSpace = {
  \override NoteHead.stencil = #square-line-circle-space
}

smartSquareLineCircleSpace = {
  \squareLineCircleSpace
  \override NoteHead.Y-extent =
   #(ly:make-unpure-pure-container
      ly:grob::stencil-height
      (lambda (grob start end) (ly:grob::stencil-height grob)))
}

\new Voice \with { \remove "Stem_engraver" }
\relative c'' {
  \squareLineCircleSpace
  cis4 ces disis d
  \smartSquareLineCircleSpace
  cis4 ces disis d
}
```



Nella prima misura, senza il contenitore unpure-pure, il motore della spaziatura non conosce la larghezza della testa di nota e lascia che collida con le alterazioni. Nella seconda misura, che usa i contenitori unpure-pure, il motore della spaziatura conosce la larghezza della testa di nota e evita le collisioni allungando la linea dello spazio necessario.

Di solito per semplici calcoli si possono usare funzioni quasi identiche per entrambe le parti, 'unpure' e 'pure', cambiando soltanto il numero di argomenti passati e la portata della funzione. Questo caso d'uso è così frequente che `ly:make-unpure-pure-container` costruisce tale seconda funzione automaticamente quando è richiamata con un solo argomento.

> **Nota:** Se una funzione viene etichettata come 'pure' ma viene fuori che non lo è, i risultati possono essere imprevedibili.

## 5.6 Uso delle funzioni musicali

Quando le modifiche devono essere riusate con diverse espressioni musicali, è spesso conveniente fare in modo che la modifica faccia parte di una *funzione musicale*. In questa sezione, trattiamo

soltanto le funzioni di *sostituzione*, dove l'obiettivo è sostituire una variabile in un punto del codice di input di LilyPond. Altre funzioni più complesse sono descritte in Sezione "Music functions" in *Estendere*.

### 5.6.1 Sintassi della funzione di sostituzione

Creare una funzione che sostituisca una variabile presente nel codice LilyPond è semplice. La forma generica di queste funzioni è:

```
funzione =
#(define-music-function
     (arg1 arg2 ...)
     (tipo1? tipo2? ...)
   #{
     ...musica...
   #})
```

dove

| | |
|---|---|
| *argN* | *n*esimo argomento |
| *tipoN?* | un *tipo di predicato* Scheme per il quale *argN* deve ritornare `#t`. |
| *...musica...* | il normale input LilyPond, usando `$` (nei punti in cui sono consentiti solo i costrutti LilyPond) o `#` (per usarlo come un valore Scheme o come argomento di una funzione musicale o come musica all'interno di liste di musica) per riferirsi agli argomenti (es: '`#arg1`'). |

La lista del tipo di predicati è obbligatoria. Alcuni dei tipi di predicati più comuni usati nelle funzioni musicali sono:

```
boolean?
cheap-list?  (usarlo al posto di 'list?' per un'elaborazione più veloce)
ly:duration?
ly:music?
ly:pitch?
markup?
number?
pair?
string?
symbol?
```

Un elenco dei tipi di predicati disponibili si trova in Sezione A.21 [Tipi di predicati predefiniti], pagina 818. Sono consentiti anche tipi di predicati definiti dagli utenti.

### Vedi anche

Guida alla notazione: Sezione A.21 [Tipi di predicati predefiniti], pagina 818.

Estendere LilyPond: Sezione "Music functions" in *Estendere*.

File installati: `lily/music-scheme.cc`, `scm/c++.scm`, `scm/lily.scm`.
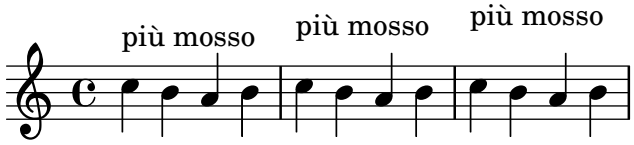
### 5.6.2 Esempi della funzione di sostituzione

Questa sezione presenta alcuni esempi di funzione di sostituzione. L'obiettivo non è fornire un elenco esaustivo, ma mostrare soltanto alcune delle possibilità di semplici funzioni di sostituzione.

Nel primo esempio, viene definita una funzione che semplifica il padding di un oggetto TextScript:

```
padText =
#(define-music-function
     (padding)
     (number?)
   #{
     \once \override TextScript.padding = #padding
   #})
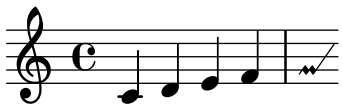
\relative {
  c''4^"più mosso" b a b
  \padText #1.8
  c4^"più mosso" b a b
  \padText #2.6
  c4^"più mosso" b a b
}
```




Oltre ai numeri, si possono usare come argomenti delle funzioni musicali anche le espressioni musicali, come le note:

```
custosNote =
#(define-music-function
     (note)
     (ly:music?)
   #{
     \tweak NoteHead.stencil #ly:text-interface::print
     \tweak NoteHead.text
        \markup \musicglyph "custodes.mensural.u0"
     \tweak Stem.stencil ##f
     #note
   #})

\relative { c'4 d e f \custosNote g }
```
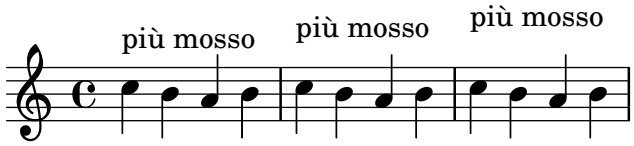


Entrambe queste funzioni sono semplici espressioni singole dove manca soltanto l'ultimo elemento di una chiamata di una funzione o di una sovrascrittura. Per queste particolari definizioni di funzione, esiste una sintassi alternativa più semplice, che consiste nello scrivere interamente la parte costante dell'espressione e sostituire il suo elemento finale mancante con `\etc`:

```
padText =
  \once \override TextScript.padding = \etc
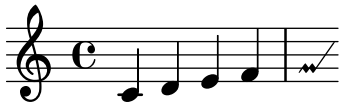
\relative {
  c''4^"più mosso" b a b
  \padText #1.8
  c4^"più mosso" b a b
```

```
  \padText #2.6
  c4^"più mosso" b a b
}
```




```
custosNote =
  \tweak NoteHead.stencil #ly:text-interface::print
  \tweak NoteHead.text
     \markup \musicglyph "custodes.mensural.u0"
  \tweak Stem.stencil ##f
  \etc

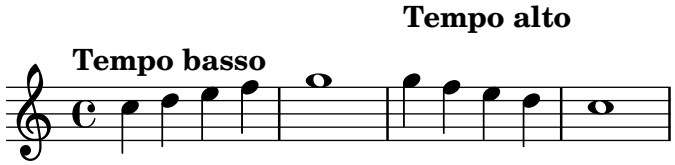
\relative { c'4 d e f \custosNote g }
```



Si possono definire funzioni di sostituzione con molteplici argomenti:

```
tempoPadded =
#(define-music-function
     (padding tempotext)
     (number? markup?)
   #{
     \once \override Score.MetronomeMark.padding = #padding
     \tempo \markup { \bold #tempotext }
   #})

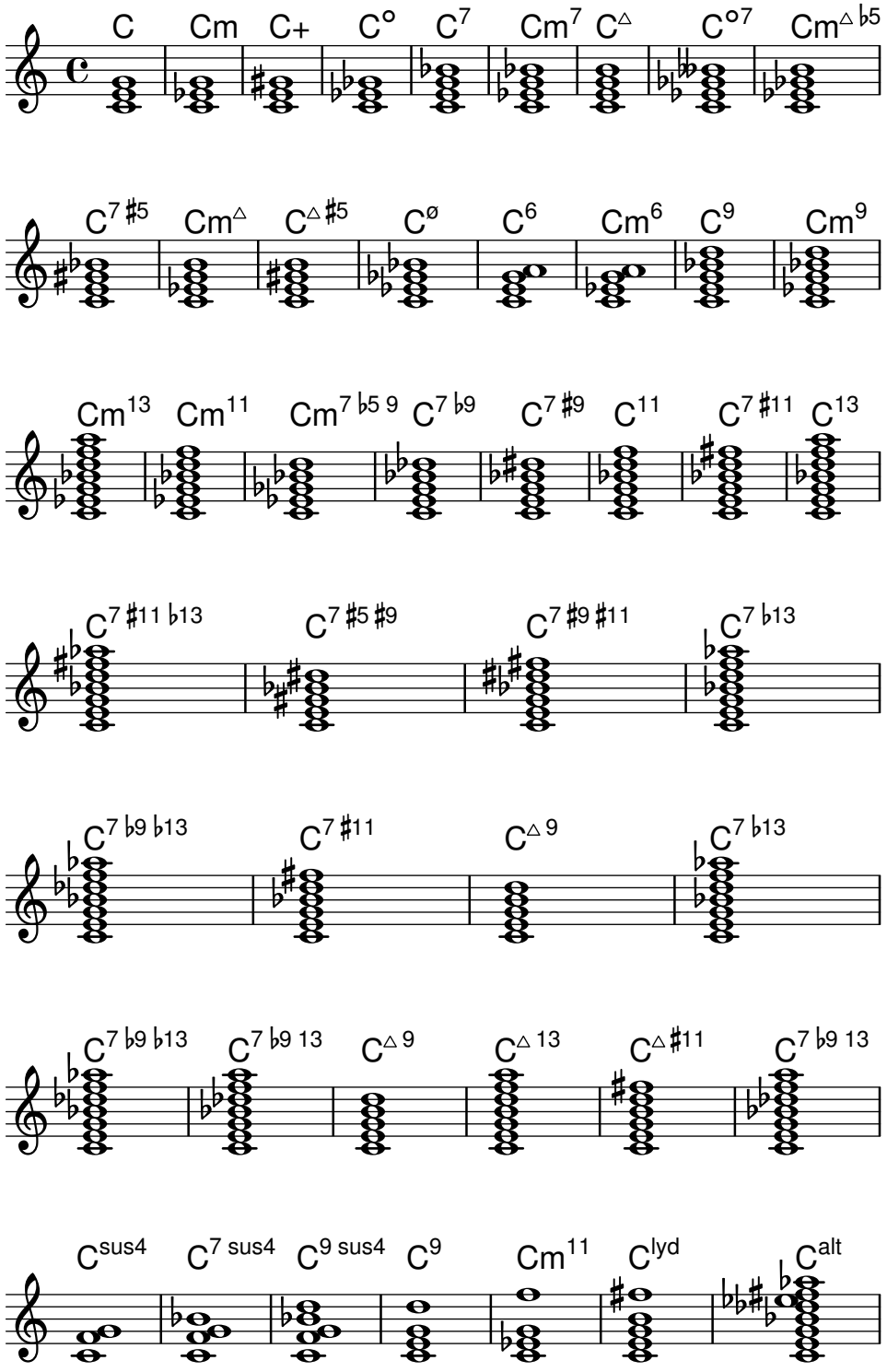
\relative {
  \tempo \markup { "Tempo basso" }
  c''4 d e f g1
  \tempoPadded #4.0 "Tempo alto"
  g4 f e d c1
}
```



# Appendice A Tabelle del manuale della notazione
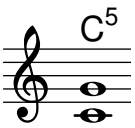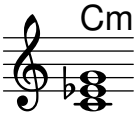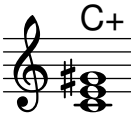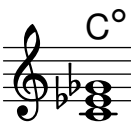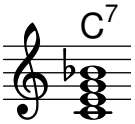
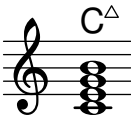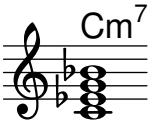
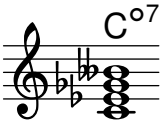
## A.1 Grafico dei nomi degli accordi
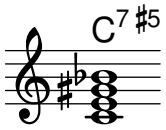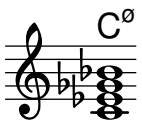
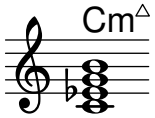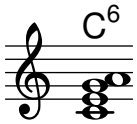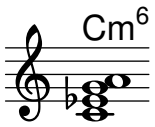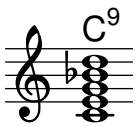
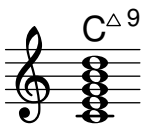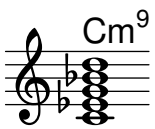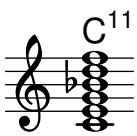
Il grafico seguente mostra due sistemi standard di rappresentazione dei nomi degli accordi, insieme alle altezze che li compongono.

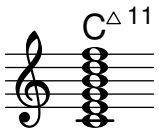
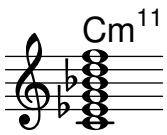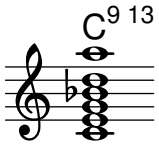
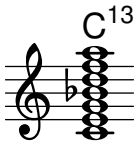
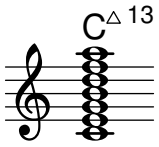
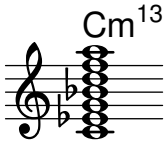
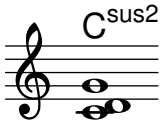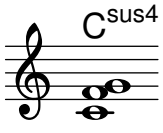


## A.2 Modificatori degli accordi comuni

La tabella seguente mostra i modificatori degli accordi che possono essere usati per generare gli accordi comuni.

| Tipo | Intervallo | Modificatore | Esempio | Output |
|---|---|---|---|---|
| Maggiore | Terza maggiore, quinta perfetta | `5` o niente | `c1:5` |  |
| Minore | Terza minore, quinta perfetta | `m` o `m5` | `c1:m` |  |
| Aumentato | Terza maggiore, quinta aumentata | `aug` | `c1:aug` |  |
| Diminuito | Terza minore, quinta diminuita | `dim` | `c1:dim` |  |
| Settima dominante | Triade maggiore, settima minore | `7` | `c1:7` |  |
| Settima maggiore | Triade maggiore, settima maggiore | `maj7` o `maj` | `c1:maj7` |  |
| Settima minore | Triade minore, settima minore | `m7` | `c1:m7` |  |
| Settima diminuita | Triade diminuita, settima diminuita | `dim7` | `c1:dim7` |  |

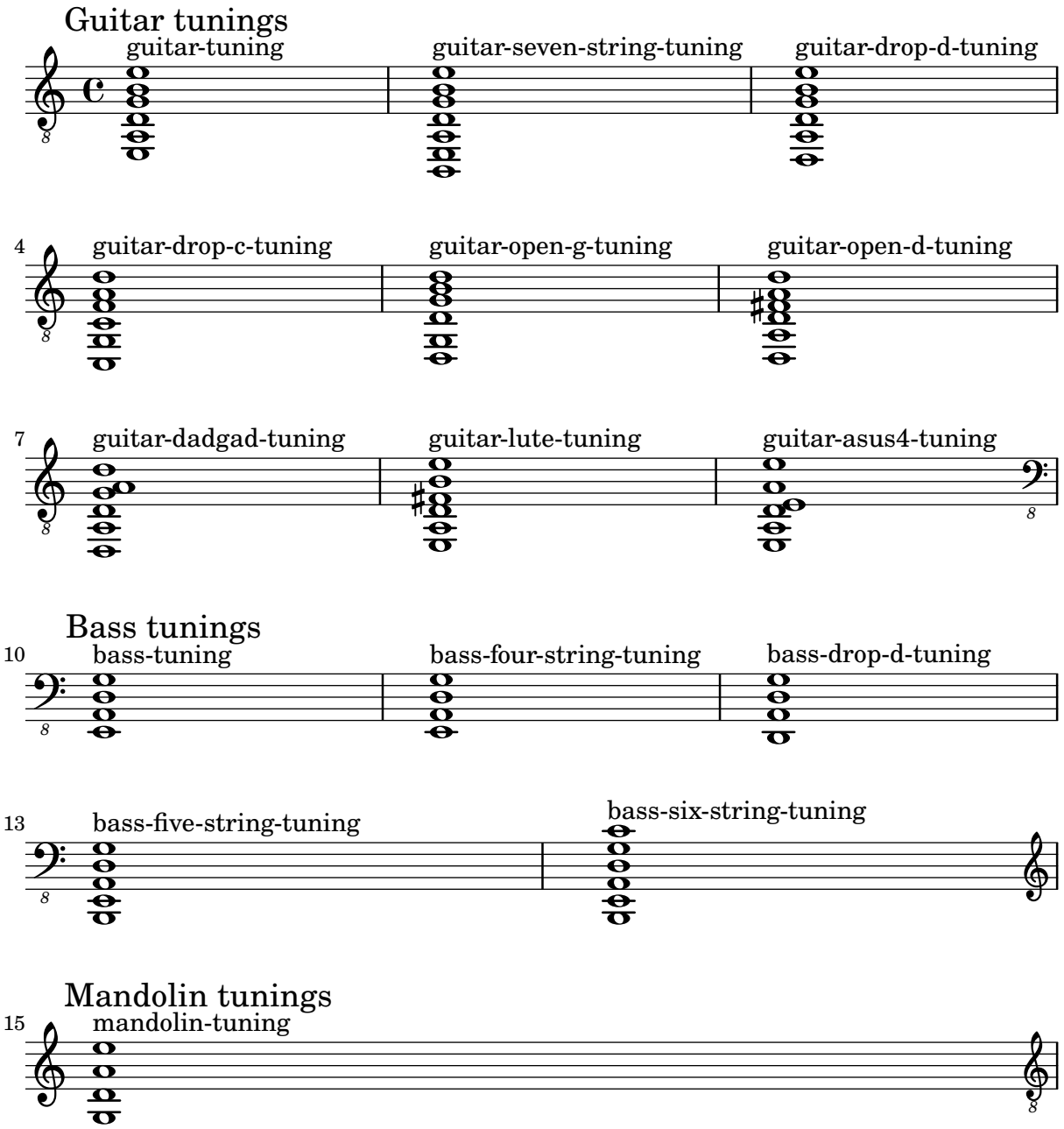
| | | | | |
|---|---|---|---|---|
| Settima aumentata | Triade aumentata, settima minore | `aug7` | `c1:aug7` |  |
| Settima semidiminuita | Triade diminuita, settima minore | `m7.5-` | `c1:m7.5-` |  |
| Minore di settima maggiore | Triade minore, settima maggiore | `m7+` | `c1:m7+` |  |
| Sesta maggiore | Triade maggiore, sesta | `6` | `c1:6` |  |
| Sesta minore | Triade minore, sesta | `m6` | `c1:m6` |  |
| Nona dominante | Settima dominante, nona maggiore | `9` | `c1:9` |  |
| Nona maggiore | Settima maggiore, nona maggiore | `maj9` | `c1:maj9` |  |
| Nona minore | Settima minore, nona maggiore | `m9` | `c1:m9` |  |
| Undicesima dominante | Nona dominante, undicesima perfetta | `11` | `c1:11` |  |

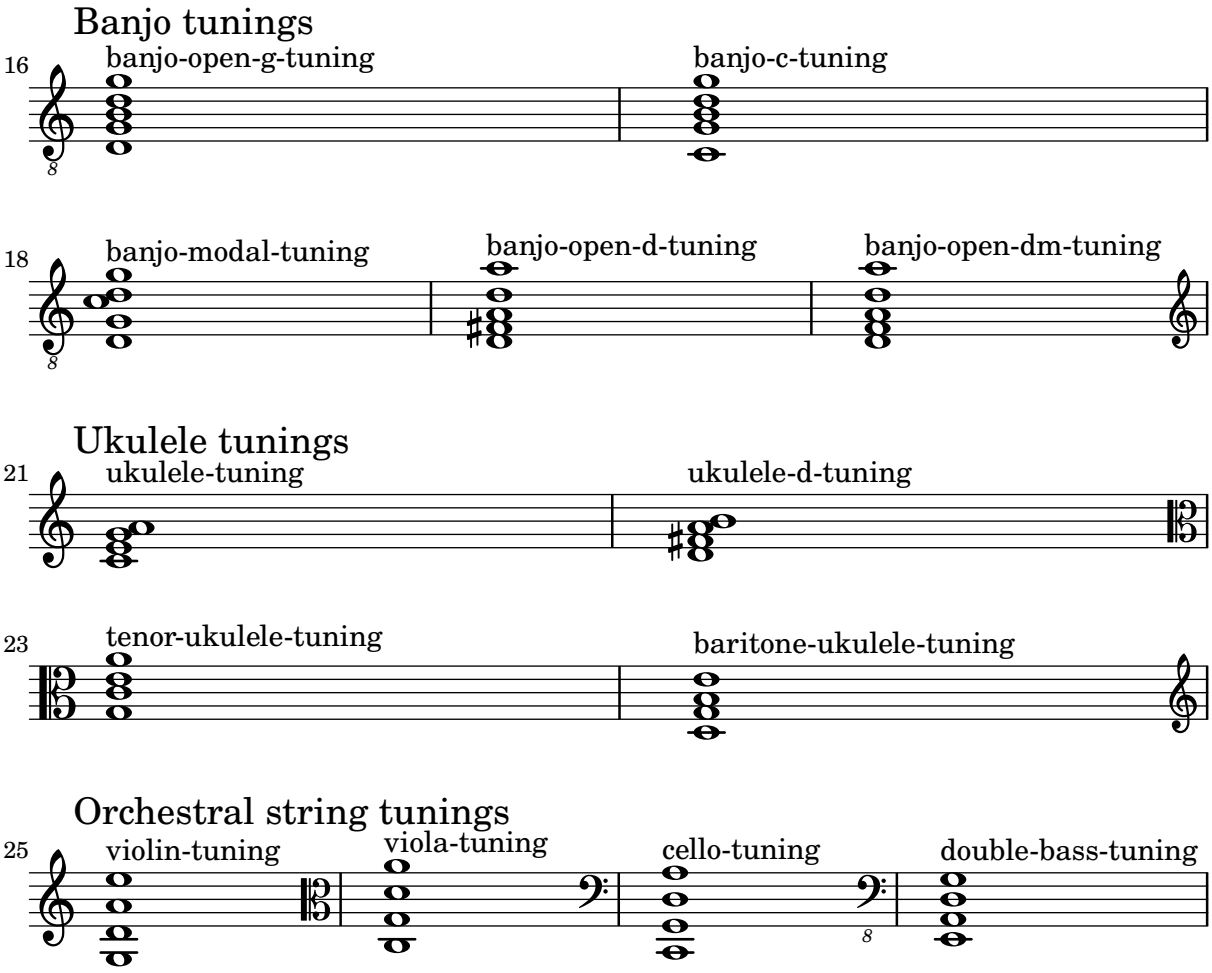
| | | | | |
|---|---|---|---|---|
| Undicesima maggiore | Nona maggiore, undicesima perfetta | `maj11` | `c1:maj11` |  |
| Undicesima minore | Nona minore, undicesima perfetta | `m11` | `c1:m11` |  |
| Tredicesima dominante | Nona dominante, tredicesima maggiore | `13` | `c1:13` |  |
| Tredicesima dominante | Undicesima dominante, tredicesima maggiore | `13.11` | `c1:13.11` |  |
| Tredicesima maggiore | Undicesima maggiore, tredicesima maggiore | `maj13.11` | `c1:maj13.11` |  |
| Tredicesima minore | Undicesima minore, tredicesima maggiore | `m13.11` | `c1:m13.11` |  |
| Sospeso di seconda | Seconda maggiore, quinta perfetta | `sus2` | `c1:sus2` |  |
| Sospeso di quarta | Quarta perfetta, quinta perfetta | `sus4` | `c1:sus4` |  |

| | | | |
|---|---|---|---|
| Power chord (bicordo) | Quinta perfetta | `1.5` | `c1:5` |
| Power chord (tricordo) | Quinta perfetta, ottava | `1.5.8` | `c1:5.8` |

## A.3 Accordature predefinite

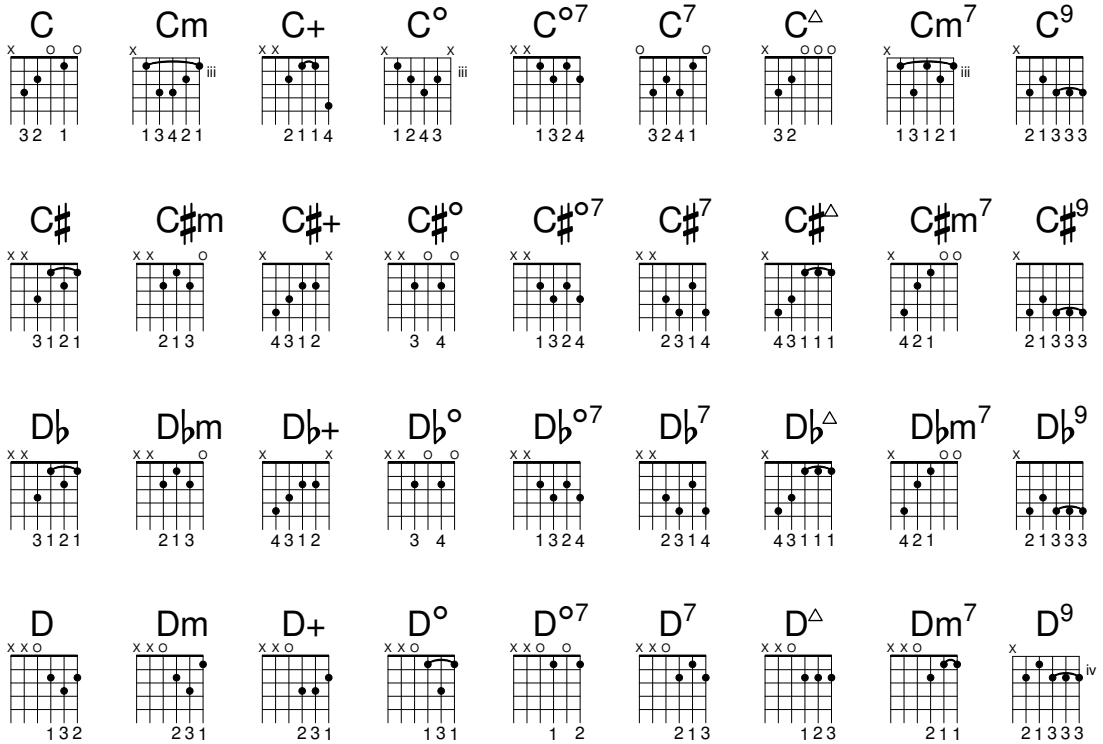
Il grafico seguente mostra le accordature predefinite degli strumenti a corda.

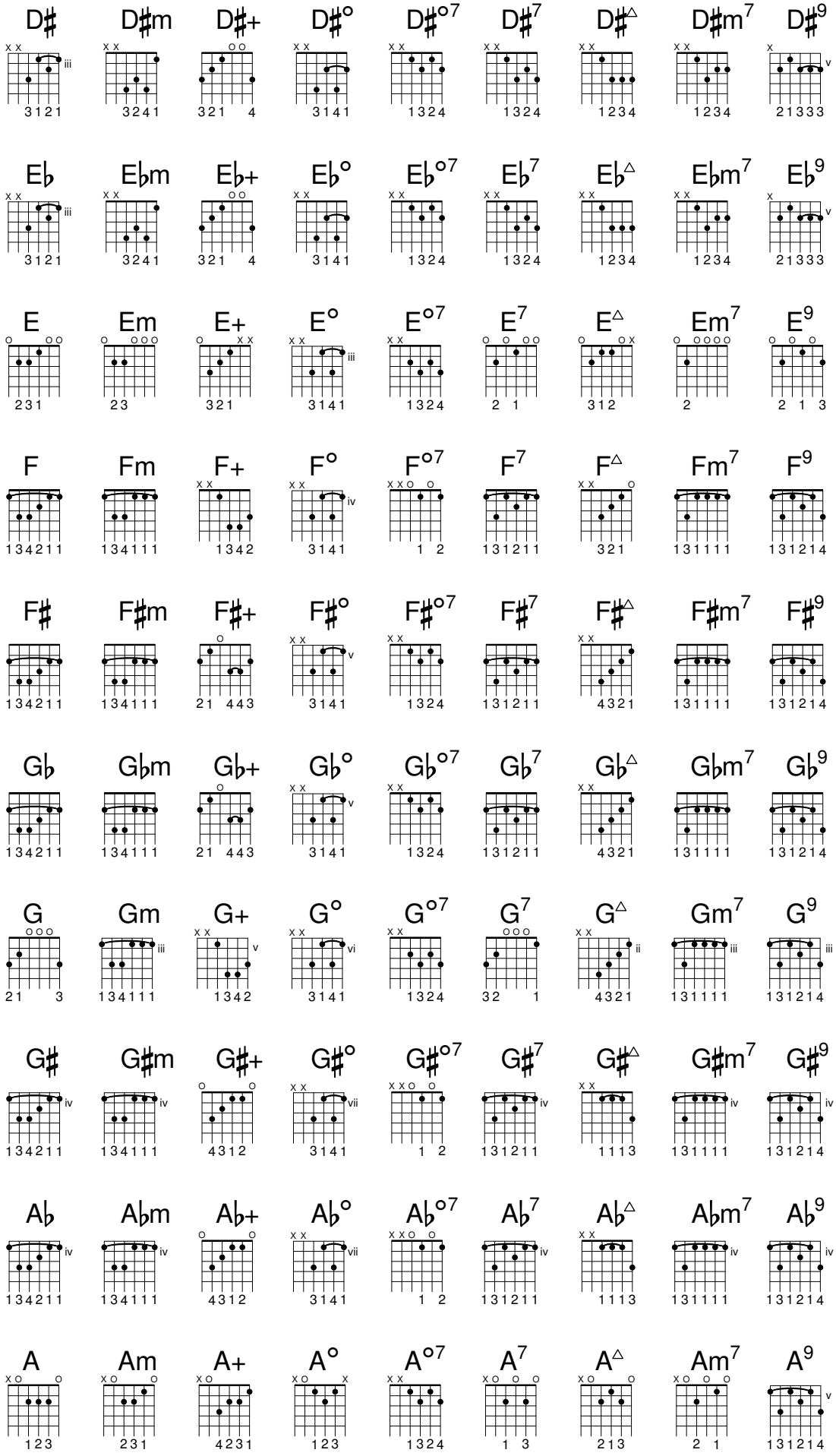


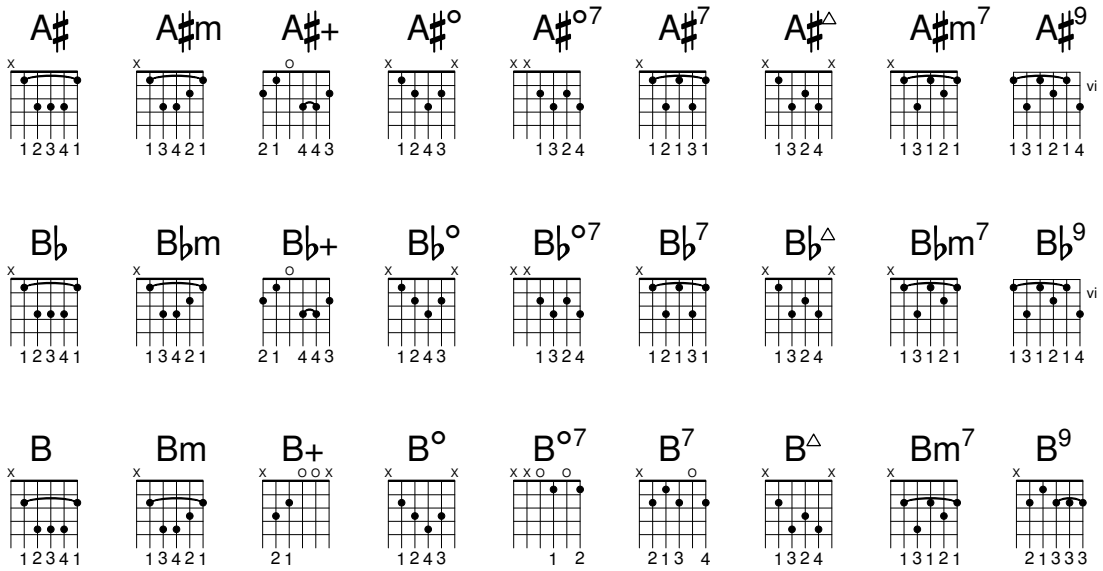



## A.4 Diagrammi degli accordi predefiniti

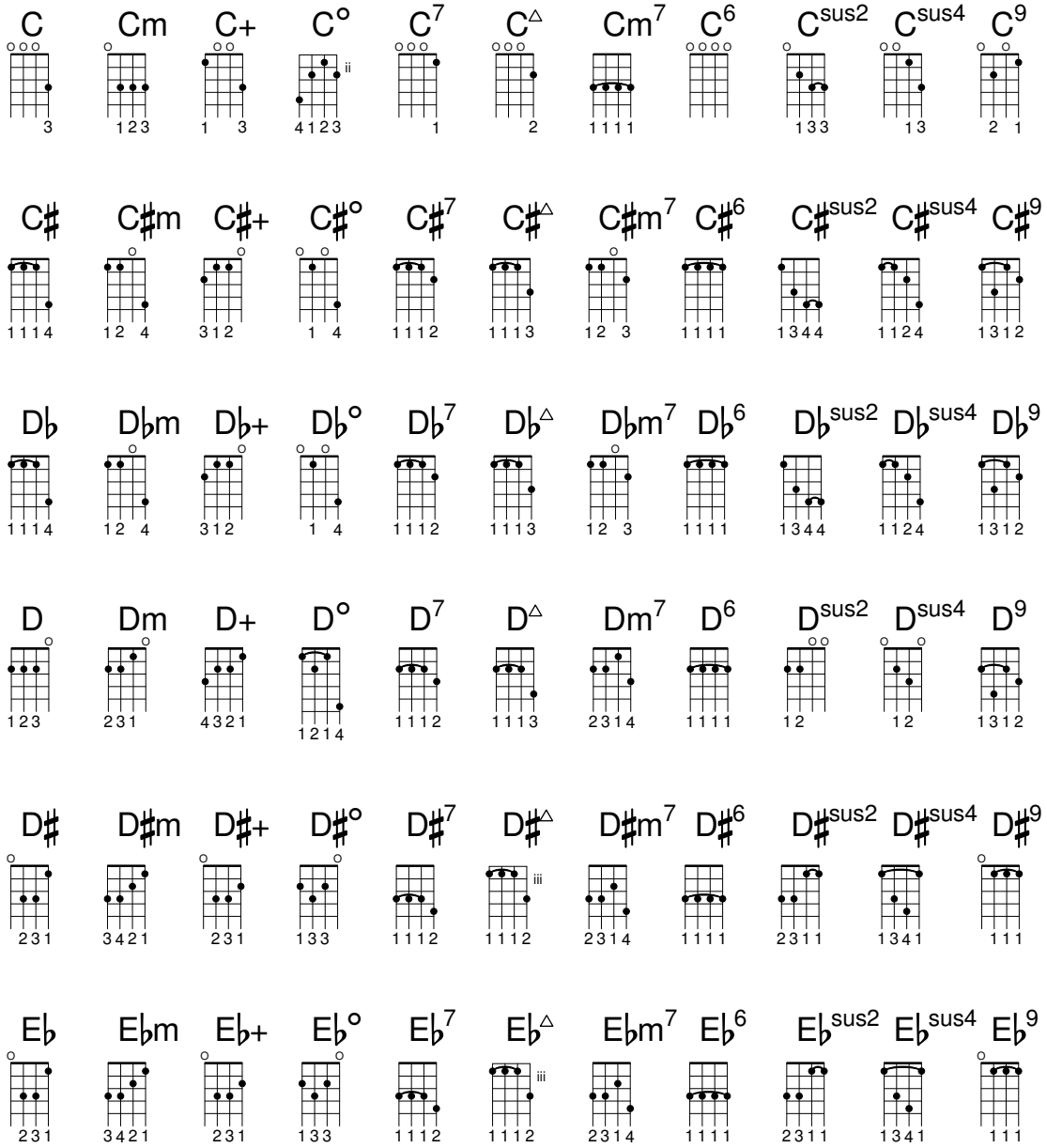
### Diagrammi per chitarra

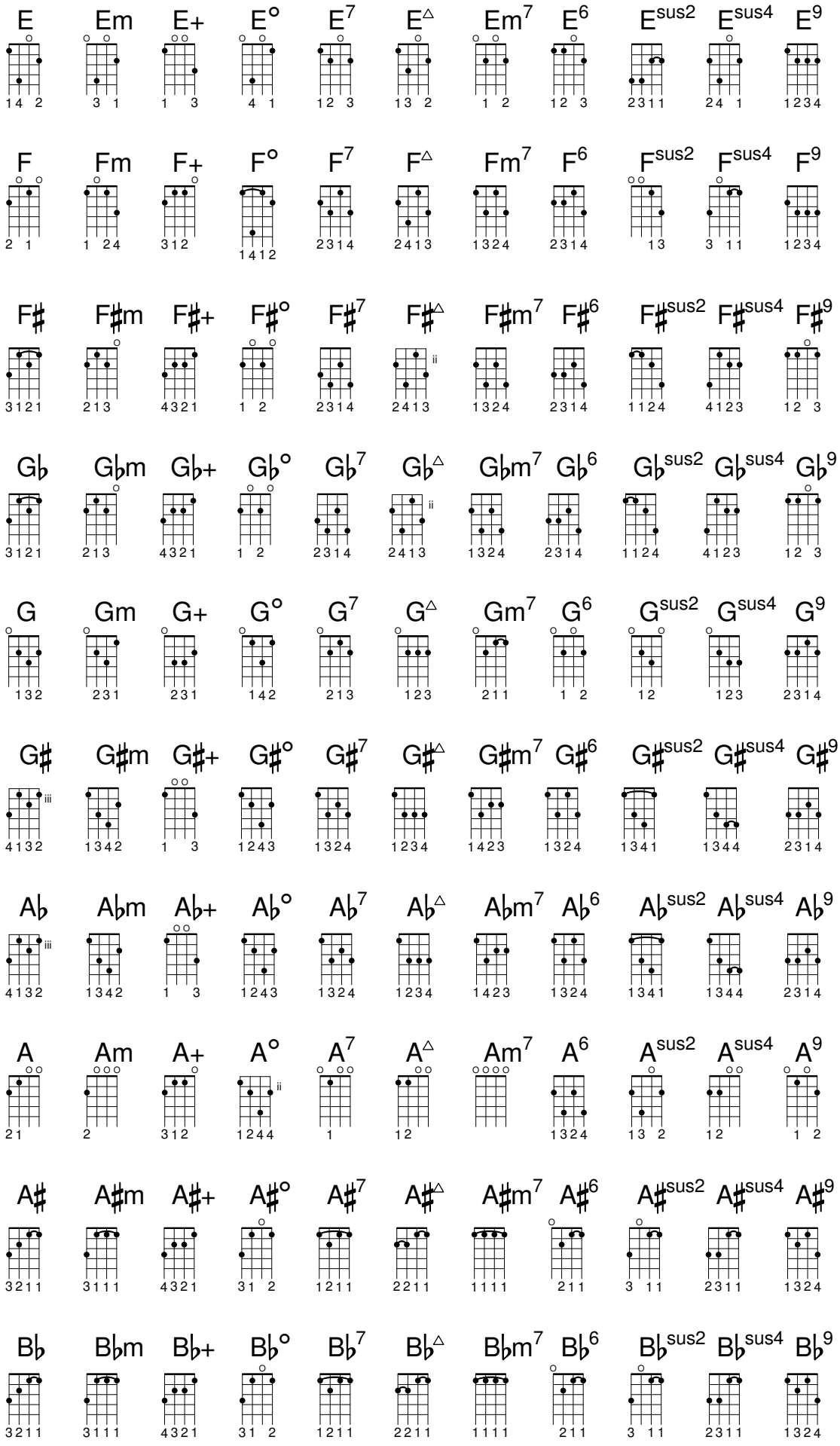


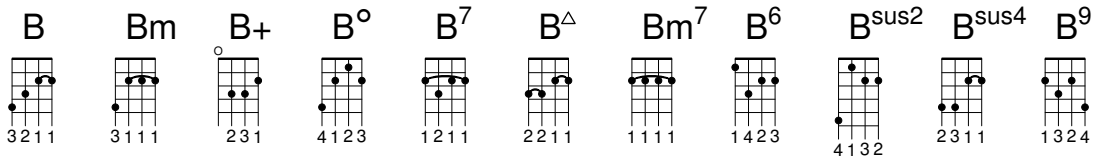

D♯ D♯m D♯+ D♯° D♯°7 D♯7 D♯△ D♯m7 D♯9
E♭ E♭m E♭+ E♭° E♭°7 E♭7 E♭△ E♭m7 E♭9
E Em E+ E° E°7 E7 E△ Em7 E9
F Fm F+ F° F°7 F7 F△ Fm7 F9
F♯ F♯m F♯+ F♯° F♯°7 F♯7 F♯△ F♯m7 F♯9
G♭ G♭m G♭+ G♭° G♭°7 G♭7 G♭△ G♭m7 G♭9
G Gm G+ G° G°7 G7 G△ Gm7 G9
G♯ G♯m G♯+ G♯° G♯°7 G♯7 G♯△ G♯m7 G♯9
A♭ A♭m A♭+ A♭° A♭°7 A♭7 A♭△ A♭m7 A♭9
A Am A+ A° A°7 A7 A△ Am7 A9



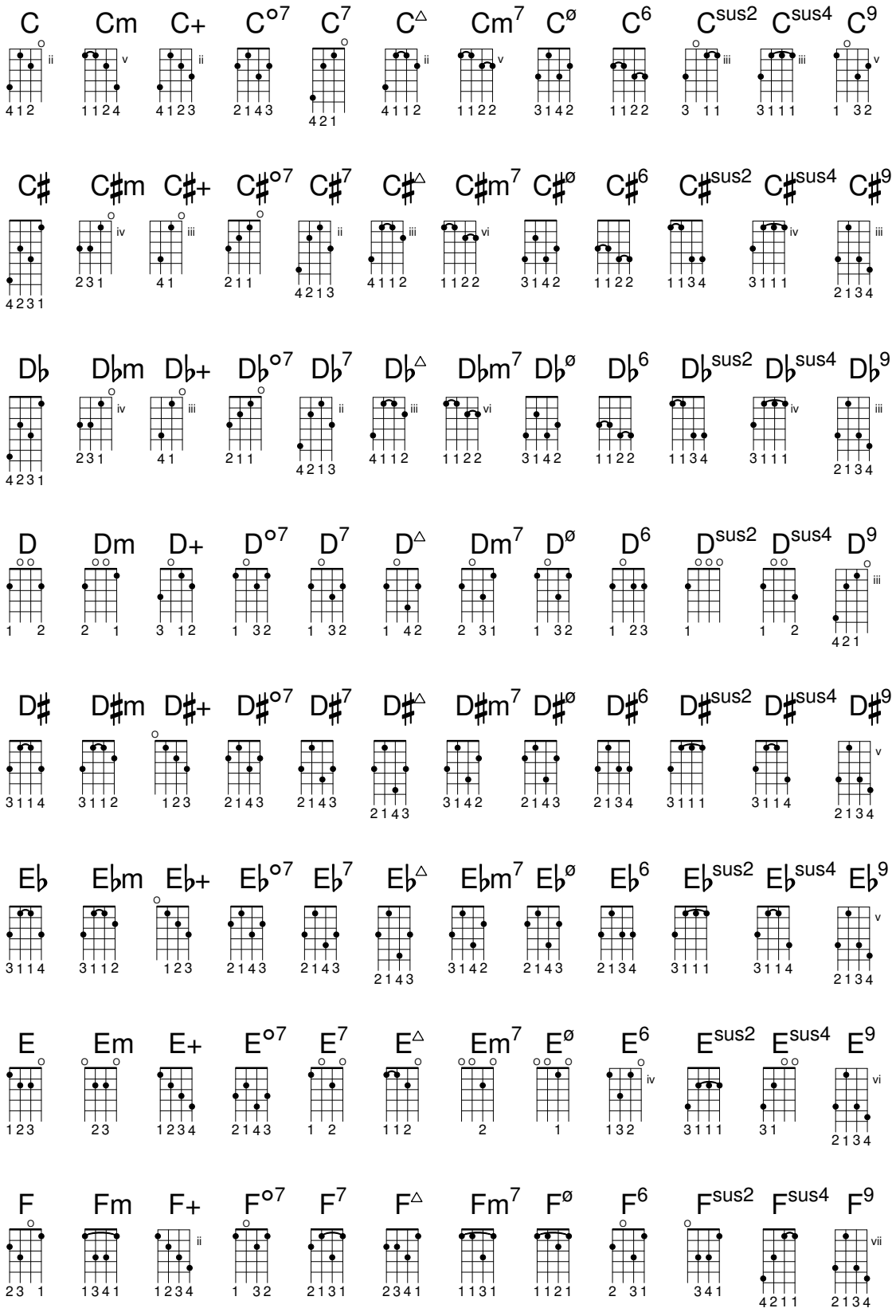

## Diagrammi per ukulele

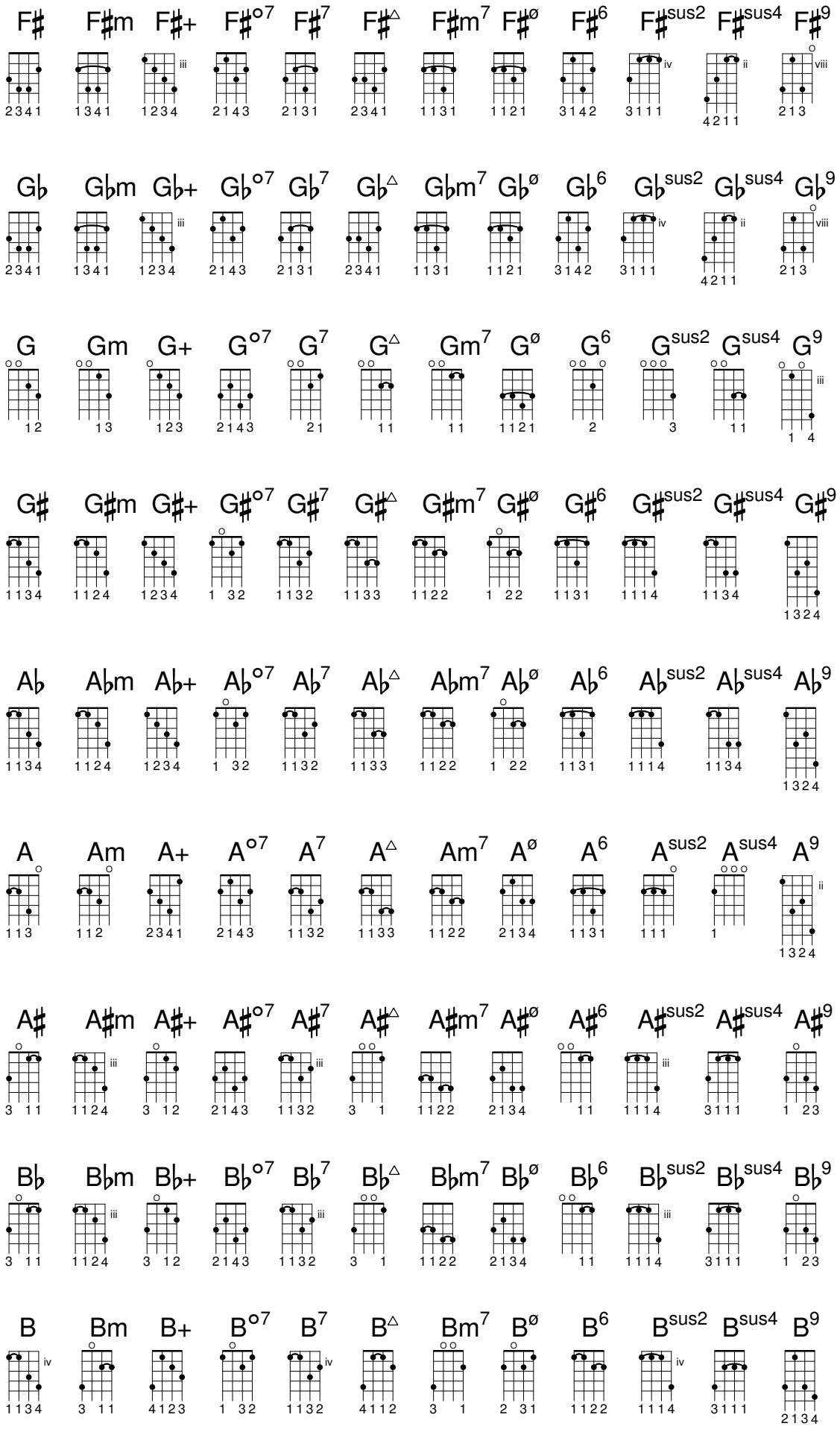



E Em E+ E° E7 E△ Em7 E6 Esus2 Esus4 E9
F Fm F+ F° F7 F△ Fm7 F6 Fsus2 Fsus4 F9
F♯ F♯m F♯+ F♯° F♯7 F♯△ F♯m7 F♯6 F♯sus2 F♯sus4 F♯9
G♭ G♭m G♭+ G♭° G♭7 G♭△ G♭m7 G♭6 G♭sus2 G♭sus4 G♭9
G Gm G+ G° G7 G△ Gm7 G6 Gsus2 Gsus4 G9
G♯ G♯m G♯+ G♯° G♯7 G♯△ G♯m7 G♯6 G♯sus2 G♯sus4 G♯9
A♭ A♭m A♭+ A♭° A♭7 A♭△ A♭m7 A♭6 A♭sus2 A♭sus4 A♭9
A Am A+ A° A7 A△ Am7 A6 Asus2 Asus4 A9
A♯ A♯m A♯+ A♯° A♯7 A♯△ A♯m7 A♯6 A♯sus2 A♯sus4 A♯9
B♭ B♭m B♭+ B♭° B♭7 B♭△ B♭m7 B♭6 B♭sus2 B♭sus4 B♭9




## Diagrammi per mandolino




F♯ F♯m F♯+ F♯°7 F♯7 F♯△ F♯m7 F♯ø F♯6 F♯sus2 F♯sus4 F♯9
G♭ G♭m G♭+ G♭°7 G♭7 G♭△ G♭m7 G♭ø G♭6 G♭sus2 G♭sus4 G♭9
G Gm G+ G°7 G7 G△ Gm7 Gø G6 Gsus2 Gsus4 G9
G♯ G♯m G♯+ G♯°7 G♯7 G♯△ G♯m7 G♯ø G♯6 G♯sus2 G♯sus4 G♯9
A♭ A♭m A♭+ A♭°7 A♭7 A♭△ A♭m7 A♭ø A♭6 A♭sus2 A♭sus4 A♭9
A Am A+ A°7 A7 A△ Am7 Aø A6 Asus2 Asus4 A9
A♯ A♯m A♯+ A♯°7 A♯7 A♯△ A♯m7 A♯ø A♯6 A♯sus2 A♯sus4 A♯9
B♭ B♭m B♭+ B♭°7 B♭7 B♭△ B♭m7 B♭ø B♭6 B♭sus2 B♭sus4 B♭9
B Bm B+ B°7 B7 B△ Bm7 Bø B6 Bsus2 Bsus4 B9

## A.5 Formati carta predefiniti

I formati carta sono definiti in `scm/paper.scm`

**La serie A dell'"ISO 216"**

`"a10"` (26 x 37 mm)

`"a9"` (37 x 52 mm)

`"a8"` (52 x 74 mm)

`"a7"` (74 x 105 mm)

`"a6"` (105 x 148 mm)

`"a5"` (148 x 210 mm)

`"a4"` (210 x 297 mm)

`"a3"` (297 x 420 mm)

`"a2"` (420 x 594 mm)

`"a1"` (594 x 841 mm)

`"a0"` (841 x 1189 mm)

**La serie B dell'"ISO 216"**

`"b10"` (31 x 44 mm)

`"b9"` (44 x 62 mm)

`"b8"` (62 x 88 mm)

`"b7"` (88 x 125 mm)

`"b6"` (125 x 176 mm)

`"b5"` (176 x 250 mm)

`"b4"` (250 x 353 mm)

`"b3"` (353 x 500 mm)

`"b2"` (500 x 707 mm)

`"b1"` (707 x 1000 mm)

`"b0"` (1000 x 1414 mm)

**Due formati estesi, definiti nel "DIN 476"**

`"4a0"` (1682 x 2378 mm)

`"2a0"` (1189 x 1682 mm)

**La serie C standard dell'"ISO 269"**

`"c10"` (28 x 40 mm)

`"c9"` (40 x 57 mm)

`"c8"` (57 x 81 mm)

`"c7"` (81 x 114 mm)

`"c6"` (114 x 162 mm)

`"c5"` (162 x 229 mm)

`"c4"` (229 x 324 mm)

`"c3"` (324 x 458 mm)

`"c2"` (458 x 648 mm)

`"c1"` (648 x 917 mm)

`"c0"` (917 x 1297 mm)

**Formati carta nordamericani**

`"junior-legal"`
(8.0 x 5.0 in)

`"legal"` (8.5 x 14.0 in)

`"ledger"` (17.0 x 11.0 in)

`"letter"` (8.5 x 11.0 in)

`"tabloid"`
(11.0 x 17.0 in)

`"11x17"` (11.0 x 17.0 in)

`"17x11"` (17.0 x 11.0 in)

**Government-letter dell'IEEE Printer Working Group, per la scrittura dei bambini**

`"government-letter"`
(8 x 10.5 in)

`"government-legal"`
(8.5 x 13.0 in)

`"philippine-legal"`
(8.5 x 13.0 in)

**Formati ANSI**

`"ansi a"` (8.5 x 11.0 in)

`"ansi b"` (17.0 x 11.0 in)

`"ansi c"` (17.0 x 22.0 in)

`"ansi d"` (22.0 x 34.0 in)

`"ansi e"` (34.0 x 44.0 in)

`"engineering f"`
(28.0 x 40.0 in)

**Formati nordamericani per l'architettura**

`"arch a"` (9.0 x 12.0 in)

`"arch b"` (12.0 x 18.0 in)

`"arch c"` (18.0 x 24.0 in)

`"arch d"` (24.0 x 36.0 in)

`"arch e"` (36.0 x 48.0 in)

`"arch e1"` (30.0 x 42.0 in)

**Formati antichi ancora in uso nel Regno Unito**

`"statement"`
(5.5 x 8.5 in)

`"half letter"`
(5.5 x 8.5 in)

`"quarto"` (8.0 x 10.0 in)

`"octavo"` (6.75 x 10.5 in)

`"executive"`
(7.25 x 10.5 in)

`"monarch"`
(7.25 x 10.5 in)

`"foolscap"`
(8.27 x 13.0 in)

`"folio"` (8.27 x 13.0 in)

`"super-b"`
(13.0 x 19.0 in)

`"post"` (15.5 x 19.5 in)

`"crown"` (15.0 x 20.0 in)

`"large post"`
(16.5 x 21.0 in)

`"demy"` (17.5 x 22.5 in)

`"medium"` (18.0 x 23.0 in)

`"broadsheet"`
(18.0 x 24.0 in)

`"royal"` (20.0 x 25.0 in)

`"elephant"`
(23.0 x 28.0 in)

`"double demy"`
(22.5 x 35.0 in)

`"quad demy"`
(35.0 x 45.0 in)

`"atlas"` (26.0 x 34.0 in)

`"imperial"`
(22.0 x 30.0 in)

`"antiquarian"`
(31.0 x 53.0 in)

**Formati basati su PA4**

`"pa0"` (840 x 1120 mm)

`"pa1"` (560 x 840 mm)

`"pa2"` (420 x 560 mm)

`"pa3"` (280 x 420 mm)

`"pa4"` (210 x 280 mm)

`"pa5"` (140 x 210 mm)

`"pa6"` (105 x 140 mm)

`"pa7"` (70 x 105 mm)

`"pa8"` (52 x 70 mm)

`"pa9"` (35 x 52 mm)

`"pa10"` (26 x 35 mm)

**Formato usato nel Sudest asiatico e in Australia**

`"f4"` (210 x 330 mm)

**Formato usato in esempi `@lilypond` molto piccoli della documentazione, basato sul formato a8 landscape.**

`"a8landscape"`
(74 x 52 mm)

## A.6 Strumenti MIDI

Di seguito un elenco dei nomi che possono essere usati per la proprietà `midiInstrument`. L'ordine degli strumenti, a partire dalla colonna sinistra e proseguendo in basso, corrisponde ai 128 "numeri di programma" dello standard General MIDI.

```
acoustic grand
bright acoustic
electric grand
honky-tonk
electric piano 1
electric piano 2
harpsichord
clav
celesta
glockenspiel
music box
vibraphone
marimba
xylophone
tubular bells
dulcimer
drawbar organ
percussive organ
rock organ
church organ
reed organ
accordion
harmonica
concertina
acoustic guitar (nylon)
acoustic guitar (steel)
electric guitar (jazz)
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
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
```

```
electric guitar (clean)   bassoon              agogo
electric guitar (muted)   clarinet            steel drums
overdriven guitar         piccolo             woodblock
distorted guitar          flute               taiko drum
guitar harmonics          recorder            melodic tom
acoustic bass             pan flute           synth drum
electric bass (finger)    blown bottle        reverse cymbal
electric bass (pick)      shakuhachi          guitar fret noise
fretless bass             whistle             breath noise
slap bass 1               ocarina             seashore
slap bass 2               lead 1 (square)     bird tweet
synth bass 1              lead 2 (sawtooth)   telephone ring
synth bass 2              lead 3 (calliope)   helicopter
violin                    lead 4 (chiff)      applause
viola                     lead 5 (charang)    gunshot
cello                     lead 6 (voice)
```

## A.7 Elenco dei colori

### Colori normali

La sintassi è descritta in [Colorare gli oggetti], pagina 229.

```
black       white          red         green
blue        cyan           magenta     yellow
grey        darkred        darkgreen   darkblue
darkcyan    darkmagenta    darkyellow
```

### Nomi di colore X

I nomi di colore X hanno diverse varianti:

Qualsiasi nome scritto come un'unica parola con lettere iniziali maiuscole (es: 'LightSlateBlue') può essere scritto anche con parole separate da spazio e senza maiuscola (es: 'light slate blue').

La parola 'grey' può essere sempre scritta come 'gray' (es: 'DarkSlateGray').

Alcuni nomi possono avere un suffisso numerico (es: 'LightSalmon4').

### Nomi di colori senza un suffisso numerico

```
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
```

### Nomi di colori con un suffisso numerico

Nei seguenti nomi il suffisso N può essere un numero compreso tra 1 e 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
```

### Scala di grigi

Una scala di grigi si ottiene con:

```
greyN
```

dove N è un numero compreso tra 0 e 100.

## A.8 Il font Emmentaler

Il font Emmentaler è composto da due *sottoinsiemi* di glifi. “Feta”, usato nella notazione classica e “Parmesan”, usato nella notazione antica.

A qualsiasi glifo del font Emmentaler si può accedere direttamente usando del testo insieme al nome del glifo (come è mostrato nelle tabelle seguenti). Per esempio:

```
g^\markup {\musicglyph "scripts.segno" }
```

oppure

```
\markup {\musicglyph "five"}.
```

Maggiori informazioni in Sezione 1.8.2 [Formattazione del testo], pagina 246.

## Glifi della chiave

| | | | |
|---|---|---|---|
| clefs.C |  | clefs.C_change |  |
| clefs.varC |  | clefs.varC_change |  |
| clefs.F |  | clefs.F_change |  |
| clefs.G |  | clefs.G_change |  |
| clefs.GG |  | clefs.GG_change |  |
| clefs.tenorG |  | clefs.tenorG_change |  |
| clefs.percussion |  | clefs.percussion_change |  |
| clefs.varpercussion |  | clefs .varpercussion_change |  |
| clefs.tab |  | clefs.tab_change |  |

## Glifi delle indicazioni di tempo

| | | | |
|---|---|---|---|
| timesig.C44 |  | timesig.C22 |  |

## Glifi dei numeri

| | | | |
|---|---|---|---|
| plus | + | comma | , |
| hyphen | - | period | . |
| zero | 0 | one | 1 |

two 

three 

four 

five 

six 

seven 

eight 

nine 

## Glifi delle alterazioni

accidentals.sharp 

accidentals
.sharp.arrowup 

accidentals
.sharp.arrowdown 

accidentals
.sharp.arrowboth 

accidentals.sharp
.slashslash.stem 

accidentals.sharp
.slashslashslash.stemstem 

accidentals.sharp
.slashslashslash.stem 

accidentals
.sharp.slash.stem 

accidentals.sharp
.slashslash.stemstemstem 

accidentals.doublesharp 

accidentals.natural 

accidentals
.natural.arrowup 

accidentals
.natural.arrowdown 

accidentals
.natural.arrowboth 

accidentals.flat 

accidentals.flat.arrowup 

accidentals
.flat.arrowdown 

accidentals
.flat.arrowboth 

accidentals.flat.slash 

accidentals.flat
.slashslash 

accidentals
.mirroredflat.flat 

accidentals.mirroredflat 

| | | | |
|---|---|---|---|
| accidentals .mirroredflat.backslash |  | accidentals.flatflat |  |
| accidentals .flatflat.slash |  | accidentals.rightparen |  |
| accidentals.leftparen |  | | |

## Glifi delle teste di nota predefinite

| | | | |
|---|---|---|---|
| noteheads.uM2 |  | noteheads.dM2 |  |
| noteheads.sM1 |  | noteheads.s0 |  |
| noteheads.s1 |  | noteheads.s2 |  |

## Glifi delle teste di nota speciali

| | | | |
|---|---|---|---|
| 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 |  |

## Glifi delle teste di nota a forma variabile

| | | | |
|---|---|---|---|
| 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 |  | | |

## Glifi delle pause

rests.0 

rests.1 

rests.0o 

rests.1o 

rests.M3 

rests.M2 

rests.M1 

rests.M1o 

rests.2 

rests.2classical 

rests.2z 

rests.3 
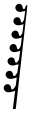
rests.4 

rests.5 

rests.6 

rests.7 

rests.8 

rests.9 

rests.10 

## Glifi delle code

flags.u3 

flags.u4 

flags.u5 

flags.u6 

| | | | |
|---|---|---|---|
| flags.u7 |  | flags.u8 |  |
| flags.u9 |  | flags.u10 |  |
| flags.d3 |  | flags.d4 |  |
| flags.d5 |  | flags.d6 |  |
| flags.d7 |  | flags.d8 |  |
| flags.d9 |  | flags.d10 |  |
| flags.ugrace |  | flags.dgrace |  |

## Glifi dei punti

| | |
|---|---|
| dots.dot |  |

## Glifi delle dinamiche

| | | | |
|---|---|---|---|
| space | | f |  |
| m |  | n |  |
| p |  | r |  |
| s |  | z |  |

## Glifi dei segni

| | | | |
|---|---|---|---|
| scripts.ufermata |  | scripts.dfermata |  |
| scripts .uhenzeshortfermata |  | scripts .dhenzeshortfermata |  |
| scripts .uhenzelongfermata |  | scripts .dhenzelongfermata |  |
| scripts.ushortfermata |  | scripts.dshortfermata |  |
| scripts .uveryshortfermata |  | scripts .dveryshortfermata |  |
| scripts.ulongfermata |  | scripts.dlongfermata |  |
| scripts.uverylongfermata |  | scripts.dverylongfermata |  |
| scripts.thumb |  | scripts.sforzato |  |
| scripts.espr |  | scripts.staccato |  |
| scripts.ustaccatissimo |  | scripts.dstaccatissimo |  |
| scripts.tenuto |  | scripts.uportato |  |
| scripts.dportato |  | scripts.umarcato |  |
| scripts.dmarcato |  | scripts.open |  |
| scripts.halfopen |  | scripts.halfopenvertical |  |

| | | | |
|---|---|---|---|
| scripts.stopped |  | scripts.upbow |  |
| scripts.downbow |  | scripts.reverseturn |  |
| scripts.turn |  | scripts.slashturn |  |
| scripts.haydnturn |  | scripts.trill |  |
| scripts.upedalheel |  | scripts.dpedalheel |  |
| scripts.upedaltoe |  | scripts.dpedaltoe |  |
| scripts.flageolet |  | scripts.segno |  |
| scripts.varsegno |  | scripts.coda |  |
| scripts.varcoda |  | scripts.rcomma |  |
| scripts.lcomma |  | scripts.rvarcomma |  |
| scripts.lvarcomma |  | scripts.arpeggio |  |
| scripts.trill_element |  | scripts.arpeggio .arrow.M1 |  |
| scripts.arpeggio.arrow.1 |  | scripts.trillelement |  |
| scripts.prall |  | scripts.mordent |  |
| scripts.prallprall |  | scripts.prallmordent |  |

scripts.upprall 

scripts.upmordent 

scripts.prallup 

scripts.downprall 

scripts.downmordent 

scripts.pralldown 

scripts.lineprall 

scripts.caesura.curved 

scripts.caesura.straight 

scripts.tickmark 

scripts.snappizzicato 

scripts.ictus 

scripts.uaccentus 

scripts.daccentus 

scripts.usemicirculus 

scripts.dsemicirculus 

scripts.circulus 

scripts.augmentum 

scripts
.usignumcongruentiae 

scripts
.dsignumcongruentiae 

## Glifi delle teste a forma di freccia

arrowheads.open.01 

arrowheads.open.0M1 

arrowheads.open.11 

arrowheads.open.1M1 

arrowheads.close.01 

arrowheads.close.0M1 

arrowheads.close.11 

arrowheads.close.1M1 

## Glifi delle estremità delle parentesi

| | | | |
|---|---|---|---|
| brackettips.up |  | brackettips.down |  |

## Glifi dei pedali

| | | | |
|---|---|---|---|
| pedal.* |  | pedal.M |  |
| pedal.. | . | pedal.P |  |
| pedal.d |  | pedal.e |  |
| pedal.Ped |  | | |

## Glifi della fisarmonica

| | | | |
|---|---|---|---|
| accordion.discant |  | accordion.dot |  |
| accordion.freebass |  | accordion.stdbass |  |
| accordion.bayanbass |  | accordion.oldEE |  |
| accordion.push |  | accordion.pull |  |

## Glifi delle legature di valore

| | | | |
|---|---|---|---|
| ties.lyric.short |  | ties.lyric.default |  |

## Glifi della notazione vaticana

clefs.vaticana.do 

clefs.vaticana.do_change 

clefs.vaticana.fa 

clefs.vaticana.fa_change 

custodes.vaticana.u0 

custodes.vaticana.u1 

custodes.vaticana.u2 

custodes.vaticana.d0 

custodes.vaticana.d1 

custodes.vaticana.d2 

accidentals.vaticanaM1 

accidentals.vaticana0 

dots.dotvaticana 

noteheads<br>.svaticana.punctum 

noteheads.svaticana<br>.punctum.cavum 

noteheads.svaticana<br>.linea.punctum 

noteheads.svaticana<br>.linea.punctum.cavum 

noteheads.svaticana<br>.inclinatum 

noteheads.svaticana.lpes 

noteheads<br>.svaticana.vlpes 

noteheads.svaticana.upes 

noteheads<br>.svaticana.vupes 

noteheads<br>.svaticana.plica 

noteheads<br>.svaticana.vplica 

noteheads<br>.svaticana.epiphonus 

noteheads.svaticana<br>.vepiphonus 

noteheads.svaticana<br>.reverse.plica 

noteheads.svaticana<br>.reverse.vplica 

noteheads.svaticana<br>.inner.cephalicus 

noteheads.svaticana<br>.cephalicus 

noteheads<br>.svaticana.quilisma 

## Glifi della notazione medicea

| | | | |
|---|---|---|---|
| clefs.medicaea.do |  | clefs.medicaea.do_change |  |
| clefs.medicaea.fa |  | clefs.medicaea.fa_change |  |
| custodes.medicaea.u0 |  | custodes.medicaea.u1 |  |
| custodes.medicaea.u2 |  | custodes.medicaea.d0 |  |
| custodes.medicaea.d1 |  | custodes.medicaea.d2 |  |
| accidentals.medicaeaM1 |  | noteheads.smedicaea .inclinatum |  |
| noteheads .smedicaea.punctum |  | noteheads .smedicaea.rvirga |  |
| noteheads .smedicaea.virga |  | | |

## Glifi Hufnagel

| | | | |
|---|---|---|---|
| clefs.hufnagel.do |  | clefs.hufnagel.do_change |  |
| clefs.hufnagel.fa |  | clefs.hufnagel.fa_change |  |
| clefs.hufnagel.do.fa |  | clefs.hufnagel .do.fa_change |  |
| custodes.hufnagel.u0 |  | custodes.hufnagel.u1 |  |
| custodes.hufnagel.u2 |  | custodes.hufnagel.d0 |  |
| custodes.hufnagel.d1 |  | custodes.hufnagel.d2 |  |

accidentals.hufnagelM1 

noteheads .shufnagel.punctum 

noteheads .shufnagel.virga 

noteheads.shufnagel.lpes 

## Glifi della notazione mensurale

rests.M3mensural 

rests.M2mensural 

rests.M1mensural 

rests.0mensural 

rests.1mensural 

rests.2mensural 

rests.3mensural 

rests.4mensural 

clefs.mensural.c 

clefs.mensural.c_change 

clefs.blackmensural.c 

clefs.blackmensural .c_change 

clefs.mensural.f 

clefs.mensural.f_change 

clefs.mensural.g 

clefs.mensural.g_change 

custodes.mensural.u0 

custodes.mensural.u1 

custodes.mensural.u2 

custodes.mensural.d0 

custodes.mensural.d1 

custodes.mensural.d2 

accidentals.mensural1 

accidentals.mensuralM1 

flags.mensuralu03 

flags.mensuralu13 

flags.mensuralu23 

flags.mensurald03 

flags.mensurald13 

flags.mensurald23 

flags.mensuralu04 

flags.mensuralu14 

flags.mensuralu24 

flags.mensurald04 

flags.mensurald14 

flags.mensurald24 

flags.mensuralu05 

flags.mensuralu15 

flags.mensuralu25 

flags.mensurald05 

flags.mensurald15 

flags.mensurald25 

flags.mensuralu06 

flags.mensuralu16 

flags.mensuralu26 

flags.mensurald06 

flags.mensurald16 

flags.mensurald26 

timesig.mensural44 

timesig.mensural22 

timesig.mensural32 

timesig.mensural64 

timesig.mensural94 

timesig.mensural34 

| | | | |
|---|---|---|---|
| timesig.mensural68 |  | timesig.mensural98 |  |
| timesig.mensural48 |  | timesig.mensural68alt |  |
| timesig.mensural24 |  | noteheads.uM3mensural |  |
| noteheads.dM3mensural |  | noteheads.sM3ligmensural |  |
| noteheads.uM2mensural |  | noteheads.dM2mensural |  |
| noteheads.sM2ligmensural |  | noteheads.sM1mensural |  |
| noteheads.urM3mensural |  | noteheads.drM3mensural |  |
| noteheads .srM3ligmensural |  | noteheads.urM2mensural |  |
| noteheads.drM2mensural |  | noteheads .srM2ligmensural |  |
| noteheads.srM1mensural |  | noteheads .uM3semimensural |  |
| noteheads .dM3semimensural |  | noteheads .sM3semiligmensural |  |
| noteheads .uM2semimensural |  | noteheads .dM2semimensural |  |
| noteheads .sM2semiligmensural |  | noteheads .sM1semimensural |  |
| noteheads .urM3semimensural |  | noteheads .drM3semimensural |  |
| noteheads .srM3semiligmensural |  | noteheads .urM2semimensural |  |

noteheads .drM2semimensural

noteheads .srM2semiligmensural

noteheads .srM1semimensural

noteheads .uM3blackmensural

noteheads .dM3blackmensural

noteheads .sM3blackligmensural

noteheads .uM2blackmensural

noteheads .dM2blackmensural

noteheads .sM2blackligmensural

noteheads .sM1blackmensural

noteheads.s0mensural

noteheads.s1mensural

noteheads.s2mensural

noteheads .s0blackmensural

## Glifi della notazione neomensurale

rests.M3neomensural

rests.M2neomensural

rests.M1neomensural

rests.0neomensural

rests.1neomensural

rests.2neomensural

rests.3neomensural

rests.4neomensural

clefs.neomensural.c

clefs.neomensural .c_change

timesig.neomensural44

timesig.neomensural22

timesig.neomensural32

timesig.neomensural64

timesig.neomensural94

timesig.neomensural34

| | | | |
|---|---|---|---|
| timesig.neomensural68 |  | timesig.neomensural98 |  |
| timesig.neomensural48 |  | timesig.neomensural68alt |  |
| timesig.neomensural24 |  | noteheads.uM3neomensural |  |
| noteheads.dM3neomensural |  | noteheads.uM2neomensural |  |
| noteheads.dM2neomensural |  | noteheads.sM1neomensural |  |
| noteheads .urM3neomensural |  | noteheads .drM3neomensural |  |
| noteheads .urM2neomensural |  | noteheads .drM2neomensural |  |
| noteheads .srM1neomensural |  | noteheads.s0neomensural |  |
| noteheads.s1neomensural |  | noteheads.s2neomensural |  |

## Glifi Petrucci

| | | | |
|---|---|---|---|
| clefs.petrucci.c1 |  | clefs.petrucci.c1_change |  |
| clefs.petrucci.c2 |  | clefs.petrucci.c2_change |  |
| clefs.petrucci.c3 |  | clefs.petrucci.c3_change |  |
| clefs.petrucci.c4 |  | clefs.petrucci.c4_change |  |
| clefs.petrucci.c5 |  | clefs.petrucci.c5_change |  |

clefs.petrucci.f 

clefs.petrucci.f_change 

clefs.petrucci.g 

clefs.petrucci.g_change 

noteheads.s0petrucci 

noteheads.s1petrucci 

noteheads.s2petrucci 

noteheads
.s0blackpetrucci 

noteheads
.s1blackpetrucci 

noteheads
.s2blackpetrucci 

## Glifi Solesmes

noteheads.ssolesmes
.incl.parvum 

noteheads
.ssolesmes.auct.asc 

noteheads
.ssolesmes.auct.desc 

noteheads.ssolesmes
.incl.auctum 

noteheads
.ssolesmes.stropha 

noteheads.ssolesmes
.stropha.aucta 

noteheads
.ssolesmes.oriscus 

## Glifi della notazione di Kiev

clefs.kievan.do 

clefs.kievan.do_change 

accidentals.kievan1 

accidentals.kievanM1 

scripts.barline.kievan 

dots.dotkievan 

noteheads.sM2kievan 

noteheads.sM1kievan 

noteheads.s0kievan 

noteheads.d2kievan 

noteheads.u2kievan



noteheads.s1kievan



noteheads.sr1kievan



noteheads.d3kievan

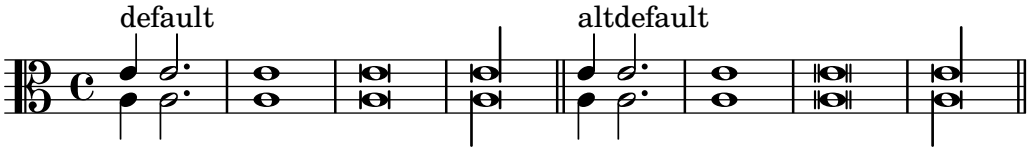


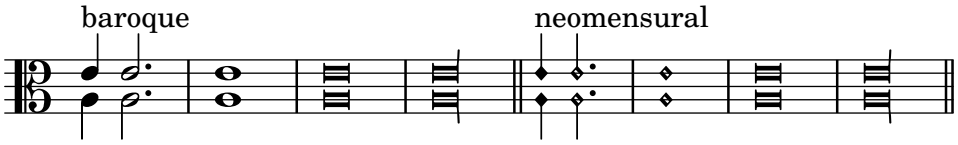
noteheads.u3kievan

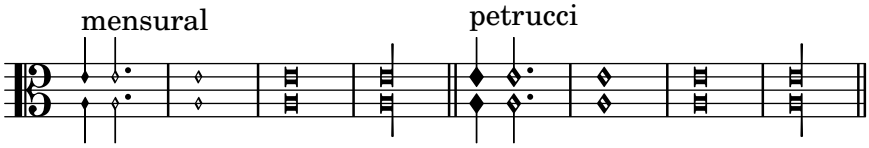


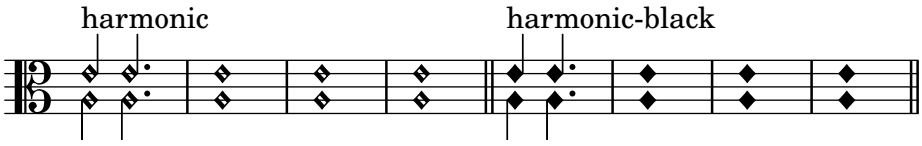
## A.9 Stili delle teste di nota

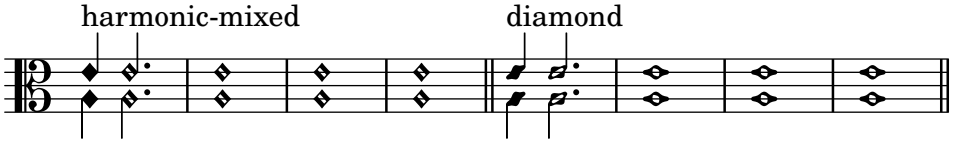
Si possono usare i seguenti stili per le teste di nota.

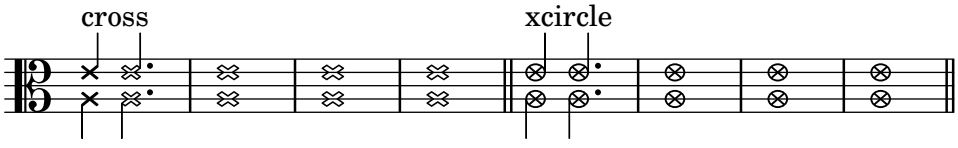


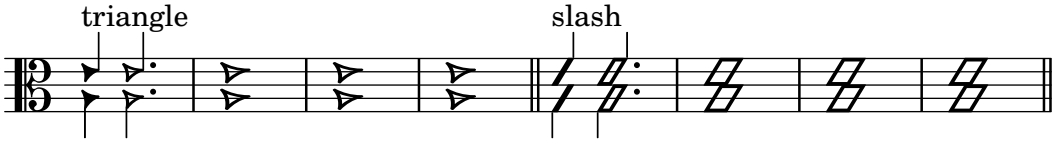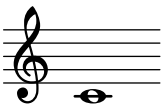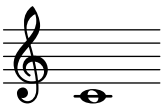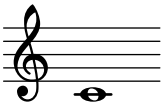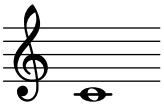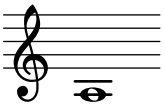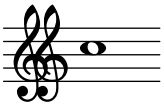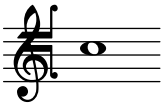



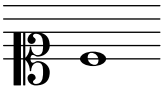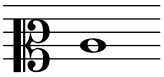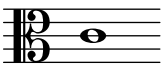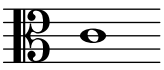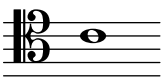
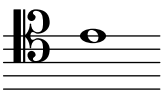
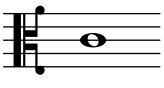
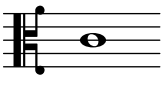



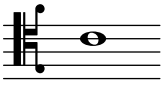
## A.10 Stili della chiave

La seguente tabella mostra tutti gli stili di chiave possibili (inclusi quelli in cui la posizione del *Do centrale*) cambia a seconda della chiave).
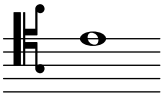
### Chiavi standard

| Esempio | Output | Esempio | Output |
|---|---|---|---|
| `\clef G` |  | `\clef "G2"` |  |
| `\clef treble` |  | `\clef violin` |  |
| `\clef french` |  | `\clef GG` |  |
| `\clef tenorG` |  | | |
| `\clef soprano` |  | `\clef mezzosoprano` |  |
| `\clef C` |  | `\clef alto` |  |
| `\clef tenor` |  | `\clef baritone` |  |
| `\clef varC` |  | `\clef altovarC` |  |
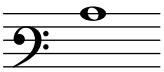
```
\clef tenorvarC
```
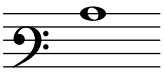


```
\clef baritonevarC
```
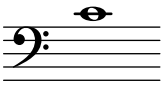


```
\clef varbaritone
```
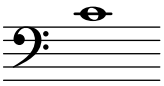


```
\clef baritonevarF
```



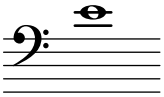
```
\clef F
```



```
\clef bass
```



```
\clef subbass
```



## Chiave del rigo della percussione

**Esempio**

```
\clef percussion
```

**Output**



**Esempio**

```
\clef varpercussion
```

**Output**

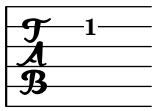


## Chiavi del rigo dell'intavolatura

**Esempio**

```
 \new TabStaff
{
  \clef tab
}
```

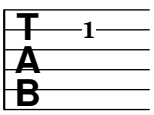
**Output**



**Esempio**

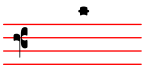
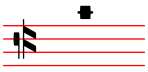
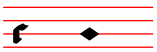
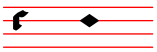
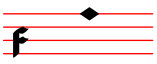
```
 \new TabStaff {
  \clef moderntab
}
```

**Output**

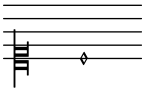
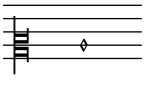


## Chiavi della musica antica

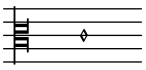
**Gregoriane**

| Esempio | Output | Esempio | Output |
|---|---|---|---|
| `\clef "vaticana-do1"` |  | `\clef "vaticana-do2"` |  |
| `\clef "vaticana-do3"` |  | `\clef "vaticana-fa1"` |  |
| `\clef "vaticana-fa2"` |  | | |
| `\clef "medicaea-do1"` |  | `\clef "medicaea-do2"` |  |
| `\clef "medicaea-do3"` |  | `\clef "medicaea-fa1"` |  |
| `\clef "medicaea-fa2"` |  | | |
| `\clef "hufnagel-do1"` |  | `\clef "hufnagel-do2"` |  |
| `\clef "hufnagel-do3"` |  | `\clef "hufnagel-fa1"` |  |
| `\clef "hufnagel-fa2"` |  | `\clef "hufnagel-do-fa"` |  |

### Mensurali

| Esempio | Output | Esempio | Output |
|---|---|---|---|
| `\clef "mensural-c1"` |  | `\clef "mensural-c2"` |  |

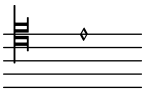
\clef "mensural-c3"



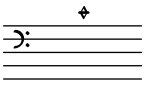
\clef "mensural-c4"



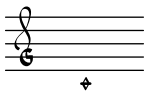
\clef "mensural-c5"



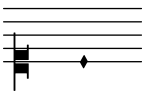
\clef "mensural-f"



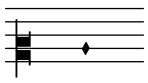
\clef "mensural-g"



\clef "blackmensural-c1"



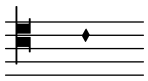
\clef "blackmensural-c2"



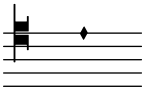
\clef "blackmensural-c3"



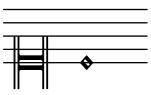
\clef "blackmensural-c4"
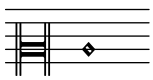


\clef "blackmensural-c5"



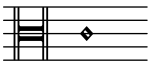
\clef "neomensural-c1"
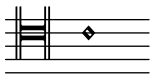


\clef "neomensural-c2"



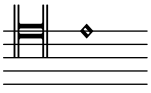
\clef "neomensural-c3"



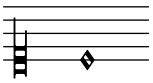
\clef "neomensural-c4"



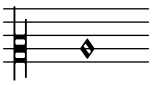
\clef "neomensural-c5"



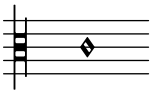
```
\clef "petrucci-c1"
```



```
\clef "petrucci-c2"
```



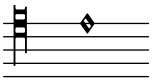
```
\clef "petrucci-c3"
```



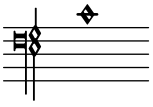
```
\clef "petrucci-c4"
```



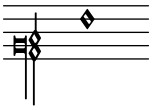
```
\clef "petrucci-c5"
```



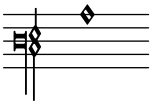
```
\clef "petrucci-f"
```



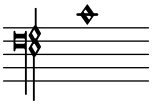
```
\clef "petrucci-f2"
```



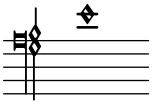
```
\clef "petrucci-f3"
```
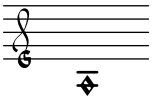


```
\clef "petrucci-f4"
```



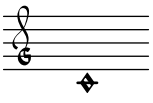
```
\clef "petrucci-f5"
```
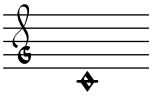


```
\clef "petrucci-g1"
```



```
\clef "petrucci-g2"
```



```
\clef "petrucci-g"
```

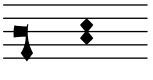


**Kievane**

**Esempio**

```
\clef "kievan-do"
```

**Output**



## A.11 Comandi per *markup*

Tutti i comandi seguenti possono essere usati all'interno di `\markup`.

The following commands can all be used inside `\markup { }`.

### A.11.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)
: Use *size* as the absolute font size (in points) to display *arg*. Adjusts `baseline-skip` and `word-space` accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size text font size 16 text font size 12

Used properties:

- `baseline-skip` (3)
- `word-space` (0.6)

`\bold` *arg* (markup)
: Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

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. }
}
```

V. S.

Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps` *arg* (markup)
Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic` *arg* (markup)
Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like 'più **f**', the normal words (like 'più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

***sfzp***

`\finger` *arg* (markup)
Set *arg* as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

**1 2 3 4 5**

`\fontCaps` *arg* (markup)
Set `font-shape` to `caps`

Note: `\fontCaps` requires the installation and selection of fonts which support the `caps` font shape.

`\fontsize` *increment* (number) *arg* (markup)
Add *increment* to the font-size. Adjusts `baseline-skip` accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

default smaller

Used properties:

- `baseline-skip` (2)

- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

default huge

`\italic` *arg* (markup)
Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

default large

`\larger` *arg* (markup)
Increase the font size relative to the current setting.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

default larger

`\magnify` *sz* (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

```
A \magnify #1.1 { A } A
```

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

`\medium` *arg* (markup)
Switch to medium font-series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

**some bold text** medium font series **bold again**

`\normal-size-sub` *arg* (markup)
Set *arg* in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default $_{\text{subscript in standard size}}$

Used properties:

- `font-size` (0)

`\normal-size-super` *arg* (markup)
Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default $^{\text{superscript in standard size}}$

Used properties:

- `font-size` (0)

`\normal-text` *arg* (markup)

Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
    \hspace #2
    as before
  }
}
```

**HUGE BOLD SANS CAPS** huge normal **AS BEFORE**

`\normalsize` *arg* (markup)

Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

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.,**

`\overtie` *arg* (markup)

Overtie *arg*.

```
\markup \line {
  \overtie "overtied"
  \override #'((offset . 5) (thickness . 1))
```

```
  \overtie "overtied"
  \override #'((offset . 1) (thickness . 5))
  \overtie "overtied"
}
```




Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\replace` *replacements* (list) *arg* (markup)

Used to automatically replace a string by another in the markup *arg*. Each pair of the alist *replacements* specifies what should be replaced. The `key` is the string to be replaced by the `value` string.

```
\markup \replace #'(("thx" . "Thanks!")) thx
```

Thanks!

Used properties:

- `replacement-alist`

`\roman` *arg* (markup)

Set font family to `roman`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}
```

**sans 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 {
    large text
    \hspace #2
    \smaller { smaller text }
    \hspace #2
    large text
  }
}
```

large text smaller text large text

`\sub` *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

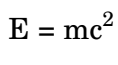


Used properties:

- `font-size` (0)

`\super` *arg* (markup)
Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

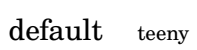


Used properties:

- `font-size` (0)

`\teeny` *arg* (markup)
Set font size to -3.

```
\markup {
  default
  \hspace #2
  \teeny
  teeny
}
```



`\text` *arg* (markup)
Use a text font instead of music symbol or music alphabet font.

```
\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
```

```
    5
  }
}
```

**1, 2,** three, four, **5**

`\tie` *arg* (markup)
: Adds a horizontal bow created with `make-tie-stencil` at bottom or top of *arg*. Looks at `thickness` to determine line thickness, and `offset` to determine y-offset. The added bow fits the extent of *arg*, `shorten-pair` may be used to modify this. *direction* may be set using an `override` or direction-modifiers or `voiceOne`, etc.

```
\markup {
  \override #'(direction . 1)
  \tie "above"
  \override #'(direction . -1)
  \tie "below"
}
```

above below

Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\tiny` *arg* (markup)
: Set font size to -2.

```
\markup {
  default
  \hspace #2
  \tiny
  tiny
}
```

default tiny

`\typewriter` *arg* (markup)
: Use `font-family` typewriter for *arg*.

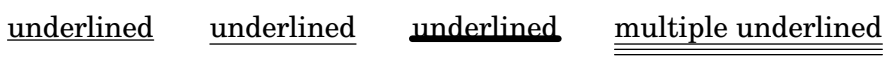
```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

default `typewriter`

`\underline` *arg* (markup)
: Underline *arg*. Looks at `thickness` to determine line thickness, `offset` to determine line y-offset from *arg* and `underline-skip` to determine the distance of additional

lines from the others. `underline-shift` is used to get subsequent calls correct. Overriding it makes little sense, it would end up adding the provided value to the one of `offset`.

```
\markup \justify-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(underline-skip . 4)
  \underline \underline \underline "multiple underlined"
}
```




Used properties:

- `underline-skip` (2)
- `underline-shift` (0)
- `offset` (2)
- `thickness` (1)

`\undertie` *arg* (markup)

```
\markup \line {
  \undertie "undertied"
  \override #'((offset . 5) (thickness . 1))
  \undertie "undertied"
  \override #'((offset . 1) (thickness . 5))
  \undertie "undertied"
}
```




Used properties:

- `shorten-pair` ((0 . 0))
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\upright` *arg* (markup)

Set `font-shape` to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
```

```
    }
    \hspace #2
    italic again
  }
}
```

*italic text* upright text *italic again*

### A.11.2 Align

`\center-align` *arg* (markup)
Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```

one
two
three

`\center-column` *args* (markup list)
Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\column` *args* (markup list)
Stack the markups in *args* vertically. The property `baseline-skip` determines the space between markups in *args*.

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)
: Print two markups on top of each other.

  Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; for this purpose use `\overlay` instead.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```



`\concat` *args* (markup list)
: Concatenate *args* in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

```
\markup {
  \concat {
    one
    two
    three
  }
}
```

onetwothree

`\dir-column` *args* (markup list)
: Make a column of *args*, going up or down, depending on the setting of the `direction` layout property.

```
\markup {
  \override #`(direction . ,UP)
  \dir-column {
    going up
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #'(direction . 1)
  \dir-column {
    going up
  }
}
```

up up
going going going
down

Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *args* (markup list)

Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```
\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
    \null
    \fill-line {
      \line { Text markups }
      \line {
        \italic { evenly spaced }
      }
      \line { across the page }
    }
    \null
    \override #'(line-width . 50)
    \fill-line {
      Width explicitly specified
    }
  }
}
```

Words evenly spaced across the page

Text markups *evenly spaced* across the page

Width explicitly specified

Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* in a horizontal line of width `line-width` with a line of markups *pattern* in between. Patterns are spaced apart by *space*. Patterns are aligned to the *dir* markup.

```
\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
```

```
  "center-aligned :"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned :"
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left first
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left second
}
```

right-aligned :
first . . . . . right
second . . . . . right

center-aligned :
left - - - - - - - - - - - - - - - - - - - - - - - - - - - - - right

left-aligned :
left : : : : : : : : : : : : : : : : first
left : : : : : : : : : : : : : : : : second

Used properties:

- `line-width`
- `word-space`

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)
Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
    \line {
      one
      \general-align #Y #UP
      two
      three
    }
    \null
    \line {
      one
      \general-align #Y #3.2
      two
      three
```

```
    }
  }
}
```

one
two
three

one
two
three

one two three

one three
two

`\halign` *dir* (number) *arg* (markup)
: Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values in between interpolate alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
    two
    three
  }
}
```

one
two
three

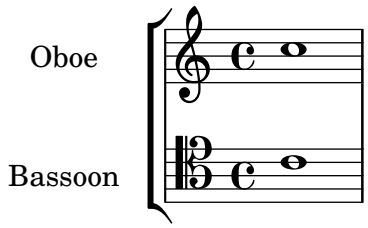
one
two
three

one
two
three

one
two
three

`\hcenter-in` *length* (number) *arg* (markup)
Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
  }
>>
```




`\hspace` *amount* (number)
Create an invisible object taking up horizontal space *amount*.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

`\justify-field` *symbol* (symbol)
Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify-line` *args* (markup list)
Put *markups* in a horizontal line of width *line-width*. The markups are spread to fill the entire line and separated by equal space. If there are no arguments, return an empty stencil.

```
\markup {
  \justify-line {
    Constant space between neighboring words
  }
}
```

Constant space between neighboring words

Used properties:

- `line-width` (#f)
- `word-space` (0.6)

- `text-direction` (1)

`\justify` *args* (markup list)
: Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\justify-string` *arg* (string)
: Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut
      labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident, sunt
      in culpa qui officia deserunt mollit anim id est
      laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align` *arg* (markup)
Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

one
two
three

`\left-column` *args* (markup list)
Put `args` in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (markup list)
Put *args* in a horizontal line. The property `word-space` determines the space between markups in *args*.

```
\markup {
```

```
  \line {
    one two three
  }
}
```

one two three

Used properties:

- `text-direction` (1)
- `word-space`

`\lower` *amount* (number) *arg* (markup)
: Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```

one three
two

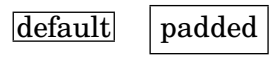
`\overlay` *args* (markup list)
: Takes a list of markups combining them.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \overlay {
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
    \translate #'(0 . 4)\arrow-head #Y #UP ##f
  }
}
```



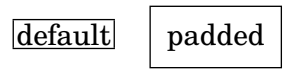
`\pad-around` *amount* (number) *arg* (markup)
: Add padding *amount* all around *arg*.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```



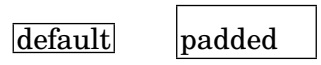

`\pad-markup` *amount* (number) *arg* (markup)
Add space around a markup object. Identical to `pad-around`.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-markup #1 {
      padded
    }
  }
}
```



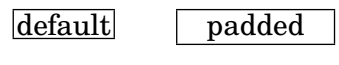

`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```




`\pad-x` *amount* (number) *arg* (markup)
Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```




`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)

Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

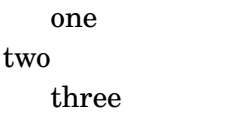
```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```




`\right-align` *arg* (markup)

Align *arg* on its right edge.

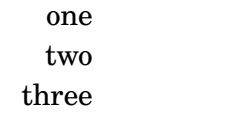
```
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
```




`\right-column` *args* (markup list)

Put `args` in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

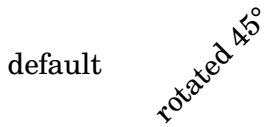



Used properties:

- `baseline-skip`

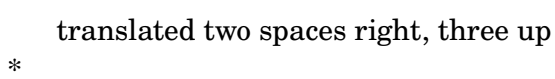
`\rotate` *ang* (number) *arg* (markup)
: Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45˚
  }
}
```



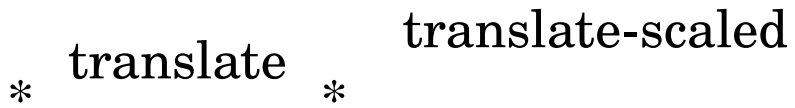

`\translate` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* relative to its surroundings. *offset* is a pair of numbers representing the displacement in the X and Y axis.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```




`\translate-scaled` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```




Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)
: Align `arg` to its Y center.

```
\markup {
  one
  \vcenter
  two
  three
}
```

one $_{\text{two}}$ three

`\vspace` *amount* (number)
Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
    \center-column {
    one
    \vspace #2
    two
    \vspace #5
    three
  }
}
```

one

two

three

`\wordwrap-field` *symbol* (symbol)
Wordwrap the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut
    labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris nisi
    ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\wordwrap` *args* (markup list)

Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where $X$ is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet,
      consectetur adipisicing elit, sed do eiusmod tempor
      incididunt ut labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident,
      sunt in culpa qui officia deserunt mollit anim id
      est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et
dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

### A.11.3 Graphic

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
```



`\beam` *width* (number) *slope* (number) *thickness* (number)

Create a beam with the specified parameters.

```
\markup {
  \beam #5 #1 #2
}
```



`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note {2.} #UP
  }
```

```
}
```



`\circle` *arg* (markup)

Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```



Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle` *radius* (number) *thickness* (number) *filled* (boolean)

A circle of radius *radius* and thickness *thickness*, optionally filled.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```

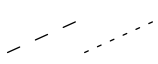


`\draw-dashed-line` *dest* (pair of numbers)

A dashed line.

If `full-length` is set to #t (default) the dashed-line extends to the whole length given by *dest*, without white space at beginning or end. `off` will then be altered to fit. To insist on the given (or default) values of `on`, `off` use `\override #'(full-length . #f)` Manual settings for `on`,`off` and `phase` are possible.

```
\markup {
  \draw-dashed-line #'(5.1 . 2.3)
  \override #'((on . 0.3) (off . 0.5))
  \draw-dashed-line #'(5.1 . 2.3)
}
```



Used properties:

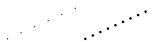
- `full-length` (#t)
- `phase` (0)
- `off` (1)
- `on` (1)

- `thickness` (1)

`\draw-dotted-line` *dest* (pair of numbers)
A dotted line.

The dotted-line always extends to the whole length given by *dest*, without white space at beginning or end. Manual settings for `off` are possible to get larger or smaller space between the dots. The given (or default) value of `off` will be altered to fit the line-length.

```
\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'((thickness . 2) (off . 0.2))
  \draw-dotted-line #'(5.1 . 2.3)
}
```

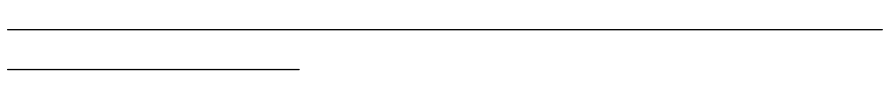


Used properties:

- `phase` (0)
- `off` (1)
- `thickness` (1)

`\draw-hline`
Draws a line across a page, where the property `span-factor` controls what fraction of the page is taken up.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```

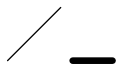


Used properties:

- `span-factor` (1)
- `line-width`
- `draw-line-markup`

`\draw-line` *dest* (pair of numbers)
A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



Used properties:

- `thickness` (1)

`\draw-squiggle-line` *sq-length* (number) *dest* (pair of numbers) *eq-end?* (boolean)
A squiggled line.

If `eq-end?` is set to `#t`, it is ensured the squiggled line ends with a bow in same direction as the starting one. `sq-length` is the length of the first bow. `dest` is the end point of the squiggled line. To match `dest` the squiggled line is scaled accordingly. Its appearance may be customized by overrides for `thickness`, `angularity`, `height` and `orientation`.

```
\markup
  \column {
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(orientation . -1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \draw-squiggle-line #0.5 #'(6 . 0) ##f
    \override #'(height . 1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(thickness . 5)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(angularity . 2)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
  }
```



Used properties:

- `orientation` (1)
- `height` (0.5)
- `angularity` (0)
- `thickness` (0.5)

`\ellipse` *arg* (markup)
Draw an ellipse around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \ellipse {
    Hi
  }
}
```



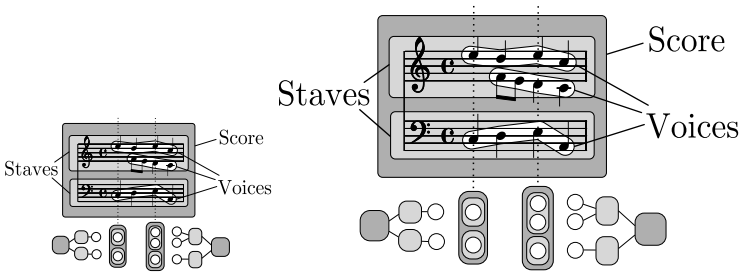

Used properties:

- `y-padding` (0.2)
- `x-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an EPS image. The image is scaled along *axis* to *size*.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```




`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)

Draw a box with rounded corners of dimensions *xext* and *yext*. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```



`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```

one two three

`\oval` *arg* (markup)

Draw an oval around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
```

```
  \oval {
    Hi
  }
}
```



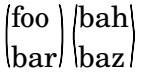

Used properties:

- `y-padding` (0.75)
- `x-padding` (0.75)
- `font-size` (0)
- `thickness` (1)

`\parenthesize` *arg* (markup)

Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \parenthesize
  \column {
    foo
    bar
  }
  \override #'(angularity . 2)
  \parenthesize
  \column {
    bah
    baz
  }
}
```




Used properties:

- `width` (0.25)
- `line-thickness` (0.1)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line *thickness* according to the directions given in *commands*. *commands* is a list of lists where the `car` of each sublist is a drawing command and the `cdr` comprises the associated arguments for each command.

There are seven commands available to use in the list `commands`: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin with *r* are the relative variants of the other three commands.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments; they are the X and Y coordinates for the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands *must* begin with a `moveto` or `rmoveto` to work with the SVG output.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are `'butt`, `'round`, and `'square`. Available line-join styles are `'miter`, `'round`, and `'bevel`.

The property `filled` specifies whether or not the path is filled with color.

```
samplePath =
  #'((moveto 0 0)
     (lineto -1 1)
     (lineto 1 1)
     (lineto 1 -1)
     (curveto -5 -5 -5 5 -1 0)
     (closepath))

\markup {
  \path #0.25 #samplePath

  \override #'(line-join-style . miter)
  \path #0.25 #samplePath

  \override #'(filled . #t)
  \path #0.25 #samplePath
}
```



Used properties:

- `filled` (#f)
- `line-join-style` (round)
- `line-cap-style` (round)

`\postscript` *str* (string)
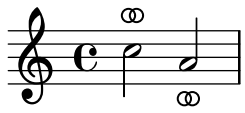
This inserts *str* directly into the output as a PostScript command string.

```
ringsps = #"
  0.15 setlinewidth
  0.9 0.6 moveto
  0.4 0.6 0.5 0 361 arc
  stroke
  1.0 0.6 0.5 0 361 arc
  stroke
  "

rings = \markup {
```

```
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}
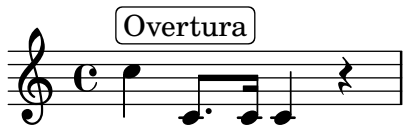
\relative c'' {
  c2^\rings
  a2_\rings
}
```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup; the `corner-radius` property makes it possible to define another shape for the corners (default is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```




Used properties:

- `box-padding` (0.5)
- `font-size` (0)
- `corner-radius` (1)
- `thickness` (1)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*. *factor-pair* is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

```
\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
    mirrored
  }
}
```

**stretched** mirrored

`\triangle` *filled* (boolean)

A triangle, either filled or empty.

```
\markup {
```

```
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```

▲ △

Used properties:

- `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 #"https://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

### A.11.4 Music

`\compound-meter` *time-sig* (number or pair)
Draw a numeric time signature.

```
\markup {
  \column {
    \line { Single number:
            \compound-meter #3 }
    \line { Conventional:
            \compound-meter #'(4 . 4) or
            \compound-meter #'(4 4) }
    \line { Compound:
            \compound-meter #'(2 3 8) }
    \line { Single-number compound:
            \compound-meter #'((2) (3)) }
    \line { Complex compound:
            \compound-meter #'((2 3 8) (3 4)) }
  }
}
```

Single number: $\mathbf{3}$
Conventional: $\frac{4}{4}$ or $\frac{4}{4}$
Compound: $\frac{2+3}{8}$
Single-number compound: $2+3$
Complex compound: $\frac{2+3}{8}+\frac{3}{4}$

`\customTabClef` *num-strings* (integer) *staff-space* (number)
Draw a tab clef sans-serif style.

`\doubleflat`

Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```



`\doublesharp`

Draw a double sharp symbol.

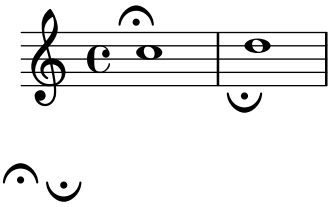
```
\markup {
  \doublesharp
}
```



`\fermata` Create a fermata glyph. When *direction* is `DOWN`, use an inverted glyph. Note that within music, one would usually use the `\fermata` articulation instead of a markup.

```
{ c''1^\markup \fermata d''1_\markup \fermata }

\markup { \fermata \override #`(direction . ,DOWN) \fermata }
```



Used properties:

- `direction` (1)

`\flat` Draw a flat symbol.

```
\markup {
  \flat
}
```



`\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 Sezione "The Emmentaler font" in *Guida alla Notazione* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```



`\natural` Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem and flag. By using fractional values for *dir*, longer or shorter stems can be obtained. Supports all note-head-styles. Ancient note-head-styles will get mensural-style-flags. `flag-style` may be overridden independently. Supported flag-styles are `default`, `old-straight-flag`, `modern-straight-flag`, `flat-flag`, `mensural` and `neomensural`. The latter two flag-styles will both result in mensural-flags. Both are supplied for convenience.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- `style` ('())
- `flag-style` ('())
- `font-size` (0)

`\note` *duration* (duration) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note {4.} #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross)
  \note {4..} #UP
  \hspace #2
  \note {\breve} #0
}
```



Used properties:

- `style` ('())
- `flag-style` ('())
- `font-size` (0)

`\rest-by-number` *log* (number) *dot-count* (number)

A rest or multi-measure-rest symbol.

```
\markup {
  \rest-by-number #3 #2
```

```
  \hspace #2
  \rest-by-number #0 #1
  \hspace #2
  \override #'(multi-measure-rest . #t)
  \rest-by-number #0 #0
}
```



Used properties:

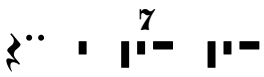
- `multi-measure-rest` (#f)
- `style` ('())
- `font-size` (0)

`\rest` *duration* (string)

This produces a rest, with the *duration* for the rest type and augmentation dots. `"breve"`, `"longa"` and `"maxima"` are valid input-strings.

Printing MultiMeasureRests could be enabled with `\override #'(multi-measure-rest . #t)` If MultiMeasureRests are taken, the MultiMeasureRestNumber is printed above. This is enabled for all styles using default-glyphs. Could be disabled with `\override #'(multi-measure-rest-number . #f)`

```
\markup {
  \rest #"4.."
  \hspace #2
  \rest #"breve"
  \hspace #2
  \override-lines #'(multi-measure-rest . #t)
  {
  \rest #"7"
  \hspace #2
  \override #'(multi-measure-rest-number . #f)
  \rest #"7"
  }
}
```



Used properties:

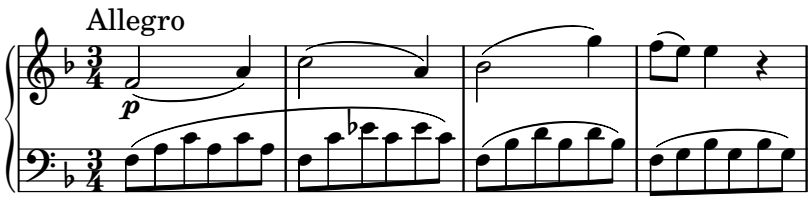
- `word-space` (0.6)
- `multi-measure-rest-number` (#t)
- `multi-measure-rest` (#f)
- `style` ('())

`\score` *score* (score)

Inline an image of music. The reference point (usually the middle staff line) of the lowest staff in the top system is placed on the baseline.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
```

```
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
        f,( bes d bes d bes)
        f( g bes g bes g)
      }
    >>
    \layout {
      indent = 0.0\cm
      \context {
        \Score
        \override RehearsalMark.break-align-symbols =
           #'(time-signature key-signature)
        \override RehearsalMark.self-alignment-X = #LEFT
      }
      \context {
        \Staff
        \override TimeSignature
                   .break-align-anchor-alignment = #LEFT
      }
    }
  }
}
```




Used properties:

- `baseline-skip`

`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```



`\semisharp`
Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```



`\sesquiflat`
Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```



`\sesquisharp`
Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```



`\sharp` Draw a sharp symbol.

```
\markup {
  \sharp
}
```



`\tied-lyric` *str* (string)
Like simple-markup, but use tie characters for '~' tilde symbols.

```
\markup \column {
  \tied-lyric
    #"Siam navi~all'onde~algenti Lasciate~in abbandono"
  \tied-lyric
    #"Impetuosi venti I nostri~affetti sono"
  \tied-lyric
    #"Ogni diletto~e scoglio Tutta la vita~e~un mar."
}
```

Siam navi‿all'onde‿algenti Lasciate‿in abbandono
Impetuosi venti I nostri‿affetti sono
Ogni diletto‿e scoglio Tutta la vita‿e‿un mar.

Used properties:

- `word-space`

### A.11.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)

Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t:`*number* – Set the line thickness (relative to normal line thickness). Default: 0.5.
  - `h:`*number* – Set the height of the diagram in frets. Default: 4.
  - `w:`*number* – Set the width of the diagram in strings. Default: 6.
  - `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
  - `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
  - `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
  - `c:`*string1*`-`*string2*`-`*fret* – Include a barre mark from *string1* to *string2* on *fret*.
  - *string*`-`*fret* – Place a dot on *string* at *fret*. If *fret* is '`o`', *string* is identified as open. If *fret* is '`x`', *string* is identified as muted.
  - *string*`-`*fret*`-`*fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.
- Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.

- Fingerings are given by following the fret number with a `-`, followed by the finger indicator, e.g. '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
  #'((mute 6) (mute 5) (open 4)
     (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute` *string-number*`)`

Place a small 'x' at the top of string *string-number*.

`(open` *string-number*`)`

Place a small 'o' at the top of string *string-number*.

`(barre` *start-string end-string fret-number*`)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo` *fret-number*`)`

Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret` *string-number fret-number* `[`*finger-value*`] [`*color-modifier*`] [`*color*`] ['parenthesized ['default-paren-color]])`

Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, an optional color modifier *color-modifier*, an optional color *color*, an optional parenthesis `'parenthesized` and an optional paranthesis color `'default-paren-color`. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable *dot-color* or for a single dot by setting the value of *color*. The dot can be parenthesized by adding `'parenthesized`. By default the color for the parenthesis is taken from the dot. Adding `'default-paren-color` will take the parenthesis-color from the global *dot-color*, as a fall-back black will be used. Setting *color-modifier* to `inverted` inverts the dot color for a specific fingering. The values for *string-number*, *fret-number*, and the optional *finger* should be entered first in that order. The order of the other optional arguments does not matter. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\harp-pedal` *definition-string* (string)

Make a harp pedal diagram.

Possible elements in *definition-string*:

| | |
|---|---|
| `^` | pedal is up |
| `-` | pedal is neutral |
| `v` | pedal is down |
| `\|` | vertical divider line |
| `o` | the following pedal should be circled (indicating a change) |

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (`\override Voice.TextScript.size = #0.3`) for the overall, the thickness property (`\override Voice.TextScript.thickness = #3`) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (`\override Voice.TextScript.harp-pedal-details.box-width = #1`). It contains the following settings: `box-offset` (vertical shift of the box center for up/down pedals), `box-width`, `box-height`, `space-before-divider` (the spacing between two boxes before the divider) and `space-after-divider` (box spacing after the divider).

```
\markup \harp-pedal #"^-v|--ov^"
```



Used properties:

- `thickness` (0.5)
- `harp-pedal-details` ('())
- `size` (1.2)

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)

Make a woodwind-instrument diagram. For example, say

```
\markup \woodwind-diagram
  #'oboe #'((lh . (d ees)) (cc . (five3qT1q)) (rh . (gis)))
```

for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:

– piccolo

– flute
– oboe
– clarinet
– bass-clarinet
– saxophone
– bassoon
– contrabassoon

To see all of the callable keys for a given instrument, include the function `(print-keys 'instrument)` in your .ly file, where instrument is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

– 1q (1/4 covered)
– 1h (1/2 covered)
– 3q (3/4 covered)
– R (ring depressed)
– F (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between R and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke `(print-keys-verbose 'instrument)`.

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

```
\markup \woodwind-diagram #'oboe #'()
```

Used properties:

- `graphical` (#t)
- `thickness` (0.1)
- `size` (1)

### A.11.6 Accordion Registers

`\discant` *name* (string)

`\discant` *name* generates a discant accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

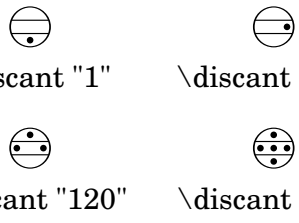
is required near the top of your input file.

The register names in the default `\discant` register set have modeled after numeric Swiss notation like depicted in `http://de.wikipedia.org/wiki/Register_%28Akkordeon%29`, omitting the slashes and dropping leading zeros.

The string *name* is basically a three-digit number with the lowest digit specifying the number of 16' reeds, the tens the number of 8' reeds, and the hundreds specifying the number of 4' reeds. Without modification, the specified number of reeds in 8' is centered in the symbol. Newer instruments may have registrations where 8' can be used either within or without a tone chamber, 'cassotto'. Notationally, the central dot then indicates use of cassotto. One can suffix the tens' digits '`1`' and '`2`' with

'+' or '-' to indicate clustering the dots at the right or left respectively rather than centered.

Some examples are




Used properties:

- `font-size` (0)

`\freeBass` *name* (string)

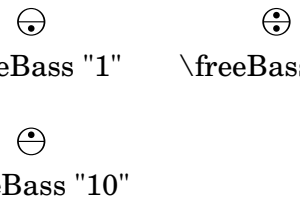
`\freeBass` ***name*** generates a free bass/converter accordion register symbol for the usual two-reed layout.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

Available registrations are




Used properties:

- `font-size` (0)

`\stdBass` *name* (string)

`\stdBass` ***name*** generates a standard bass accordion register symbol.

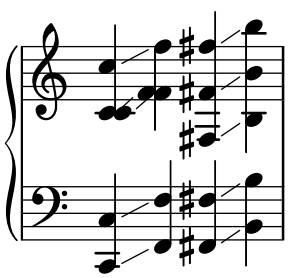
To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The default bass register definitions have been modeled after the article `http://www.accordions.com/index/art/stradella.shtml` originally appearing in Accord Magazine.

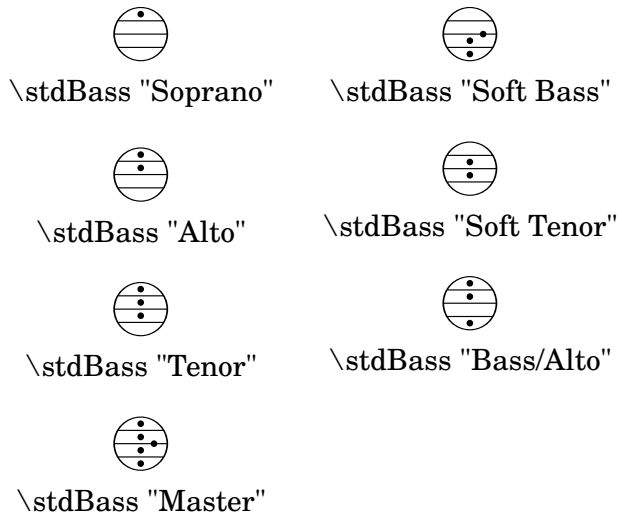
The underlying register model is



This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five reed definitions makes more sense than using a four reed layout: in that manner, the '`Master`' register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are




Used properties:

- `font-size` (0)

`\stdBassIV` *name* (string)
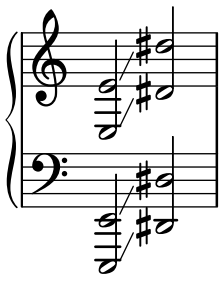
`\stdBassIV` *`name`* generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

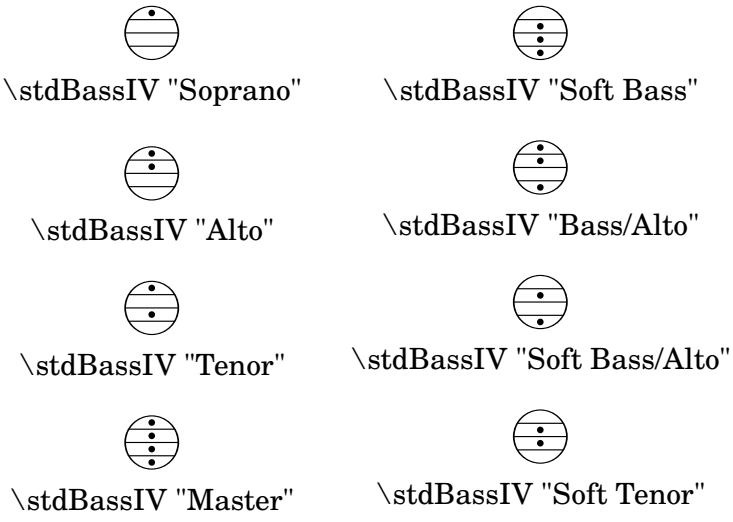
The main use is for four-reed standard bass instruments with reedbank layout



Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches '`Tenor`', '`Master`', '`Soft Bass`'. Note that the sound is quite darker than the same registrations of '`c,`'-based instruments.

Available registrations are




Used properties:

- `font-size` (0)

`\stdBassV` *name* (string)
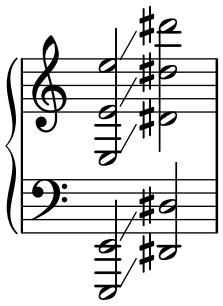
`\stdBassV` *`name`* generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout



This tends to be the bass layout for Hohner's Morino series without convertor or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. The Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional '`Solo Bass`' setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are



\stdBassV "Bass/Alto"



\stdBassV "Soft Bass"



\stdBassV "Soft Bass/Alto"



\stdBassV "Soft Tenor"



\stdBassV "Alto"



\stdBassV "Soprano"



\stdBassV "Tenor"



\stdBassV "Sopranos"



\stdBassV "Master"



\stdBassV "Solo Bass"

Used properties:

- `font-size` (0)

`\stdBassVI` *name* (string)
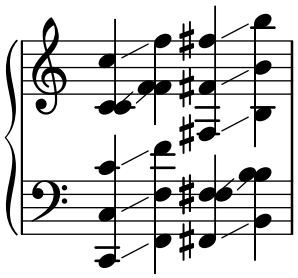
`\stdBassVI` ***name*** generates a standard bass accordion register symbol for six reed basses.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

This is primarily the register layout for the Hohner "Gola" model. The layout is



The registers are effectively quite similar to that of `\stdBass`. An additional bass reed at alto pitch is omitted for esthetical reasons from the '`Master`' setting, so the symbols are almost the same except for the '`Alto/Soprano`' register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are



\stdBassVI "Soprano"



\stdBassVI "Alto/Soprano"



\stdBassVI "Alto"



\stdBassVI "Bass/Alto"



\stdBassVI "Soft Tenor"



\stdBassVI "Soft Bass"



\stdBassVI "Master"

Used properties:

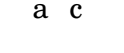
- `font-size` (0)

### A.11.7 Other

`\auto-footnote` *mkup* (markup) *note* (markup)

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```



The footnote will be annotated automatically.

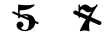
Used properties:

- `padding` (0.0)
- `raise` (0.5)

`\backslashed-digit` *num* (integer)

A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\char` *num* (integer)

Produce a single character. Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
```

```
  \char #65 \char ##x00a9
}
```

A ©

`\eyeglasses`
: Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\first-visible` *args* (markup list)
: Use the first markup in *args* that yields a non-empty stencil and ignore the rest.

```
\markup {
  \first-visible {
    \fromproperty #'header:composer
    \italic Unknown
  }
}
```

*Unknown*

`\footnote` *mkup* (markup) *note* (markup)
: Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

a [1] c [2]

The footnote will not be annotated automatically.

`\fraction` *arg1* (markup) *arg2* (markup)
: Make a fraction of two markups.

```
\markup {
  π ≈
  \fraction 355 113
}
```

$\pi \approx \frac{355}{113}$

Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
: Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
```

```
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

**from *myTitle***

`\left-brace` *size* (number)
: A feta brace in point size *size*.

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```



`\lookup` *glyph-name* (string)
: Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```



`\markalphabet` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

H Z

`\markletter` *num* (integer)
Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

H AA

`\null`
An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (procedure) *arg* (markup)
Apply the *procedure* markup command to *arg*. *procedure* takes the same arguments as `interpret-markup` and returns a stencil.

`\override` *new-prop* (pair) *arg* (markup)
Add the argument *new-prop* to the property list. Properties may be any property supported by Sezione "font-interface" in *Guida al Funzionamento Interno*, Sezione "text-interface" in *Guida al Funzionamento Interno* and Sezione "instrument-specific-markup-interface" in *Guida al Funzionamento Interno*.

*new-prop* may be either a single alist pair, or non-empty alist of its own.

```
\markup {
  \undertie "undertied"
  \override #'(offset . 15)
  \undertie "offset undertied"
  \override #'((offset . 15)(thickness . 3))
  \undertie "offset thick undertied"
}
```

undertied offset undertied offset thick undertied

`\page-link` *page-number* (number) *arg* (markup)
Add a link to the page *page-number* around *arg*. This only works in the PDF backend.

```
\markup {
  \page-link #2  { \italic { This links to page 2... } }
}
```

*This links to page 2...*

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)

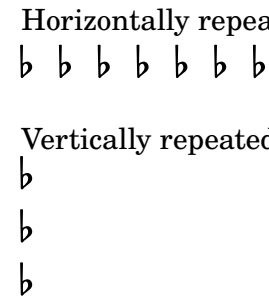
Reference to a page number. *label* is the label set on the referenced page (using `\label` or `\tocItem`), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

(If the current book or bookpart is set to use roman numerals for page numbers, the reference will be formatted accordingly – in which case the *gauge*'s width may require additional tweaking.)

`\pattern` *count* (non-negative integer) *axis* (non-negative integer) *space* (number) *pattern* (markup)

Prints *count* times a *pattern* markup. Patterns are spaced apart by *space* (defined as for `\hspace` or `\vspace`, respectively). Patterns are distributed on *axis*.

```
\markup \column {
  "Horizontally repeated :"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated :"
  \pattern #3 #Y #0.5 \flat
}
```




`\property-recursive` *symbol* (symbol)

Print out a warning when a header field markup contains some recursive markup definition.

`\right-brace` *size* (number)

A feta brace in point size *size*, rotated 180 degrees.

```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```



`\slashed-digit` *num* (integer)

A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
```

```
  \slashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (stencil)
: Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`
: Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)
: Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)
: Read the contents of file *name*, and include it verbatim.

```
\markup {
  \verbatim-file #"simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative {
  c' d e f g a b c
}
%% Optional helper for automatic updating
%% by convert-ly.  May be omitted.
\version "2.19.21"
```

`\whiteout` *arg* (markup)
: Provide a white background for *arg*. The shape of the white background is determined by `style`. The default is `box` which produces a rectangle. `rounded-box` produces a rounded rectangle. `outline` approximates the outline of the markup.

```
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
```

```
    \override #'(thickness . 1.5)
    \whiteout whiteout-box
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'((style . rounded-box) (thickness . 3))
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'((style . outline) (thickness . 3))
    \whiteout whiteout-outline
}
```










Used properties:

- `thickness` ('())
- `style` (box)

`\with-color` *color* (color) *arg* (markup)
Draw *arg* in color specified by *color*.

```
\markup {
  \with-color #red
  red
  \hspace #2
  \with-color #green
  green
  \hspace #2
  \with-color "#0000ff"
  blue
}
```

red green blue

`\with-dimensions-from` *arg1* (markup) *arg2* (markup)
Print *arg2* with the horizontal and vertical dimensions of *arg1*.

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
Set the horizontal and vertical dimensions of *arg* to *x* and *y*.

`\with-link` *label* (symbol) *arg* (markup)
: Add a link to the page holding label *label* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-link #'label {
    \italic { This links to the page
              containing the label... }
  }
}
```

*This links to the page containing the label...*

`\with-outline` *outline* (markup) *arg* (markup)
: Print *arg* with the outline and dimensions of *outline*. The outline is used by skylines to resolve collisions (not for whiteout).

## A.12 Comandi per una lista di *markup*

Tutti i comandi seguenti possono essere usati all'interno di `\markuplist`:

`\column-lines` *args* (markup list)
: Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

  Used properties:

  - `baseline-skip`

`\justified-lines` *args* (markup list)
: Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\map-markup-commands` *compose* (procedure) *args* (markup list)
: This applies the function *compose* to every markup in *args* (including elements of markup list command calls) in order to produce a new markup list. Since the return value from a markup list command call is not a markup list but rather a list of stencils, this requires passing those stencils off as the results of individual markup calls. That way, the results should work out as long as no markups rely on side effects.

`\override-lines` *new-prop* (pair) *args* (markup list)
: Like `\override`, for markup lists.

`\score-lines` *score* (score)
: This is the same as the `\score` markup but delivers its systems as a list of lines. Its *score* argument is entered in braces like it would be for `\score`.

`\table` *column-align* (number list) *lst* (markup list)
: Returns a table.

*column-align* specifies how each column is aligned, possible values are -1, 0, 1. The number of elements in *column-align* determines how many columns will be printed. The entries to print are given by *lst*, a markup-list. If needed, the last row is filled up with `point-stencil`s. Overriding `padding` may be used to increase columns horizontal distance. Overriding `baseline-skip` to increase rows vertical distance.

```
\markuplist {
  \override #'(padding . 2)
  \table
    #'(0 1 0 -1)
    {
      \underline { center-aligned right-aligned
                   center-aligned left-aligned }
      one      \number    1 thousandth \number 0.001
      eleven   \number   11 hundredth  \number 0.01
      twenty   \number   20 tenth      \number 0.1
      thousand \number 1000 one        \number 1.0
    }
}
```

| center-aligned | right-aligned | center-aligned | left-aligned |
|---|---|---|---|
| one | 1 | thousandth | 0.001 |
| eleven | 11 | hundredth | 0.01 |
| twenty | 20 | tenth | 0.1 |
| thousand | 1000 | one | 1.0 |

Used properties:

- `baseline-skip`
- `padding` (0)

`\table-of-contents`

Used properties:

- `baseline-skip`

`\wordwrap-internal` *justify* (boolean) *args* (markup list)

Internal markup list command used to define `\justify` and `\wordwrap`.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)

`\wordwrap-lines` *args* (markup list)

Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where $X$ is the number of staff spaces.

Used properties:

- `text-direction` (1)

- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string-internal` *justify* (boolean) *arg* (string)
Internal markup list command used to define `\justify-string` and `\wordwrap-string`.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`

## A.13 Elenco dei caratteri speciali

Si possono usare i seguenti riferimenti per i caratteri speciali; maggiori informazioni in [Alias ASCII], pagina 520.

Si usa la sintassi HTML. Di questi riferimenti molti sono identici a quelli HTML, alcuni sono ispirati a LaTeX.

I caratteri sono compresi in un riquadro per rendere visibile la loro dimensione. È stato aggiunto un po' di spazio tra il carattere e il riquadro per migliorare la leggibilità.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ¡ | ¡ | ¿ | ¿ | &solidus; | ⁄ | &flq; | ‹ | | |
| &frq; | › | &flqq; | « | &frqq; | » | &glq; | ‚ | | |
| &grq; | ‘ | &glqq; | „ | &grqq; | “ | &elq; | ‘ | | |
| &erq; | ’ | &elqq; | “ | &erqq; | ” |   | | | |
|   | |   | |   | | &nnbsp; | | | |
| ‍ | | ‌ | | · | · | • | • | | |
| &copyright; | © | &registered; | ® | &trademark; | ™ | † | † | | |
| ‡ | ‡ | № | № | ª | ª | º | º | | |
| ¶ | ¶ | § | § | ° | ° | № | № | | |
| ‰ | ‰ | ¦ | ¦ | ´ | ´ | &acutedbl; | ˝ | | |
| ` | ` | ˘ | ˘ | ˇ | ˇ | &cedilla; | ¸ | | |
| &circumflex; | ^ | &diaeresis; | ¨ | &macron; | ¯ | &aa; | å | | |
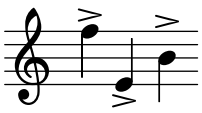
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| &AA; | Å | &ae; | æ | &AE; | Æ | ä | ä | | |
| Ä | Ä | &dh; | ð | &DH; | Ð | &dj; | đ | | |
| &DJ; | Đ | &l; | ł | &L; | Ł | &ng; | ŋ | | |
| &NG; | Ŋ | &o; | ø | &O; | Ø | &oe; | œ | | |
| &OE; | Œ | ö | ö | Ö | Ö | &s; | ſ | | |
| &ss; | ß | &th; | þ | &TH; | Þ | ü | ü | | |
| Ü | Ü | + | + | − | − | × | × | | |
| ÷ | ÷ | ¹ | ¹ | ² | ² | ³ | ³ | | |
| &sqrt; | √ | &increment; | Δ | &infty; | ∞ | ∑ | ∑ | | |
| ± | ± | &bulletop; | ∙ | &partial; | ∂ | &neg; | ¬ | | |
| &currency; | ¤ | $ | $ | € | € | &pounds; | £ | | |
| ¥ | ¥ | ¢ | ¢ | | | | | | |

## A.14 Elenco delle articolazioni

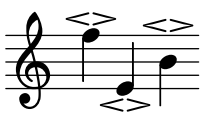
Le liste seguenti mostrano tutti i segni del tipo di carattere Feta che possono essere attaccati alle note (es: '`f\accent`' o '`f->`'). Ogni esempio mostra il segno nelle posizioni *up*, *down* e *neutral*.

### Articolazioni
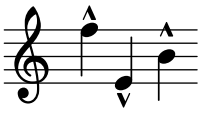
`\accent` or `->`
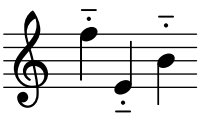


`\espressivo`
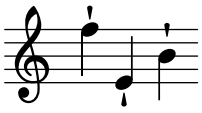


`\marcato` or `-^`
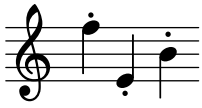


`\portato` or `-_`
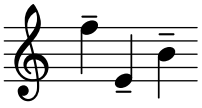


`\staccatissimo` or `-!`



`\staccato` or `-.`
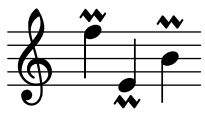


`\tenuto` or `--`
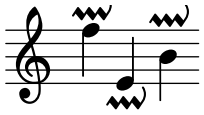


## Ornamenti

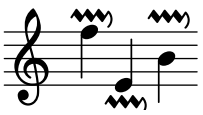
\prall
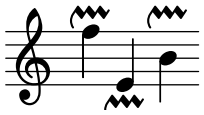


\prallup
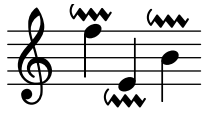


\pralldown
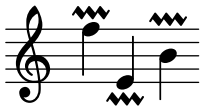


\upprall
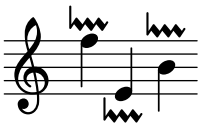


\downprall
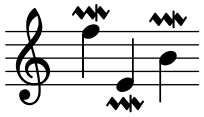


\prallprall



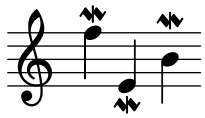
\lineprall
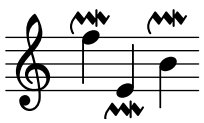


\prallmordent
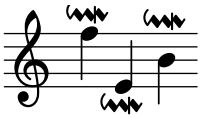


\mordent
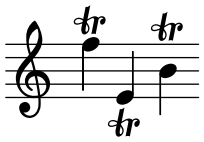


\upmordent
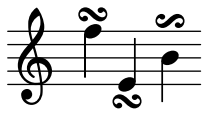


\downmordent
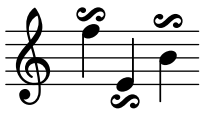


\trill



\turn
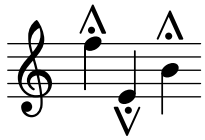


\reverseturn
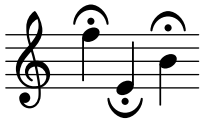


## Punti coronati

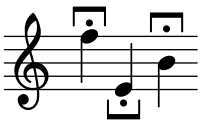
\shortfermata



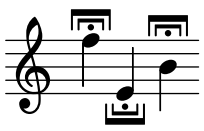
\fermata



\longfermata

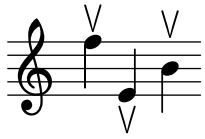


\verylongfermata
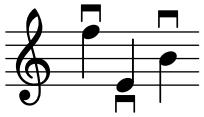


## Segni specifici per strumento

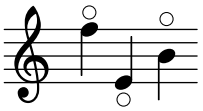
\upbow



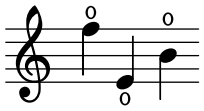
\downbow
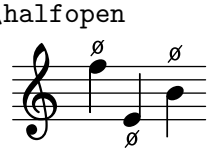


\flageolet



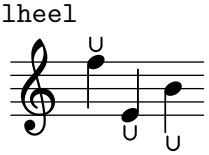
\open



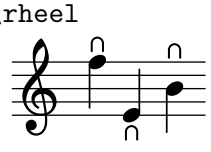
\halfopen



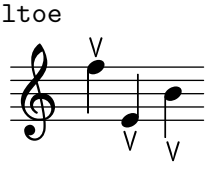
\lheel



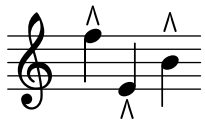
\rheel



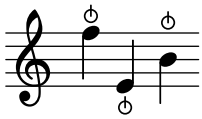
\ltoe



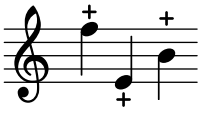
\rtoe



\snappizzicato
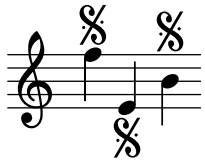


\stopped or -+
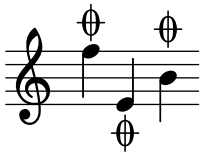


### Segni di ripetizione

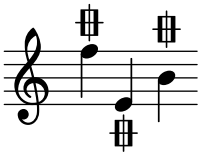
\segno



\coda

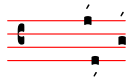


\varcoda

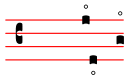


### Segni antichi

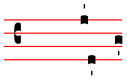
\accentus



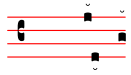
\circulus



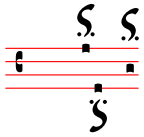
\ictus



\semicirculus
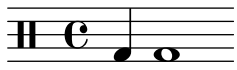


\signumcongruentiae
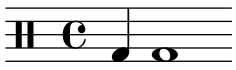


## A.15 Note percussive
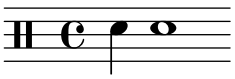
bassdrum
bd
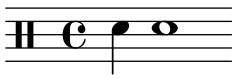


acousticbassdrum
bda
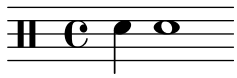


snare
sn
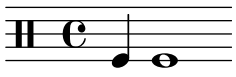


acousticsnare
sna
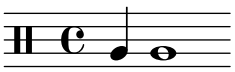


electricsnare
sne
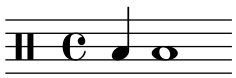


lowfloortom
tomfl
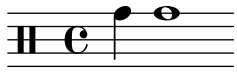


highfloortom
tomfh
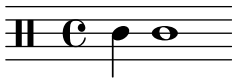


lowtom
toml
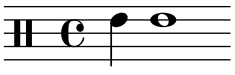


hightom
tomh
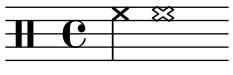


lowmidtom
tomml
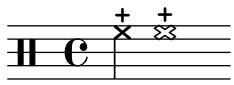


himidtom
tommh
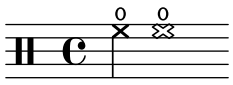


highhat
hh
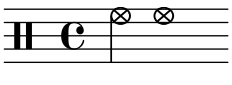


closedhihat
hhc
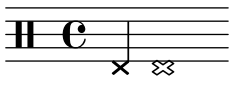


openhighhat
hho
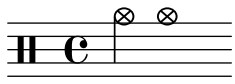


halfopenhihat
hhho
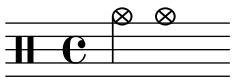


pedalhihat
hhp
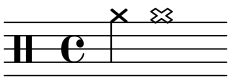


crashcymbal
cymc
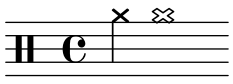


crashcymbala
cymca
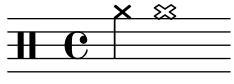


crashcymbalb
cymcb
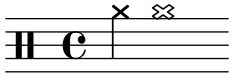


ridecymbal
cymr
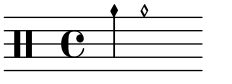


ridecymbala
cymra
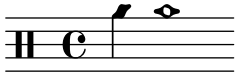


ridecymbalb
cymrb
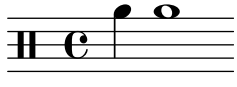


chinesecymbal
cymch
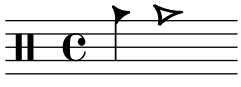


splashcymbal
cyms



ridebell
rb



cowbell
cb



hibongo
boh



openhibongo
boho
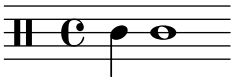


mutehibongo
bohm
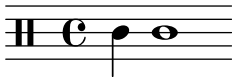


lobongo
bol
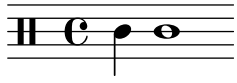


openlobongo
bolo
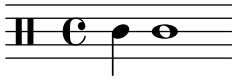


mutelobongo
bolm
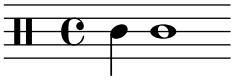


hiconga
cgh
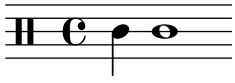


openhiconga
cgho
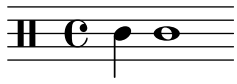


mutehiconga
cghm
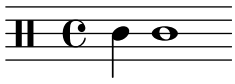


loconga
cgl
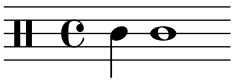


openloconga
cglo
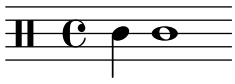


muteloconga
cglm
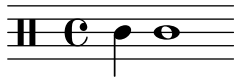


hitimbale
timh
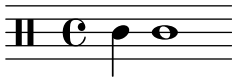


lotimbale
timl
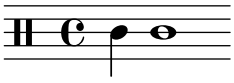


hiagogo
agh
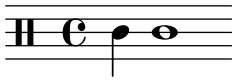


loagogo
agl
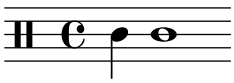


sidestick
ss
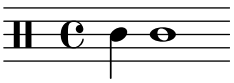


hisidestick
ssh
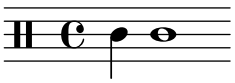


losidestick
ssl
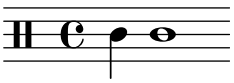


guiro
gui
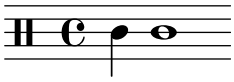


shortguiro
guis
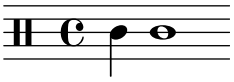


longguiro
guil
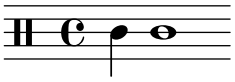


cabasa
cab
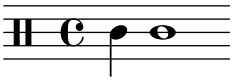


maracas
mar



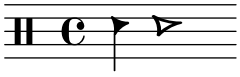
shortwhistle
whs
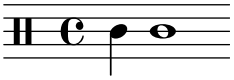


longwhistle
whl
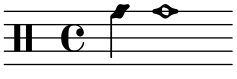


handclap
hc
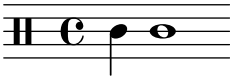


tambourine
tamb
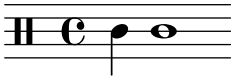


vibraslap
vibs
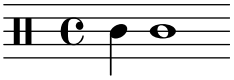


tamtam
tt
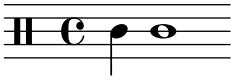


claves
cl
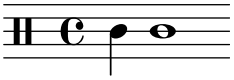


hiwoodblock
wbh
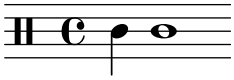


lowoodblock
wbl
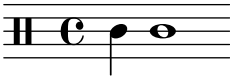


opencuica
cuio
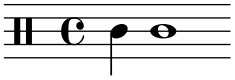


mutecuica
cuim
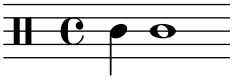


triangle
tri



opentriangle
trio



mutetriangle
trim



oneup
ua
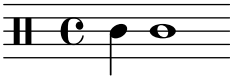


twoup
ub
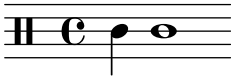


threeup
uc
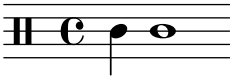


fourup
ud
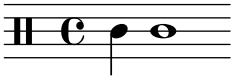


fiveup
ue



onedown
da
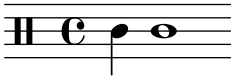


twodown
db
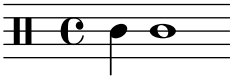


threedown
dc



fourdown
dd



fivedown
de



## A.16 Glossario tecnico

Un glossario dei termini tecnici e dei concetti usati internamente in LilyPond. Questi termini appaiono nei manuali, nelle mailing list e nel codice sorgente.

### alist

Una lista di associazioni – **alist** per *association list* –, è una coppia Scheme che associa un valore a una parola chiave: `(chiave . valore)`. Per esempio, in `scm/lily.scm`, la lista di associazioni "type-p-name-alist" associa alcuni tipi di predicato (come `ly:music?`) ai nomi (come "music") in modo che gli errori relativi al controllo del tipo possano essere segnalati con un messaggio che includa il nome del tipo di predicato atteso.

### callback

**callback** indica una routine, funzione o metodo il cui riferimento sia passato come argomento quando si richiama un'altra routine, permettendo così alla routine richiamata di invocarla. La tecnica fa sì che a un livello più basso del software si possa richiamare una funzione definita a un livello più alto. I "callback" sono molto utilizzati in LilyPond per far sì che il codice Scheme a livello utente possa definire quante azioni di basso livello sono eseguite.

### closure

In Scheme, si parla di **closure** (chiusura) quando una funzione, di solito un'espressione lambda, viene passata come variabile. La chiusura contiene il codice della funzione più i riferimenti ai collegamenti lessicali delle variabili libere della funzione (ovvero quelle variabili usate nell'espressione ma definite al di fuori di essa). Quando questa funzione viene applicata a diversi argomenti successivamente, i collegamenti delle variabili libere che sono stati catturati nella chiusura vengono usati per ottenere i valori delle variabili libere da usare nel calcolo. Una caratteristica utile delle chiusure è la conservazione dei valori delle variabili interne tra un'invocazione e l'altra, facendo sì che uno stato possa essere mantenuto.

### glyph

Un **glifo** è una particolare rappresentazione grafica di un carattere tipografico o una combinazione di due caratteri che formano una legatura. Un insieme di glifi con un solo stile e forma costituiscono un tipo di carattere (font), e un insieme di tipi di carattere con vari stili e dimensioni costituiscono una famiglia di caratteri tipografici.

#### Vedi anche

Guida alla notazione: Sezione 1.8.3 [Tipi di carattere], pagina 260, Sezione 3.3.3 [Caratteri speciali], pagina 519.

### grob

Gli oggetti di LilyPond che rappresentano elementi della notazione nell'output – come teste di nota, gambi, legature di portamento e di valore, diteggiature, chiavi, etc. – sono chiamati, in inglese, 'Layout objects' ('Oggetti della formattazione') o anche 'GRaphical OBjects' o **grobs** in forma breve. Sono rappresentati da istanze della classe `Grob`.

#### Vedi anche

Manuale di apprendimento: Sezione "Oggetti e interfacce" in *Manuale di Apprendimento*, Sezione "Convenzioni per i nomi di oggetti e proprietà" in *Manuale di Apprendimento*, Sezione "Proprietà degli oggetti di formattazione" in *Manuale di Apprendimento*.

Guida al funzionamento interno: Sezione "grob-interface" in *Guida al Funzionamento Interno*, Sezione "All layout objects" in *Guida al Funzionamento Interno*.

### immutable

Un oggetto si dice **immutabile** – in inglese *immutable* – se il suo stato non può essere modificato dopo la sua creazione, in contrasto con un oggetto variabile, che può essere modificato dopo la sua creazione.

In LilyPond, le proprietà immutabili o condivise definiscono lo stile e il comportamento predefinito dei grob. Sono condivise tra molti oggetti. In apparente contraddizione col loro nome, possono essere modificate con `\override` e `\revert`.

### Vedi anche

Guida alla notazione: [mutable], pagina 770.

### interface

Le azioni e le proprietà comuni a un insieme di grob sono raggruppate in un oggetto chiamato `grob-interface` o semplicemente 'interface'.

### Vedi anche

Manuale di apprendimento: Sezione "Oggetti e interfacce" in *Manuale di Apprendimento*, Sezione "Convenzioni per i nomi di oggetti e proprietà" in *Manuale di Apprendimento*, Sezione "Proprietà presenti nelle interfacce" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.2.2 [Interfacce di formattazione], pagina 612.

Guida al funzionamento interno: Sezione "Graphical Object Interfaces" in *Guida al Funzionamento Interno*.

### 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], pagina 770. 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

Si dice che un oggetto è **variabile** – *mutable* in inglese – se il suo stato può essere modificato dopo la sua creazione, in contrasto con un oggetto immutevole, il cui stato viene fissato al momento della sua creazione.

In LilyPond, le proprietà variabili contengono valori specifici di un grob. Di solito, le liste di altri oggetti o i risultati di calcoli sono salvati in proprietà variabili.

### Vedi anche

Guida alla notazione: [immutable], pagina 770.

### output-def

Un'istanza della classe `Output-def` contiene i metodi e le strutture dei dati associate con un blocco di output. Tali istanze vengono create per i blocchi midi, layout e paper.

### parser

Un **analizzatore sintattico** – in inglese *parser* – analizza la sequenza di *token* prodotti da un *lexer* per determinare la sua struttura grammaticale, raggruppando i token progressivamente in gruppi più ampi in base a certe regole grammaticali. Se la sequenza di token è valida, il risultato

finale è l'insieme dei token ordinati a albero, la cui radice è il simbolo iniziale della grammatica. Se ciò non può essere ottenuto, il file non è valido e viene generato un appropriato messaggio di errore. I gruppi sintattici e le regole che li definiscono nella sintassi di LilyPond sono definiti in `lily/parser.yy` e mostrati in Backus Normal Form (BNF) in Sezione "LilyPond grammar" in *Guida del Collaboratore*. Questo file viene usato dal generatore di parser Bison per generare il parser durante la compilazione del programma. Fa parte del codice sorgente e non è incluso nell'installazione binaria di LilyPond.

## parser variable

Si tratta di variabili definite direttamente in Scheme. Il loro uso da parte degli utente è fortemente scoraggiato, perché la semantica del loro raggio d'azione può creare confusione.

Se il valore di una simile variabile viene cambiato in un file `.ly`, la modifica è globale e, se non viene ripristinato esplicitamente, il nuovo valore viene mantenuto fino alla fine del file, agendo su blocchi `\score` successivi così come su file esterni aggiunti col comando `\include`. Ciò può portare a conseguenze non volute e in progetti complessi gli errori conseguenti possono essere difficili da individuare.

LilyPond usa le seguenti variabili dell'analizzatore sintattico:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- partCombineListener
- pitchnames
- toplevel-bookparts
- toplevel-scores
- showLastLength
- showFirstLength

## prob

Le proprietà dell'oggetto – **prob** per PRoperty OBjects – sono istanze della classe `Prob`, una semplice classe per oggetti che hanno liste associative di proprietà variabili e invariabili e metodi per manipolarle. Le classi `Music` e `Stream_event` derivano da `Prob`. Vengono create istanze della classe `Prob` anche per conservare il contenuto formattato dei grob di un sistema e i blocchi dei titoli durante la formattazione della pagina.

## smob

Gli oggetti Scheme – **Smobs** per ScheMe OBjects – fanno parte del meccanismo con cui Guile esporta gli oggetti C e C++ in codice Scheme. In LilyPond, gli smob vengono creati dagli oggetti C++ attraverso delle macro. Esistono due tipi di oggetti smob: smob semplici, intesi per oggetti invariabili semplici come i numeri; e smob complessi, usati per oggetti aventi delle identità. Maggiori informazioni si trovano nei sorgenti di LilyPond e precisamente nel file `lily/includes/smob.hh`.

## stencil

Un'istanza della classe **stencil** contiene l'informazione necessaria per stampare un oggetto tipografico. È un semplice smob che contiene un riquadro che definisce l'estensione verticale e orizzontale dell'oggetto, e un'espressione Scheme che stamperà l'oggetto quando esaminata. Gli

stencil possono essere combinati per formare stencil più complessi, definiti da una gerarchia di espressioni Scheme degli stencil che li compongono.

La proprietà `stencil`, che connette un grob al suo stencil, è definita nell'interfaccia `grob-interface`.

### Vedi anche

Guida al funzionamento interno: Sezione "grob-interface" in *Guida al Funzionamento Interno*.

## A.17 Tutte le proprietà di contesto

`accidentalGrouping` (symbol)
: If set to `'voice`, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`additionalBassStrings` (list)
: The additional tablature bass-strings, which will not get a seprate line in TabStaff. It is a list of the pitches of each string (starting with the lowest numbered one).

`additionalPitchPrefix` (string)
: Text with which to prefix additional pitches within a chord name.

`aDueText` (markup)
: Text to print at a unisono passage.

`alignAboveContext` (string)
: Where to insert newly created context in vertical alignment.

`alignBelowContext` (string)
: Where to insert newly created context in vertical alignment.

`alternativeNumberingStyle` (symbol)
: The style of an alternative's bar numbers. Can be `numbers` for going back to the same number or `numbers-with-letters` for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

`alternativeRestores` (symbol list)
: Timing variables that are restored to their value at the start of the first alternative in subsequent alternatives.

`associatedVoice` (string)
: Name of the context (see `associatedVoiceType` for its type, usually `Voice`) that has the melody for this `Lyrics` line.

`associatedVoiceType` (symbol)
: Type of the context that has the melody for this `Lyrics` line.

`autoAccidentals` (list)
: List of different ways to typeset an accidental.

  For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

  Each entry in the list is either a symbol or a procedure.

  *symbol* The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Sezione "Score" in *Guida al Funzionamento Interno* then all staves share accidentals, and if *context* is Sezione "Staff" in *Guida al Funzionamento Interno* then all voices in the same staff share accidentals, but staves do not.

*procedure* The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

`context` The current context to which the rule should be applied.

`pitch` The pitch of the note to be evaluated.

`barnum` The current bar number.

`measurepos`
The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. `(#t . #f)` does not make sense.

`autoBeamCheck` (procedure)
A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-`#f` return value starts or stops the auto beam.

`autoBeaming` (boolean)
If set to true then beams are generated automatically.

`autoCautionaries` (list)
List similar to `autoAccidentals`, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

`automaticBars` (boolean)
If set to false then bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` keyword, measures are still counted. Bar line generation will resume according to that count if this property is unset.

`barAlways` (boolean)
If set to true a bar line is drawn after each note.

`barCheckSynchronize` (boolean)
If true then reset `measurePosition` when finding a bar check.

`barNumberFormatter` (procedure)
A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

`barNumberVisibility` (procedure)
A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the `break-visibility` property.

The following procedures are predefined:

`all-bar-numbers-visible`
Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

`first-bar-number-invisible`
Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

`first-bar-number-invisible-save-broken-bars`
Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

`first-bar-number-invisible-and-no-parenthesized-bar-numbers`
: Enable bar numbers for all bars except the first bar and broken bars. This is the default.

`(every-nth-bar-number-visible` *n*`)`
: Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

`(modulo-bar-number-visible` *n m*`)`
: If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

`baseMoment` (moment)
: Smallest unit of time that will stand on its own as a subdivided section.

`beamExceptions` (list)
: An alist of exceptions to autobeam rules that normally end on beats.

`beamHalfMeasure` (boolean)
: Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

`beatStructure` (list)
: List of `baseMoment`s that are combined to make beats.

`chordChanges` (boolean)
: Only show changes in chords scheme?

`chordNameExceptions` (list)
: An alist of chord exceptions. Contains `(`*`chord . markup`*`)` entries.

`chordNameFunction` (procedure)
: The function that converts lists of pitches to chord names.

`chordNameLowercaseMinor` (boolean)
: Downcase roots of minor chords?

`chordNameSeparator` (markup)
: The markup object used to separate parts of a chord name.

`chordNoteNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for single pitches.

`chordPrefixSpacer` (number)
: The space added between the root symbol and the prefix of a chord name.

`chordRootNamer` (procedure)
: A function that converts from a pitch object to a text markup. Used for chords.

`clefGlyph` (string)
: Name of the symbol within the music font.

`clefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`clefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`clefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`clefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`completionBusy` (boolean)
: Whether a completion-note head is playing.

`completionFactor` (an exact rational or procedure)
: When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.

  If `#f`, the completion engraver uses the scale-factor of each duration being split.

  If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

`completionUnit` (moment)
: Sub-bar unit of completion.

`connectArpeggios` (boolean)
: If set, connect arpeggios across piano staff.

`countPercentRepeats` (boolean)
: If set, produce counters for percent repeats.

`createKeyOnClefChange` (boolean)
: Print a key signature whenever the clef is changed.

`createSpacing` (boolean)
: Create `StaffSpacing` objects? Should be set for staves.

`crescendoSpanner` (symbol)
: The type of spanner to be used for crescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin crescendo is used.

`crescendoText` (markup)
: The text to print at start of non-hairpin crescendo, i.e., '`cresc.`'.

`cueClefGlyph` (string)
: Name of the symbol within the music font.

`cueClefPosition` (number)
: Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.

`cueClefTransposition` (integer)
: Add this much extra transposition. Values of 7 and -7 are common.

`cueClefTranspositionFormatter` (procedure)
: A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.

`cueClefTranspositionStyle` (symbol)
: Determines the way the ClefModifier grob is displayed. Possible values are '`default`', '`parenthesized`' and '`bracketed`'.

`currentBarNumber` (integer)
: Contains the current barnumber. This property is incremented at every bar line.

`decrescendoSpanner` (symbol)
: The type of spanner to be used for decrescendi. Available values are '`hairpin`' and '`text`'. If unset, a hairpin decrescendo is used.

`decrescendoText` (markup)
: The text to print at start of non-hairpin decrescendo, i.e., '`dim.`'.

`defaultBarType` (string)
: Set the default type of bar line. See `whichBar` for information on available bar types.

  This variable is read by Sezione "Timing_translator" in *Guida al Funzionamento Interno* at Sezione "Score" in *Guida al Funzionamento Interno* level.

`defaultStrings` (list)
: A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.

`doubleRepeatSegnoType` (string)
: Set the default bar line for the combinations double repeat with segno. Default is '`:|.S.|:`'.

`doubleRepeatType` (string)
: Set the default bar line for double repeats.

`doubleSlurs` (boolean)
: If set, two slurs are created for every slurred note, one above and one below the chord.

`drumPitchTable` (hash table)
: A table mapping percussion instruments (symbols) to pitches.

`drumStyleTable` (hash table)
: A hash table which maps drums to layout settings. Predefined values: '`drums-style`', '`agostini-drums-style`', '`timbales-style`', '`congas-style`', '`bongos-style`', and '`percussion-style`'.

  The layout style is a hash table, containing the drum-pitches (e.g., the symbol '`hihat`') as keys, and a list `(`*`notehead-style script vertical-position`*`)` as values.

`endAtSkip` (boolean)
: End `DurationLine` grob on `skip-event`

`endRepeatSegnoType` (string)
: Set the default bar line for the combinations ending of repeat with segno. Default is '`:|.S`'.

`endRepeatType` (string)
: Set the default bar line for the ending of repeats.

`explicitClefVisibility` (vector)
: '`break-visibility`' function for clef changes.

`explicitCueClefVisibility` (vector)
: '`break-visibility`' function for cue clef changes.

`explicitKeySignatureVisibility` (vector)
: '`break-visibility`' function for explicit key changes. '`\override`' of the `break-visibility` property will set the visibility for normal (i.e., at the start of the line) key signatures.

`extendersOverRests` (boolean)
: Whether to continue extenders as they cross a rest.

`extraNatural` (boolean)
: Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

`figuredBassAlterationDirection` (direction)
: Where to put alterations relative to the main figure.

`figuredBassCenterContinuations` (boolean)
: Whether to vertically center pairs of extender lines. This does not work with three or more lines.

`figuredBassFormatter` (procedure)
: A routine generating a markup for a bass figure.

`figuredBassPlusDirection` (direction)
: Where to put plus signs relative to the main figure.

`fingeringOrientations` (list)
: A list of symbols, containing '`left`', '`right`', '`up`' and/or '`down`'. This list determines where fingerings are put relative to the chord being fingered.

`firstClef` (boolean)
: If true, create a new clef when starting a staff.

`followVoice` (boolean)
: If set, note heads are tracked across staff switches by a thin line.

`fontSize` (number)
: The relative size of all grobs in a context.

`forbidBreak` (boolean)
: If set to `#t`, prevent a line break at this point.

`forceClef` (boolean)
: Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.

`fretLabels` (list)
: A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.

`glissandoMap` (list)
: A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '() will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.

`gridInterval` (moment)
: Interval for which to generate `GridPoint`s.

`handleNegativeFrets` (symbol)
: How the automatic fret calculator should handle calculated negative frets. Values include `'ignore`, to leave them out of the diagram completely, `'include`, to include them as calculated, and `'recalculate`, to ignore the specified string and find a string where they will fit with a positive fret number.

`harmonicAccidentals` (boolean)
: If set, harmonic notes in chords get accidentals.

`harmonicDots` (boolean)
: If set, harmonic notes in dotted chords get dots.

`highStringOne` (boolean)
: Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.

`ignoreBarChecks` (boolean)
: Ignore bar checks.

`ignoreBarNumberChecks` (boolean)
: Ignore bar number checks.

`ignoreFiguredBassRest` (boolean)
: Don't swallow rest events.

`ignoreMelismata` (boolean)
: Ignore melismata for this Sezione "Lyrics" in *Guida al Funzionamento Interno* line.

`implicitBassFigures` (list)
: A list of bass figures that are not printed as numbers, but only as extender lines.

`includeGraceNotes` (boolean)
: Do not ignore grace notes for Sezione "Lyrics" in *Guida al Funzionamento Interno*.

`initialTimeSignatureVisibility` (vector)
: break visibility for the initial time signature.

`instrumentCueName` (markup)
: The name to print if another instrument is to be taken.

`instrumentEqualizer` (procedure)
: A function taking a string (instrument name), and returning a `(`*`min . max`*`)` pair of numbers for the loudness range of the instrument.

`instrumentName` (markup)
: The name to print left of a staff. The `instrumentName` property labels the staff in the first system, and the `shortInstrumentName` property labels following lines.

`instrumentTransposition` (pitch)
: Define the transposition of the instrument. Its value is the pitch that sounds when the instrument plays written middle C. This is used to transpose the MIDI output, and `\quote`s.

`internalBarNumber` (integer)
: Contains the current barnumber. This property is used for internal timekeeping, among others by the `Accidental_engraver`.

`keepAliveInterfaces` (list)
: A list of symbols, signifying grob interfaces that are worth keeping a staff with `remove-empty` set around for.

`keyAlterationOrder` (list)
: An alist that defines in what order alterations should be printed. The format is `(`*`step . alter`*`)`, where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

`keyAlterations` (list)
: The current key signature. This is an alist containing `(`*`step . alter`*`)` or `((`*`octave . step`*`) . alter)`, where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. `keyAlterations = #`((6 . ,FLAT))`.

`lyricMelismaAlignment` (number)
: Alignment to use for a melisma syllable.

`magnifyStaffValue` (positive number)
: The most recent value set with `\magnifyStaff`.

`majorSevenSymbol` (markup)
: How should the major 7th be formatted in a chord name?

`markFormatter` (procedure)
: A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

`maximumFretStretch` (number)
: Don't allocate frets further than this from specified frets.

`measureLength` (moment)
: Length of one measure in the current time signature.

`measurePosition` (moment)
: How much of the current measure have we had. This can be set manually to create incomplete measures.

`melismaBusyProperties` (list)
: A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example, if set to `'(melismaBusy beamMelismaBusy)`, only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)
: How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)
: The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)
: The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)
: The offset of middle C from the position given by `middleCClefPosition` This is used for ottava brackets.

`middleCPosition` (number)
: The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiBalance` (number)
: Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChannelMapping` (symbol)
: How to map MIDI channels: per `staff` (default), `instrument` or `voice`.

`midiChorusLevel` (number)
: Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiExpression` (number)
: Expression control for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiInstrument` (string)
: Name of the MIDI instrument to use.

`midiMaximumVolume` (number)
: Analogous to `midiMinimumVolume`.

`midiMergeUnisons` (boolean)
: If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

`midiMinimumVolume` (number)
: Set the minimum loudness for MIDI. Ranges from 0 to 1.

`midiPanPosition` (number)
: Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)
: Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`minimumFret` (number)
: The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)
: Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)
: Minimum length of a repeated section for a page turn to be allowed within that section.

`minorChordModifier` (markup)
: Markup displayed following the root for a minor chord

`noChordSymbol` (markup)
: Markup to be displayed for rests in a ChordNames context.

`noteNameFunction` (procedure)
: Function used to convert pitches into strings and markups.

`noteNameSeparator` (string)
: String used to separate simultaneous NoteName objects.

`noteToFretFunction` (procedure)
: Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`nullAccidentals` (boolean)
: The `Accidental_engraver` generates no accidentals for notes in contexts were this is set. In addition to supressing the printed accidental, this option removes any effect the note would have had on accidentals in other voices.

`ottavation` (markup)
: If set, the text for an ottava spanner. Changing this creates a new text spanner.

`ottavationMarkups` (list)
: An alist defining the markups used for ottava brackets. It contains entries of the form (*`number of octaves . markup`*).

`output` (music output)
: The output produced by a score-level translator during music interpretation.

`partCombineForced` (symbol)
: Override for the partCombine decision. Can be `apart`, `chords`, `unisono`, `solo1`, or `solo2`.

`partCombineTextsOnNote` (boolean)
: Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)
: See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)
: See `pedalSustainStyle`.

`pedalSustainStrings` (list)
: A list of strings to print for sustain-pedal. Format is `(up updown down)`, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)
: A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)
: See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)
: See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)
: The hash table of predefined fret diagrams to use in FretBoards.

`printAccidentalNames` (boolean or symbol)
: Print accidentals in the `NoteNames` context.

`printKeyCancellation` (boolean)
: Print restoration alterations before a key signature change.

`printNotesLanguage` (string)
: Use a specific language in the `NoteNames` context.

`printOctaveNames` (boolean or symbol)
: Print octave marks in the `NoteNames` context.

`printPartCombineTexts` (boolean)
: Set 'Solo' and 'A due' texts in the part combiner?

`proportionalNotationDuration` (moment)
: Global override for shortest-playing duration. This is used for switching on proportional notation.

`rehearsalMark` (integer)
: The last rehearsal mark printed.

`repeatCommands` (list)
: This property is a list of commands of the form `(list 'volta x)`, where $x$ is a string or `#f`. `'end-repeat` is also accepted as a command.

`repeatCountVisibility` (procedure)
: A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when `countPercentRepeats` is set.

`restCompletionBusy` (boolean)
: Signal whether a completion-rest is active.

`restNumberThreshold` (number)
: If a multimeasure rest has more measures than this, a number is printed.

`restrainOpenStrings` (boolean)
: Exclude open strings from the automatic fret calculator.

`searchForVoice` (boolean)
: Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

`segnoType` (string)
: Set the default bar line for a requested segno. Default is '`S`'.

`shapeNoteStyles` (vector)
: Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

`shortInstrumentName` (markup)
: See `instrumentName`.

`shortVocalName` (markup)
: Name of a vocal line, short version.

`skipBars` (boolean)
: If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

`skipTypesetting` (boolean)
: If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

`slashChordSeparator` (markup)
: The markup object used to separate a chord name from its root note in case of inversions or slash chords.

`soloIIText` (markup)
: The text for the start of a solo for voice 'two' when part-combining.

`soloText` (markup)
: The text for the start of a solo when part-combining.

`squashedPosition` (integer)
: Vertical position of squashing for Sezione "Pitch_squash_engraver" in *Guida al Funzionamento Interno*.

`staffLineLayoutFunction` (procedure)
: Layout of staff lines, `traditional`, or `semitone`.

`stanza` (markup)
: Stanza 'number' to print before the start of a verse. Use in `Lyrics` context.

`startAtNoteColumn` (boolean)
: Start `DurationLine` grob at entire `NoteColumn`.

`startAtSkip` (boolean)
: Start `DurationLine` grob at `skip-event`.

`startRepeatSegnoType` (string)
: Set the default bar line for the combinations beginning of repeat with segno. Default is '`S.|:`'.

`startRepeatType` (string)
: Set the default bar line for the beginning of repeats.

`stemLeftBeamCount` (integer)
: Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

`stemRightBeamCount` (integer)
: See `stemLeftBeamCount`.

`strictBeatBeaming` (boolean)
: Should partial beams reflect the beat structure even if it causes flags to hang out?

`stringNumberOrientations` (list)
: See `fingeringOrientations`.

`stringOneTopmost` (boolean)
: Whether the first string is printed on the top line of the tablature.

`stringTunings` (list)
: The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

`strokeFingerOrientations` (list)
: See `fingeringOrientations`.

`subdivideBeams` (boolean)
: If set, multiple beams will be subdivided at `baseMoment` positions by only drawing one beam over the beat.

`suggestAccidentals` (boolean or symbol)
: If set to `#t`, accidentals are typeset as suggestions above the note. Setting it to `'cautionary` only applies that to cautionary accidentals.

`supportNonIntegerFret` (boolean)
: If set in `Score` the `TabStaff` will print micro-tones as '$2\frac{1}{2}$'

`suspendRestMerging` (boolean)
: When using the Merge_rest_engraver do not merge rests when this is set to true.

`systemStartDelimiter` (symbol)
: Which grob to make for the start of the system/staff? Set to `SystemStartBrace`, `SystemStartBracket` or `SystemStartBar`.

`systemStartDelimiterHierarchy` (pair)
: A nested list, indicating the nesting of a start delimiters.

`tablatureFormat` (procedure)
: A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

`tabStaffLineLayoutFunction` (procedure)
: A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

`tempoHideNote` (boolean)
: Hide the note = count in tempo marks.

`tempoWholesPerMinute` (moment)
: The tempo in whole notes per minute.

`tieWaitForNote` (boolean)
: If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

`timeSignatureFraction` (fraction, as pair)
: A pair of numbers, signifying the time signature. For example, `'(4 . 4)` is a 4/4 time signature.

`timeSignatureSettings` (list)
: A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for `baseMoment`, `beatStructure`, and `beamExceptions`.

`timing` (boolean)
: Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

`tonic` (pitch)
: The tonic of the current scale.

`topLevelAlignment` (boolean)
: If true, the *Vertical_align_engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

`tupletFullLength` (boolean)
: If set, the tuplet is printed up to the start of the next note.

`tupletFullLengthNote` (boolean)
: If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

`tupletSpannerDuration` (moment)
: Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

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

`useBassFigureExtenders` (boolean)
: Whether to use extender lines for repeated bass figures.

`vocalName` (markup)
: Name of a vocal line.

`voltaSpannerDuration` (moment)
: This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)
: This property is read to determine what type of bar line to create.

  Example:

  ```
  \set Staff.whichBar = ".|:"
  ```

  This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

## A.18 Proprietà della formattazione

`add-stem-support` (boolean)
: If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)
: Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)
: Which side to align? `-1`: left side, `0`: around center of width, `1`: right side.

`allow-loose-spacing` (boolean)
: If set, column can be detached from main spacing.

`allow-span-bar` (boolean)
: If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)
: Alteration numbers for accidental.

`alteration-alist` (list)
: List of `(`*`pitch . accidental`*`)` pairs for key signature.

`annotation-balloon` (boolean)
: Print the balloon around an annotation.

`annotation-line` (boolean)
: Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)
: If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)
: Arrow length.

`arrow-width` (number)
: Arrow width.

`auto-knee-gap` (dimension, in staff space)
: If a gap is found between note heads where a horizontal beam fits and it is larger than this number, make a kneed beam.

`automatically-numbered` (boolean)
: If set, footnotes are automatically numbered.

`average-spacing-wishes` (boolean)
: If set, the spacing wishes are averaged over staves.

`avoid-note-head` (boolean)
: If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

`avoid-scripts` (boolean)
: If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

`avoid-slur` (symbol)
: Method of handling slur collisions. Choices are `inside`, `outside`, `around`, and `ignore`. `inside` adjusts the slur if needed to keep the grob inside the slur. `outside` moves the grob vertically to the outside of the slur. `around` moves the grob vertically to the outside of the slur only if there is a collision. `ignore` does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), `outside` and `around` behave like `ignore`.

`axes` (list)
: List of axis numbers. In the case of alignment grobs, this should contain only one number.

`bar-extent` (pair of numbers)
: The Y-extent of the actual bar line. This may differ from `Y-extent` because it does not include the dots in a repeat bar line.

`base-shortest-duration` (moment)
: Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

`baseline-skip` (dimension, in staff space)
: Distance between base lines of multiple lines of text.

`beam-thickness` (dimension, in staff space)
: Beam thickness, measured in `staff-space` units.

`beam-width` (dimension, in staff space)
: Width of the tremolo sign.

`beamed-stem-shorten` (list)
: How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

`beaming` (pair)
: Pair of number lists. Each number list specifies which beams to make. `0` is the central beam, `1` is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

`beamlet-default-length` (pair)
: A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by `beamlet-max-length-proportion`, whichever is smaller.

`beamlet-max-length-proportion` (pair)
: The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

`before-line-breaking` (boolean)
: Dummy property, used to trigger a callback function.

`between-cols` (pair)
: Where to attach a loose column to.

`bound-details` (list)
: An alist of properties for determining attachments of spanners to edges.

`bound-padding` (number)
: The amount of padding to insert around spanner bounds.

`bracket-flare` (pair of numbers)
: A pair of numbers specifying how much edges of brackets should slant outward. Value `0.0` means straight edges.

`bracket-visibility` (boolean or symbol)
: This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

`break-align-anchor` (number)
: Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

`break-align-anchor-alignment` (number)
: Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

`break-align-orders` (vector)
: This is a vector of 3 lists: `#(`*`end-of-line unbroken start-of-line`*`)`. Each list contains *break-align symbols* that specify an order of breakable items (see Sezione "break-alignment-interface" in *Guida al Funzionamento Interno*).

  For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
                    ambitus
                    breathing-sign
                    time-signature
                    clef
                    cue-clef
                    staff-bar
                    key-cancellation
                    key-signature
                    custos))
```

`break-align-symbol` (symbol)
: This key is used for aligning, ordering, and spacing breakable items. See Sezione "break-alignment-interface" in *Guida al Funzionamento Interno*.

`break-align-symbols` (list)
: A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to `break-visibility`, we will align to the next grob (and so on). Choices are listed in Sezione "break-alignment-interface" in *Guida al Funzionamento Interno*.

`break-overshoot` (pair of numbers)
: How much does a broken spanner stick out of its bounds?

`break-visibility` (vector)
: A vector of 3 booleans, `#(`*`end-of-line unbroken begin-of-line`*`)`. `#t` means visible, `#f` means killed.

`breakable` (boolean)
: Allow breaks here.

`broken-bound-padding` (number)
: The amount of padding to insert when a spanner is broken at a line break.

`chord-dots-limit` (integer)
: Limits the column of dots on each chord to the height of the chord plus `chord-dots-limit` staff-positions.

`circled-tip` (boolean)
: Put a circle at start/end of hairpins (al/del niente).

`clef-alignments` (list)
: An alist of parent-alignments that should be used for clef modifiers with various clefs

`clip-edges` (boolean)
: Allow outward pointing beamlets at the edges of beams?

`collapse-height` (dimension, in staff space)
: Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

`collision-interfaces` (list)
: A list of interfaces for which automatic beam-collision resolution is run.

`collision-voice-only` (boolean)
: Does automatic beam collsion apply only to the voice in which the beam was created?

`color` (color)
: The color of this grob.

`common-shortest-duration` (moment)
: The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

`concaveness` (number)
: A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

`connect-to-neighbor` (pair)
: Pair of booleans, indicating whether this grob looks as a continued break.

`control-points` (list of number pairs)
: List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

`count-from` (integer)
: The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

`damping` (number)
: Amount of beam slope damping.

`dash-definition` (pair)
: List of `dash-elements` defining the dash structure. Each `dash-element` has a starting t value, an ending t-value, a `dash-fraction`, and a `dash-period`.

`dash-fraction` (number)
: Size of the dashes, relative to `dash-period`. Should be between `0.1` and `1.0` (continuous line). If set to `0.0`, a dotted line is produced

`dash-period` (number)
: The length of one dash together with whitespace. If negative, no line is drawn at all.

`dashed-edge` (boolean)
: If set, the bracket edges are dashed like the rest of the bracket.

`default-direction` (direction)
: Direction determined by note head positions.

`default-staff-staff-spacing` (list)
: The settings to use for `staff-staff-spacing` when it is unset, for ungrouped staves and for grouped staves that do not have the relevant `StaffGrouper` property set (`staff-staff-spacing` or `staffgroup-staff-spacing`).

`details` (list)
: Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a `details` property.

`digit-names` (vector)
: Names for string finger digits.

`direction` (direction)
: If `side-axis` is `0` (or `X`), then this property determines whether the object is placed `LEFT`, `CENTER` or `RIGHT` with respect to the other object. Otherwise, it determines whether the object is placed `UP`, `CENTER` or `DOWN`. Numerical values may also be used: `UP`=`1`, `DOWN`=`-1`, `LEFT`=`-1`, `RIGHT`=`1`, `CENTER`=`0`.

`dot-count` (integer)
: The number of dots.

`dot-negative-kern` (number)
: The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

`dot-placement-list` (list)
: List consisting of `(`*`description string-number fret-number finger-number`*`)` entries used to define fret diagrams.

`double-stem-separation` (number)
: The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`duration-log` (integer)
: The 2-log of the note head duration, i.e., `0` = whole note, `1` = half note, etc.

`eccentricity` (number)
: How asymmetrical to make a slur. Positive means move the center to the right.

`edge-height` (pair)
: A pair of numbers specifying the heights of the vertical edges: `(`*`left-height . right-height`*`)`.

`edge-text` (pair)
: A pair specifying the texts to be set at the edges: `(`*`left-text . right-text`*`)`.

`expand-limit` (integer)
: Maximum number of measures expanded in church rests.

`extra-dy` (number)
: Slope glissandi this much extra.

`extra-offset` (pair of numbers)
: A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in `staff-space` units of the staff's `StaffSymbol`.

`extra-spacing-height` (pair of numbers)
: In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to `(-inf.0 . +inf.0)`.

`extra-spacing-width` (pair of numbers)
: In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to `(+inf.0 . -inf.0)`.

`flag-count` (number)
: The number of tremolo beams.

`flag-style` (symbol)
: The style of the flag to be used with `MetronomeMark`. Available are `'modern-straight-flag`, `'old-straight-flag`, `flat-flag`, `mensural` and `'default`

`flat-positions` (list)
: Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`font-encoding` (symbol)
: The font encoding is the broadest category for selecting a font. Currently, only lilypond's system fonts (Emmentaler) are using this property. Available values are `fetaMusic` (Emmentaler), `fetaBraces`, `fetaText` (Emmentaler).

`font-family` (symbol)
: The font family is the broadest category for selecting text fonts. Options include: `sans`, `roman`.

`font-features` (list)
: Opentype features.

`font-name` (string)
: Specifies a file name (without extension) of the font to load. This setting overrides selection using `font-family`, `font-series` and `font-shape`.

`font-series` (symbol)
: Select the series of a font. Choices include `medium`, `bold`, `bold-narrow`, etc.

`font-shape` (symbol)
: Select the shape of a font. Choices include `upright`, `italic`, `caps`.

`font-size` (number)
: The font size, compared to the 'normal' size. `0` is style-sheet's normal size, `-1` is smaller, `+1` is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property `fontSize` is set, its value is added to this before the glyph is printed. Fractional values are allowed.

`footnote` (boolean)
: Should this be a footnote or in-note?

`footnote-music` (music)
: Music creating a footnote.

`footnote-text` (markup)
: A footnote for the grob.

`force-hshift` (number)
: This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Sezione "note-collision-interface" in *Guida al Funzionamento Interno*.

`forced-spacing` (number)
: Spacing forced between grobs, used in various ligature engravers.

`fraction` (fraction, as pair)
: Numerator and denominator of a time signature object.

`french-beaming` (boolean)
: Use French beaming style for this stem. The stem stops at the innermost beams.

`fret-diagram-details` (list)
: An alist of detailed grob properties for fret diagrams. Each alist entry consists of a `(`*`property`*` . `*`value`*`)` pair. The properties which can be included in `fret-diagram-details` include the following:

  - `barre-type` – Type of barre indication used. Choices include `curved`, `straight`, and `none`. Default `curved`.
  - `capo-thickness` – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
  - `dot-color` – Color of dots. Options include `black` and `white`. Default `black`.
  - `dot-label-font-mag` – Magnification for font used to label fret dots. Default value 1.
  - `dot-position` – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-`dot-radius` for dots with labels.
  - `dot-radius` – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
  - `finger-code` – Code for the type of fingering indication used. Options include `none`, `in-dot`, and `below-string`. Default `none` for markup fret diagrams, `below-string` for `FretBoards` fret diagrams.
  - `fret-count` – The number of frets. Default 4.
  - `fret-distance` – Multiplier to adjust the distance between frets. Default 1.0.
  - `fret-label-custom-format` – The format string to be used label the lowest fret number, when `number-type` equals to `custom`. Default `"~a"`.
  - `fret-label-font-mag` – The magnification of the font used to label the lowest fret number. Default 0.5.

- `fret-label-vertical-offset` – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
- `fret-label-horizontal-offset` – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
- `handedness` – Print the fret-diagram left- or right-handed. `-1`, `LEFT` for left ; `1`, `RIGHT` for right. Default `RIGHT`.
- `paren-padding` – The padding for the parenthesis. Default 0.05.
- `label-dir` – Side to which the fret label is attached. `-1`, `LEFT`, or `DOWN` for left or down; `1`, `RIGHT`, or `UP` for right or up. Default `RIGHT`.
- `mute-string` – Character string to be used to indicate muted string. Default `"x"`.
- `number-type` – Type of numbers to use in fret label. Choices include `roman-lower`, `roman-upper`, `arabic` and `custom`. In the later case, the format string is supplied by the `fret-label-custom-format` property. Default `roman-lower`.
- `open-string` – Character string to be used to indicate open string. Default `"o"`.
- `orientation` – Orientation of fret-diagram. Options include `normal`, `landscape`, and `opposing-landscape`. Default `normal`.
- `string-count` – The number of strings. Default 6.
- `string-distance` – Multiplier to adjust the distance between strings. Default 1.0.
- `string-label-font-mag` – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for `normal` orientation, 0.5 for `landscape` and `opposing-landscape`.
- `string-thickness-factor` – Factor for changing thickness of each string in the fret diagram. Thickness of string $k$ is given by `thickness` * (1+`string-thickness-factor`) ^ ($k$-1). Default 0.
- `top-fret-thickness` – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- `xo-font-magnification` – Magnification used for mute and open string indicators. Default value 0.5.
- `xo-padding` – Padding for open and mute indicators from top fret. Default value 0.25.

`full-length-padding` (number)
: How much padding to use at the right side of a full-length tuplet bracket.

`full-length-to-extent` (boolean)
: Run to the extent of the column for a full-length tuplet bracket.

`full-measure-extra-space` (number)
: Extra space that is allocated at the beginning of a measure with only one note. This property is read from the NonMusicalPaperColumn that begins the measure.

`full-size-change` (boolean)
: Don't make a change clef smaller.

`gap` (dimension, in staff space)
: Size of a gap in a variable symbol.

`gap-count` (integer)
: Number of gapped beams for tremolo.

`glissando-skip` (boolean)
: Should this `NoteHead` be skipped by glissandi?

`glyph` (string)
: A string determining what 'style' of glyph is typeset. Valid choices depend on the function that is reading this property.

  In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

`glyph-name` (string)
: The glyph name within the font.

  In the context of (span) bar lines, *glyph-name* represents a processed form of `glyph`, where decisions about line breaking etc. are already taken.

`glyph-name-alist` (list)
: An alist of key-string pairs.

`graphical` (boolean)
: Display in graphical (vs. text) form.

`grow-direction` (direction)
: Crescendo or decrescendo?

`hair-thickness` (number)
: Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`harp-pedal-details` (list)
: An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a `(property . value)` pair. The properties which can be included in harp-pedal-details include the following:

  - `box-offset` – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
  - `box-width` – Width of each pedal box. Default value 0.4.
  - `box-height` – Height of each pedal box. Default value 1.0.
  - `space-before-divider` – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
  - `space-after-divider` – Space between boxes after the first divider. Default value 0.8.
  - `circle-thickness` – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
  - `circle-x-padding` – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
  - `circle-y-padding` – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

`head-direction` (direction)
: Are the note heads left or right in a semitie?

`height` (dimension, in staff space)
: Height of an object in `staff-space` units.

`height-limit` (dimension, in staff space)
: Maximum slur height: The longer the slur, the closer it is to this height.

`hide-tied-accidental-after-break` (boolean)
: If set, an accidental that appears on a tied note after a line break will not be displayed.

`horizon-padding` (number)
: The amount to pad the axis along which a `Skyline` is built for the `side-position-interface`.

`horizontal-shift` (integer)
: An integer that identifies ranking of `NoteColumn`s for horizontal shifting. This is used by Sezione "note-collision-interface" in *Guida al Funzionamento Interno*.

`horizontal-skylines` (pair of skylines)
: Two skylines, one to the left and one to the right of this grob.

`id` (string)
: An id string for the grob.

`ignore-ambitus` (boolean)
: If set, don't consider this notehead for ambitus calculation.

`ignore-collision` (boolean)
: If set, don't do note collision resolution on this `NoteColumn`.

`implicit` (boolean)
: Is this an implicit bass figure?

`inspect-quants` (pair of numbers)
: If debugging is set, set beam and slur position to a (quantized) position that is as close as possible to this value, and print the demerits for the inspected position in the output.

`keep-inside-line` (boolean)
: If set, this column cannot have objects sticking into the margin.

`kern` (dimension, in staff space)
: The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`knee` (boolean)
: Is this beam kneed?

`knee-spacing-correction` (number)
: Factor for the optical correction amount for kneed beams. Set between `0` for no correction and `1` for full correction.

`knee-to-beam` (boolean)
: Determines whether a tuplet number will be positioned next to a kneed beam.

`labels` (list)
: List of labels (symbols) placed on a column.

`layer` (integer)
: An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

`ledger-extra` (dimension, in staff space)
: Extra distance from staff line to draw ledger lines for.

`ledger-line-thickness` (pair of numbers)
: The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

`ledger-positions` (list)
: Vertical positions of ledger lines. When set on a `StaffSymbol` grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

`ledger-positions-function` (any type)
: A quoted Scheme procedure that takes a `StaffSymbol` grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

`left-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`left-padding` (dimension, in staff space)
: The amount of space that is put left to an object (e.g., a lyric extender).

`length` (dimension, in staff space)
: User override for the stem length of unbeamed stems (each unit represents half a `staff-space`).

`length-fraction` (number)
: Multiplier for lengths. Used for determining ledger lines and stem lengths.

`line-break-penalty` (number)
: Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

`line-break-permission` (symbol)
: Instructs the line breaker on whether to put a line break at this column. Can be `force` or `allow`.

`line-break-system-details` (list)
: An alist of properties to use if this column is the start of a system.

`line-count` (integer)
: The number of staff lines.

`line-positions` (list)
: Vertical positions of staff lines.

`line-thickness` (number)
: For slurs and ties, this is the diameter of the virtual “pen” that draws the two arcs of the curve’s outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *`Staff`*`.StaffSymbol.thickness`).

`long-text` (markup)
: Text markup. See Sezione “Formatting text” in *Guida alla Notazione*.

`max-beam-connect` (integer)
: Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

`max-symbol-separation` (number)
: The maximum distance between symbols making up a church rest.

`maximum-gap` (number)
: Maximum value allowed for `gap` property.

`measure-count` (integer)
: The number of measures for a multi-measure rest.

`measure-length` (moment)
: Length of a measure. Used in some spacing situations.

`merge-differently-dotted` (boolean)
: Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

  `merge-differently-dotted` only applies to opposing stem directions (i.e., voice 1 & 2).

`merge-differently-headed` (boolean)
: Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Sezione "note-collision-interface" in *Guida al Funzionamento Interno*.

  `merge-differently-headed` only applies to opposing stem directions (i.e., voice 1 & 2).

`minimum-distance` (dimension, in staff space)
: Minimum distance between rest and notes or beam.

`minimum-length` (dimension, in staff space)
: Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance between noteheads.

`minimum-length-after-break` (dimension, in staff space)
: If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the `springs-and-rods` property. If added to a `Tie`, this sets the minimum distance to the notehead.

`minimum-length-fraction` (number)
: Minimum length of ledger line as fraction of note head size.

`minimum-space` (dimension, in staff space)
: Minimum distance that the victim should move (after padding).

`minimum-X-extent` (pair of numbers)
: Minimum size of an object in X dimension, measured in `staff-space` units.

`minimum-Y-extent` (pair of numbers)
: Minimum size of an object in Y dimension, measured in `staff-space` units.

`neutral-direction` (direction)
: Which direction to take in the center of the staff.

`neutral-position` (number)
: Position (in half staff spaces) where to flip the direction of custos stem.

`next` (graphical (layout) object)
: Object that is next relation (e.g., the lyric syllable following an extender).

`no-alignment` (boolean)
: If set, don't place this grob in a `VerticalAlignment`; rather, place it using its own `Y-offset` callback.

`no-ledgers` (boolean)
: If set, don't draw ledger lines on this object.

`no-stem-extend` (boolean)
: If set, notes with ledger lines do not get stems extending to the middle staff line.

`non-break-align-symbols` (list)
: A list of symbols that determine which NON-break-aligned interfaces to align this to.

`non-default` (boolean)
: Set for manually specified clefs and keys.

`non-musical` (boolean)
: True if the grob belongs to a `NonMusicalPaperColumn`.

`nonstaff-nonstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of `staff-affinity`, if both are on the same side of the related staff, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-relatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of `staff-affinity`, if there are no non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. If `staff-affinity` is `CENTER`, then `nonstaff-relatedstaff-spacing` is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See `staff-staff-spacing` for a description of the alist structure.

`nonstaff-unrelatedstaff-spacing` (list)
: The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from `staff-affinity`, if there are no other non-staff lines between the two, and `staff-affinity` is either `UP` or `DOWN`. See `staff-staff-spacing` for a description of the alist structure.

`normalized-endpoints` (pair)
: Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

`note-collision-threshold` (dimension, in staff space)
: Simultaneous notes that are this close or closer in units of `staff-space` will be identified as vertically colliding. Used by `Stem` grobs for notes in the same voice, and `NoteCollision` grobs for notes in different voices. Default value 1.

`note-names` (vector)
: Vector of strings containing names for easy-notation note heads.

`number-type` (symbol)
: Numbering style. Choices include `roman-lower`, `roman-upper` and `arabic`.

`output-attributes` (list)
: An alist of attributes for the grob, to be included in output files. When the SVG typesetting backend is used, the attributes are assigned to a group (`<g>`) containing all of the stencils that comprise a given grob. For example, `'((id . 123) (class . foo) (data-whatever . \bar"))` will produce `<g id=\123" class=\foo" data-whatever=\bar"> ... </g>`. In the Postscript backend, where there is no way to group items, the setting of the output-attributes property will have no effect.

`outside-staff-horizontal-padding` (number)
: By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

`outside-staff-padding` (number)
: The padding to place between grobs when spacing according to `outside-staff-priority`. Two grobs with different `outside-staff-padding` values have the larger value of padding between them.

`outside-staff-placement-directive` (symbol)
: One of four directives telling how outside staff objects should be placed.
  - `left-to-right-greedy` – Place each successive grob from left to right.
  - `left-to-right-polite` – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
  - `right-to-left-greedy` – Same as `left-to-right-greedy`, but from right to left.
  - `right-to-left-polite` – Same as `left-to-right-polite`, but from right to left.

`outside-staff-priority` (number)
: If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller `outside-staff-priority` is closer to the staff.

`packed-spacing` (boolean)
: If set, the notes are spaced as tightly as possible.

`padding` (dimension, in staff space)
: Add this much extra space between objects that are next to each other.

`padding-pairs` (list)
: An alist mapping `(`*`name`*` . `*`name`*`)` to distances.

`page-break-penalty` (number)
: Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

`page-break-permission` (symbol)
: Instructs the page breaker on whether to put a page break at this column. Can be `force` or `allow`.

`page-number` (number)
: Page number on which this system ends up.

`page-turn-penalty` (number)
: Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

`page-turn-permission` (symbol)
: Instructs the page breaker on whether to put a page turn at this column. Can be `force` or `allow`.

`parent-alignment-X` (number)
: Specify on which point of the parent the object is aligned. The value `-1` means aligned on parent's left edge, `0` on center, and `1` right edge, in X direction. Other

numerical values may also be specified - the unit is half the parent's width. If unset, the value from `self-alignment-X` property will be used.

`parent-alignment-Y` (number)
: Like `parent-alignment-X` but for the Y axis.

`parenthesis-friends` (list)
: A list of Grob types, as symbols. When parentheses enclose a Grob that has 'parenthesis-friends, the parentheses widen to include any child Grobs with type among 'parenthesis-friends.

`parenthesized` (boolean)
: Parenthesize this grob.

`positions` (pair of numbers)
: Pair of staff coordinates (*start . end*), where *start* and *end* are vertical positions in `staff-space` units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.

`prefer-dotted-right` (boolean)
: For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.

`protrusion` (number)
: In an arpeggio bracket, the length of the horizontal edges.

`rank-on-page` (number)
: 0-based index of the system on a page.

`ratio` (number)
: Parameter for slur shape. The higher this number, the quicker the slur attains its `height-limit`.

`remove-empty` (boolean)
: If set, remove group if it contains no interesting items.

`remove-first` (boolean)
: Remove the first staff of an orchestral score?

`remove-layer` (index or symbol)
: When set as a positive integer, the `Keep_alive_together_engraver` removes all `VerticalAxisGroup` grobs with a `remove-layer` larger than the smallest retained `remove-layer`. Set to `#f` to make a layer independent of the `Keep_alive_together_engraver`. Set to `'()`, the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: `#'any` to keep the layer alive with any other layer in the group; `#'above` or `#'below` to keep the layer alive with the context immediately before or after it, respectively.

`replacement-alist` (list)
: Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.

`restore-first` (boolean)
: Print a natural before the accidental.

`rhythmic-location` (rhythmic location)
: Where (bar number, measure position) in the score.

`right-bound-info` (list)
: An alist of properties for determining attachments of spanners to edges.

`right-padding` (dimension, in staff space)
: Space to insert on the right side of an object (e.g., between note and its accidentals).

`rotation` (list)
: Number of degrees to rotate this object, and what point to rotate around. For example, `'(45 0 0)` rotates by 45 degrees around the center of this object.

`round-up-exceptions` (list)
: A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.

`round-up-to-longer-rest` (boolean)
: Displays the longer multi-measure rest when the length of a measure is between two values of `usable-duration-logs`. For example, displays a breve instead of a whole in a 3/2 measure.

`rounded` (boolean)
: Decide whether lines should be drawn rounded or not.

`same-direction-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

`script-priority` (number)
: A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

`segno-kern` (number)
: The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`self-alignment-X` (number)
: Specify alignment of an object. The value `-1` means left aligned, `0` centered, and `1` right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

`self-alignment-Y` (number)
: Like `self-alignment-X` but for the Y axis.

`shape` (symbol)
: This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

`sharp-positions` (list)
: Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: `(alto treble tenor soprano baritone mezzosoprano bass)`. If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

`shorten-pair` (pair of numbers)
: The lengths to shorten on both sides a hairpin or text-spanner such as a pedal bracket. Positive values shorten the hairpin or text-spanner, while negative values lengthen it.

`shortest-duration-space` (number)
: Start with this multiple of `spacing-increment` space for the shortest duration. See also Sezione "spacing-spanner-interface" in *Guida al Funzionamento Interno*.

`shortest-playing-duration` (moment)
: The duration of the shortest note playing here.

`shortest-starter-duration` (moment)
: The duration of the shortest note that starts here.

`side-axis` (number)
: If the value is `X` (or equivalently `0`), the object is placed horizontally next to the other object. If the value is `Y` or `1`, it is placed vertically.

`side-relative-direction` (direction)
: Multiply direction of `direction-source` with this to get the direction of this object.

`simple-Y` (boolean)
: Should the Y placement of a spanner disregard changes in system heights?

`size` (number)
: The ratio of the size of the object to its default size.

`skip-quanting` (boolean)
: Should beam quanting be skipped?

`skyline-horizontal-padding` (number)
: For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

`skyline-vertical-padding` (number)
: The amount by which the left and right skylines of a column are padded vertically, beyond the `Y-extent`s and `extra-spacing-height`s of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

`slash-negative-kern` (number)
: The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

`slope` (number)
: The slope of this object.

`slur-padding` (number)
: Extra distance between slur and script.

`snap-radius` (number)
: The maximum distance between two objects that will cause them to snap to alignment along an axis.

`space-alist` (list)
: An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

: Standard choices for *`break-align-symbol`* are listed in Sezione "break-alignment-interface" in *Guida al Funzionamento Interno*. Additionally, three special break-align symbols available to `space-alist` are:

`first-note`
: used when the grob is just left of the first note on a line

`next-note`
: used when the grob is just left of any other note; if not set, the value of `first-note` gets used

`right-edge`
: used when the grob is the last item on the line (only compatible with the `extra-space` spacing style)

Choices for *`spacing-style`* are:

`extra-space`
: Put this much space between the two grobs. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed.

`minimum-space`
: Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with `first-note` or `next-note`; otherwise it is fixed. Not compatible with `right-edge`.

`fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much fixed space between the grob and the note.

`minimum-fixed-space`
: Only compatible with `first-note` and `next-note`. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

`semi-fixed-space`
: Only compatible with `first-note` and `next-note`. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

`space-to-barline` (boolean)
: If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

`spacing-increment` (dimension, in staff space)
: The unit of length for note-spacing. Typically, the width of a note head. See also Sezione “spacing-spanner-interface” in *Guida al Funzionamento Interno*.

`spacing-pair` (pair)
: A pair of alignment symbols which set an object's spacing relative to its left and right `BreakAlignment`s.

  For example, a `MultiMeasureRest` will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest.spacing-pair =
              #'(staff-bar . staff-bar)
```

`spanner-id` (index or symbol)
: An identifier to distinguish concurrent spanners.

`springs-and-rods` (boolean)
: Dummy variable for triggering spacing routines.

`stacking-dir` (direction)
: Stack objects in which direction?

`staff-affinity` (direction)
: The direction of the staff to use for spacing the current non-staff line. Choices are `UP`, `DOWN`, and `CENTER`. If `CENTER`, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting `staff-affinity` for a staff causes it to be treated as a non-staff line. Setting `staff-affinity` to `#f` causes a non-staff line to be treated as a staff.

`staff-padding` (dimension, in staff space)
: Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

`staff-position` (number)
: Vertical position, measured in half staff spaces, counted from the middle line.

`staff-space` (dimension, in staff space)
: Amount of space between staff lines, expressed in global `staff-space`.

`staff-staff-spacing` (list)
: When applied to a staff-group's `StaffGrouper` grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's `VerticalAxisGroup` grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the `StaffGrouper` grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

  - `basic-distance` – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
  - `minimum-distance` – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
  - `padding` – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
  - `stretchability` – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

`staffgroup-staff-spacing` (list)
: The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the `staff-staff-spacing` property of the staff's `VerticalAxisGroup` grob is set, that is used instead. See `staff-staff-spacing` for a description of the alist structure.

`stem-attachment` (pair of numbers)
: An `(x . y)` pair where the stem attaches to the notehead.

`stem-begin-position` (number)
: User override for the begin position of a stem.

`stem-spacing-correction` (number)
: Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

`stemlet-length` (number)
: How long should be a stem over a rest?

`stencil` (stencil)
: The symbol to print.

`stencils` (list)
: Multiple stencils, used as intermediate value.

`strict-grace-spacing` (boolean)
: If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

`strict-note-spacing` (boolean)
: If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

`stroke-style` (string)
: Set to `"grace"` to turn stroke through flag on.

`style` (symbol)
: This setting determines in what style a grob is typeset. Valid choices depend on the `stencil` callback reading this property.

`text` (markup)
: Text markup. See Sezione “Formatting text” in *Guida alla Notazione*.

`text-direction` (direction)
: This controls the ordering of the words. The default `RIGHT` is for roman text. Arabic or Hebrew should use `LEFT`.

`thick-thickness` (number)
: Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

`thickness` (number)
: For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual “pen” that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to `Staff.StaffSymbol.thickness`).

`tie-configuration` (list)
: List of `(position . dir)` pairs, indicating the desired tie configuration, where *position* is the offset from the center of the staff in staff space and *dir* indicates the direction of the tie (`1`=>up, `-1`=>down, `0`=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

`to-barline` (boolean)
: If true, the spanner will stop at the bar line just before it would otherwise stop.

`toward-stem-shift` (number)
: Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`toward-stem-shift-in-column` (number)
: Amount by which a script is shifted toward the stem if its direction coincides with the stem direction and it is associated with a `ScriptColumn` object. `0.0` means centered on the note head (the default position of most scripts); `1.0` means centered on the stem. Interpolated values are possible.

`transparent` (boolean)
: This makes the grob invisible.

`tuplet-slur` (boolean)
: Draw a slur instead of a bracket for tuplets.

`uniform-stretching` (boolean)
: If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

`usable-duration-logs` (list)
: List of `duration-log`s that can be used in typesetting the grob.

`use-skylines` (boolean)
: Should skylines be used for side positioning?

`used` (boolean)
: If set, this spacing column is kept in the spacing problem.

`vertical-skylines` (pair of skylines)
: Two skylines, one above and one below this grob.

`voiced-position` (number)
: The staff-position of a voiced `Rest`, negative if the rest has `direction DOWN`.

`when` (moment)
: Global time step associated with this column.

`whiteout` (boolean-or-number)
: If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The `LyricHyphen` grob uses a special implementation of whiteout: A positive number indicates how far the white background extends beyond the bounding box in multiples of `line-thickness`. The shape of the background is determined by `whiteout-style`. Usually `#f` by default.

`whiteout-style` (symbol)
: Determines the shape of the `whiteout` background. Available are `'outline`, `'rounded-box`, and the default `'box`. There is one exception: Use `'special` for `LyricHyphen`.

`width` (dimension, in staff space)
: The width of a grob measured in staff space.

`word-space` (dimension, in staff space)
: Space to insert between words in texts.

`X-align-on-main-noteheads` (boolean)
: If true, this grob will ignore suspended noteheads when aligning itself on NoteColumn.

`X-extent` (pair of numbers)
: Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

`X-offset` (number)
: The horizontal amount that this object is moved relative to its X-parent.

`X-positions` (pair of numbers)
: Pair of X staff coordinates of a spanner in the form `(left . right)`, where both *left* and *right* are in `staff-space` units of the current staff.

`Y-extent` (pair of numbers)
: Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

`Y-offset` (number)
: The vertical amount that this object is moved relative to its Y-parent.

`zigzag-length` (dimension, in staff space)
: The length of the lines of a zigzag, relative to `zigzag-width`. A value of `1` gives 60-degree zigzags.

`zigzag-width` (dimension, in staff space)
: The width of one zigzag squiggle. This number is adjusted slightly so that the spanner line can be constructed from a whole number of squiggles.

## A.19 Funzioni musicali disponibili

`\absolute` [music] - *music* (music)
: Make *music* absolute. This does not actually change the music itself but rather hides it from surrounding `\relative` and `\fixed` commands.

`\acciaccatura` [music] - *music* (music)
: Create an acciaccatura from the following music expression

`\accidentalStyle` [music] - *style* (symbol list)
: Set accidental style to symbol list *style* in the form '`piano-cautionary`'. If *style* has a form like '`Staff.piano-cautionary`', the settings are applied to that context. Otherwise, the context defaults to '`Staff`', except for piano styles, which use '`GrandStaff`' as a context.

`\addChordShape` [void] - *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair)
: Add chord shape *shape-definition* to the *chord-shape-table* hash with the key `(cons key-symbol tuning)`.

`\addInstrumentDefinition` [void] - *name* (string) *lst* (list)
: Create instrument *name* with properties *list*.

`\addQuote` [void] - *name* (string) *music* (music)
: Define *music* as a quotable music expression named *name*

`\afterGrace` [music] - *fraction* [non-negative rational, fraction, or moment] *main* (music) *grace* (music)
: Create *grace* note(s) after a *main* music expression.

The musical position of the grace expression is after a given fraction of the main note's duration has passed. If *fraction* is not specified as first argument, it is taken from `afterGraceFraction` which has a default value of `3/4`.

`\allowPageTurn` [music]
: Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

`\allowVoltaHook` [void] - *bar* (string)
: Allow the volta bracket hook being drawn over bar line *bar*.

`\alterBroken` [music] - *property* (key list or symbol) *arg* (list) *item* (key list or music)
: Override *property* for pieces of broken spanner *item* with values *arg*. *item* may either be music in the form of a starting spanner event, or a symbol list in the form '`Context.Grob`' or just '`Grob`'. Iff *item* is in the form of a spanner event, *property* may also have the form '`Grob.property`' for specifying a directed tweak.

`\ambitusAfter` [music] - *target* (symbol)
: Move the ambitus after the break-align symbol *target*.

`\appendToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
: Append *more* to the `elements` of all music expressions in *music* that are tagged with *tag*.

`\applyContext` [music] - *proc* (procedure)
: Modify context properties with Scheme procedure *proc*.

`\applyMusic` [music] - *func* (procedure) *music* (music)
: Apply procedure *func* to *music*.

`\applyOutput` [music] - *target* (symbol list or symbol) *proc* (procedure)
: Apply function `proc` to every layout object matched by *target* which takes the form `Context` or `Context.Grob`.

`\appoggiatura` [music] - *music* (music)
: Create an appoggiatura from *music*

`\assertBeamQuant` [music] - *l* (pair) *r* (pair)
: Testing function: check whether the beam quants *l* and *r* are correct

`\assertBeamSlope` [music] - *comp* (procedure)
: Testing function: check whether the slope of the beam is the same as `comp`

`\autoChange` [music] - *pitch* [pitch] *clef-1* [context modification] *clef-2* [context modification] *music* (music)
: Make voices that switch between staves automatically. As an option the pitch where to switch staves may be specified. The clefs for the staves are optional as well. Setting clefs works only for implicitly instantiated staves.

`\balloonGrobText` [music] - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
: Attach *text* to *grob-name* at offset *offset* (use like `\once`)

`\balloonText` [post event] - *offset* (pair of numbers) *text* (markup)
: Attach *text* at *offset* (use like `\tweak`)

`\bar` [music] - *type* (string)
: Insert a bar line of type *type*

`\barNumberCheck` [music] - *n* (integer)
: Print a warning if the current bar number is not *n*.

`\beamExceptions` (any type) - *music* (music)
: Extract a value suitable for setting `Timing.beamExceptions` from the given pattern with explicit beams in *music*. A bar check `|` has to be used between bars of patterns in order to reset the timing.

`\bendAfter` [post event] - *delta* (real number)
: Create a fall or doit of pitch interval *delta*.

`\bookOutputName` [void] - *newfilename* (string)
: Direct output for the current book block to *newfilename*.

`\bookOutputSuffix` [void] - *newsuffix* (string)
: Set the output filename suffix for the current book block to *newsuffix*.

`\breathe` [music]
: Insert a breath mark.

`\chordRepeats` [music] - *event-types* [list] *music* (music)
: Walk through *music* putting the notes of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(string-number-event)`.

`\clef` [music] - *type* (string)
: Set the current clef to *type*.

`\compoundMeter` [music] - *args* (pair)
: Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

`\compressMMRests` [music] - *music* (music)
: Remove the empty bars created by multi-measure rests, leaving just the first bar containing the MM rest itself.

`\crossStaff` [music] - *notes* (music)
: Create cross-staff stems

`\cueClef` [music] - *type* (string)
: Set the current cue clef to *type*.

`\cueClefUnset` [music]
: Unset the current cue clef.

`\cueDuring` [music] - *what* (string) *dir* (direction) *main-music* (music)
: Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`\cueDuringWithClef` [music] - *what* (string) *dir* (direction) *clef* (string) *main-music* (music)
: Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.

`\deadNote` [music] - *note* (music)
: Print *note* with a cross-shaped note head.

`\defineBarLine` [void] - *bar* (string) *glyph-list* (list)
: Define bar line settings for bar line *bar*. The list *glyph-list* must have three entries which define the appearance at the end of line, at the beginning of the next line, and the span bar, respectively.

`\displayLilyMusic` [music] - *port* [output port] *music* (music)
Display the LilyPond input representation of *music* to *port*, defaulting to the console.

`\displayMusic` [music] - *port* [output port] *music* (music)
Display the internal representation of *music* to *port*, default to the console.

`\displayScheme` (any type) - *port* [output port] *expr* (any type)
Display the internal representation of *expr* to *port*, default to the console.

`\dropNote` [music] - *num* (integer) *music* (music)
Drop a note of any chords in *music*, in *num* position from above.

`\endSpanners` [music] - *music* (music)
Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.

`\eventChords` [music] - *music* (music)
Compatibility function wrapping `EventChord` around isolated rhythmic events occuring since version 2.15.28, after expanding repeat chords '`q`'.

`\featherDurations` [music] - *factor* (moment) *argument* (music)
Adjust durations of music in *argument* by rational *factor*.

`\finger` [post event] - *finger* (integer or markup)
Apply *finger* as a fingering indication.

`\fixed` [music] - *pitch* (pitch) *music* (music)
Use the octave of *pitch* as the default octave for *music*.

`\footnote` [music] - *mark* [markup] *offset* (pair of numbers) *footnote* (markup) *item* (symbol list or music)
Make the markup *footnote* a footnote on *item*. The footnote is marked with a markup *mark* moved by *offset* with respect to the marked music.

If *mark* is not given or specified as `\default`, it is replaced by an automatically generated sequence number. If *item* is a symbol list of form '`Grob`' or '`Context.Grob`', then grobs of that type will be marked at the current time step in the given context (default `Bottom`).

If *item* is music, the music will get a footnote attached to a grob immediately attached to the event, like `\tweak` does. For attaching a footnote to an *indirectly* caused grob, write `\single\footnote`, use *item* to specify the grob, and follow it with the music to annotate.

Like with `\tweak`, if you use a footnote on a following post-event, the `\footnote` command itself needs to be attached to the preceding note or rest as a post-event with `-`.

`\grace` [music] - *music* (music)
Insert *music* as grace notes.

`\grobdescriptions` (any type) - *descriptions* (list)
Create a context modification from *descriptions*, a list in the format of `all-grob-descriptions`.

`\harmonicByFret` [music] - *fret* (number) *music* (music)
Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at *fret*.

`\harmonicByRatio` [music] - *ratio* (number) *music* (music)
Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through *ratio*.

`\harmonicNote` [music] - *note* (music)
: Print *note* with a diamond-shaped note head.

`\harmonicsOn` [music]
: Set the default note head style to a diamond-shaped style.

`\hide` [music] - *item* (symbol list or music)
: Set *item*'s '`transparent`' property to `#t`, making it invisible while still retaining its dimensions.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`\incipit` [music] - *incipit-music* (music)
: Output *incipit-music* before the main staff as an indication of its appearance in the original music.

`\inherit-acceptability` [void] - *to* (symbol) *from* (symbol)
: When used in an output definition, will modify all context definitions such that context *to* is accepted as a child by all contexts that also accept *from*.

`\inStaffSegno` [music]
: Put the segno variant 'varsegno' at this position into the staff, compatible with the repeat command.

`\instrumentSwitch` [music] - *name* (string)
: Switch instrument to *name*, which must be predefined with `\addInstrumentDefinition`.

`\inversion` [music] - *around* (pitch) *to* (pitch) *music* (music)
: Invert *music* about *around* and transpose from *around* to *to*.

`\invertChords` [music] - *num* (integer) *music* (music)
: Invert any chords in *music* into their *num*-th position. (Chord inversions may be directed downwards using negative integers.)

`\keepWithTag` [music] - *tags* (symbol list or symbol) *music* (music)
: Include only elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

  Each tag may be declared as a member of at most one tag group (defined with `\tagGroup`). If none of a *music* element's tags share a tag group with one of the specified *tags*, the element is retained.

`\key` [music] - *tonic* [pitch] *pitch-alist* [list of number pairs]
: Set key to *tonic* and scale *pitch-alist*. If both are null, just generate `KeyChangeEvent`.

`\killCues` [music] - *music* (music)
: Remove cue notes from *music*.

`\label` [music] - *label* (symbol)
: Create *label* as a referrable label.

`\language` [void] - *language* (string)
: Set note names for language *language*.

`\languageRestore` [void]
: Restore a previously-saved pitchnames alist.

`\languageSaveAndChange` [void] - *language* (string)
: Store the previous pitchnames alist, and set a new one.

`\magnifyMusic` [music] - *mag* (positive number) *music* (music)
: Magnify the notation of *music* without changing the staff-size, using *mag* as a size factor. Stems, beams, slurs, ties, and horizontal spacing are adjusted automatically.

`\magnifyStaff` [music] - *mag* (positive number)
: Change the size of the staff, adjusting notation size and horizontal spacing automatically, using *mag* as a size factor.

`\makeClusters` [music] - *arg* (music)
: Display chords in *arg* as clusters.

`\makeDefaultStringTuning` [void] - *symbol* (symbol) *pitches* (list)
: This defines a string tuning *symbol* via a list of *pitches*. The *symbol* also gets registered in `defaultStringTunings` for documentation purposes.

`\mark` [music] - *label* [integer or markup]
: Make the music for the \mark command.

`\markupMap` [music] - *path* (symbol list or symbol) *markupfun* (markup-function) *music* (music)
: This applies the given markup function *markupfun* to all markup music properties matching *path* in *music*.

  For example,

```
\new Voice { g'2 c'' }
\addlyrics {
  \markupMap LyricEvent.text
             \markup \with-color #red \etc
             { Oh yes! }
}
```

`\modalInversion` [music] - *around* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Invert *music* about *around* using *scale* and transpose from *around* to *to*.

`\modalTranspose` [music] - *from* (pitch) *to* (pitch) *scale* (music) *music* (music)
: Transpose *music* from pitch *from* to pitch *to* using *scale*.

`\musicMap` [music] - *proc* (procedure) *mus* (music)
: Apply *proc* to *mus* and all of the music it contains.

`\noPageBreak` [music]
: Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\noPageTurn` [music]
: Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\octaveCheck` [music] - *pitch* (pitch)
: Octave check.

`\offset` [music] - *property* (symbol list or symbol) *offsets* (any type) *item* (key list or music)
: Offset the default value of *property* of *item* by *offsets*. If *item* is a string, the result is `\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`\omit` [music] - *item* (symbol list or music)
: Set *item*'s '`stencil`' property to `#f`, effectively omitting it without taking up space.

  If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`\once` [music] - *music* (music)
: Set `once` to `#t` on all layout instruction events in *music*. This will complain about music with an actual duration. As a special exception, if *music* might be the result of a `\tweak` command, no warning will be given in order to allow for `\once \propertyTweak` to work as both one-time override and proper tweak.

`\ottava` [music] - *octave* (integer)
: Set the octavation.

`\overrideProperty` [music] - *grob-property-path* (list of indexes or symbols) *value* (any type)
: Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well.

  As opposed to `\override` which overrides the context-dependent defaults with which a grob is created, this command uses `Output_property_engraver` at the grob acknowledge stage. This may be necessary for overriding values set after the initial grob creation.

`\overrideTimeSignatureSettings` [music] - *time-signature* (fraction, as pair) *base-moment* (fraction, as pair) *beat-structure* (list) *beam-exceptions* (list)
: Override `timeSignatureSettings` for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.

`\pageBreak` [music]
: Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.

`\pageTurn` [music]
: Force a page turn between two scores or top-level markups.

`\palmMute` [music] - *note* (music)
: Print *note* with a triangle-shaped note head.

`\palmMuteOn` [music]
: Set the default note head style to a triangle-shaped style.

`\parallelMusic` [void] - *voice-ids* (list) *music* (music)
: Define parallel music sequences, separated by '`|`' (bar check signs), and assign them to the identifiers provided in *voice-ids*.

  *voice-ids*: a list of music identifiers (symbols containing only letters)

  *music*: a music sequence, containing BarChecks as limiting expressions.

  Example:

```
\parallelMusic A,B,C {
  c c | d d | e e |
  d d | e e | f f |
}
<==>
A = { c c | d d }
B = { d d | e e }
C = { e e | f f }
```

  The last bar checks in a sequence are not copied to the result in order to facilitate ending the last entry at non-bar boundaries.

`\parenthesize` [music] - *arg* (music)
: Tag *arg* to be parenthesized.

`\partCombine` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and return a music expression containing simultaneous voices, where *part1* and *part2* are combined into one voice where appropriate. Optional *chord-range* sets the distance in steps between notes that may be combined into a chord or unison.

`\partCombineDown` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed downward.

`\partCombineForce` [music] - *type* [symbol]
: Override the part-combiner.

`\partCombineUp` [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)
: Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed upward.

`\partial` [music] - *dur* (duration)
: Make a partial measure.

`\phrasingSlurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for phrasing slurs.

`\pitchedTrill` [music] - *main-note* (music) *secondary-note* (music)
: Print a trill with *main-note* as the main note of the trill and print *secondary-note* as a stemless note head in parentheses.

`\pointAndClickOff` [void]
: Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

`\pointAndClickOn` [void]
: Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

`\pointAndClickTypes` [void] - *types* (symbol list or symbol)
: Set a type or list of types (such as `#'note-event`) for which point-and-click info is generated.

`\propertyOverride` [music] - *grob-property-path* (list of indexes or symbols) *value* (any type)
: Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\override` command.

`\propertyRevert` [music] - *grob-property-path* (list of indexes or symbols)
: Revert the grob property specified by *grob-property-path* to its previous value. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\revert` command.

`\propertySet` [music] - *property-path* (symbol list or symbol) *value* (any type)
: Set the context property specified by *property-path* to *value*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\set` command.

`\propertyTweak` [music] - *prop* (key list or symbol) *value* (any type) *item* (key list or music)
Add a tweak to the following *item*, usually music. This generally behaves like `\tweak` but will turn into an `\override` when *item* is a symbol list.

In that case, *item* specifies the grob path to override. This is mainly useful when using `\propertyTweak` as as a component for building other functions like `\omit`. It is not the default behavior for `\tweak` since many input strings in `\lyricmode` can serve equally as music or as symbols which causes surprising behavior when tweaking lyrics using the less specific semantics of `\propertyTweak`.

*prop* can contain additional elements in which case a nested property (inside of an alist) is tweaked.

`\propertyUnset` [music] - *property-path* (symbol list or symbol)
Unset the context property specified by *property-path*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\unset` command.

`\pushToTag` [music] - *tag* (symbol) *more* (music) *music* (music)
Add *more* to the front of `elements` of all music expressions in *music* that are tagged with *tag*.

`\quoteDuring` [music] - *what* (string) *main-music* (music)
Indicate a section of music to be quoted. *what* indicates the name of the quoted voice, as specified in an `\addQuote` command. *main-music* is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

`\raiseNote` [music] - *num* (integer) *music* (music)
Raise a note of any chords in *music*, in *num* position from below.

`\reduceChords` [music] - *music* (music)
Reduce chords contained in *music* to single notes, intended mainly for reusing music in RhythmicStaff. Does not reduce parallel music.

`\relative` [music] - *pitch* [pitch] *music* (music)
Make *music* relative to *pitch*. If *pitch* is omitted, the first note in *music* is given in absolute pitch.

`\removeWithTag` [music] - *tags* (symbol list or symbol) *music* (music)
Remove elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

`\resetRelativeOctave` [music] - *pitch* (pitch)
Set the octave inside a \relative section.

`\retrograde` [music] - *music* (music)
Return *music* in reverse order.

`\revertTimeSignatureSettings` [music] - *time-signature* (pair)
Revert `timeSignatureSettings` for time signatures of *time-signature*.

`\rightHandFinger` [post event] - *finger* (integer or markup)
Apply *finger* as a fingering indication.

`\scaleDurations` [music] - *fraction* (non-negative rational, fraction, or moment) *music* (music)
Multiply the duration of events in *music* by *fraction*.

`\settingsFrom` (any type) - *ctx* [symbol] *music* (music)
Take the layout instruction events from *music*, optionally restricted to those applying to context type *ctx*, and return a context modification duplicating their effect.

`\shape` [music] - *offsets* (list) *item* (key list or music)
: Offset control-points of *item* by *offsets*. The argument is a list of number pairs or list of such lists. Each element of a pair represents an offset to one of the coordinates of a control-point. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`\shiftDurations` [music] - *dur* (integer) *dots* (integer) *arg* (music)
: Change the duration of *arg* by adding *dur* to the `durlog` of *arg* and *dots* to the `dots` of *arg*.

`\single` [music] - *overrides* (music) *music* (music)
: Convert *overrides* to tweaks and apply them to *music*. This does not convert `\revert`, `\set` or `\unset`.

`\skip` [music] - *dur* (duration)
: Skip forward by *dur*.

`\slashedGrace` [music] - *music* (music)
: Create slashed graces (slashes through stems, but no slur) from the following music expression

`\slurDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for slurs.

`\spacingTweaks` [music] - *parameters* (list)
: Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

`\storePredefinedDiagram` [void] - *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair)
: Add predefined fret diagram defined by *diagram-definition* for the chord pitches *chord* and the stringTuning *tuning*.

`\stringTuning` (any type) - *chord* (music)
: Convert *chord* to a string tuning. *chord* must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

`\styledNoteHeads` [music] - *style* (symbol) *heads* (symbol list or symbol) *music* (music)
: Set *heads* in *music* to *style*.

`\tabChordRepeats` [music] - *event-types* [list] *music* (music)
: Walk through *music* putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(articulation-event)`.

`\tabChordRepetition` [void]
: Include the string and fingering information in a chord repetition. This function is deprecated; try using `\tabChordRepeats` instead.

`\tag` [music] - *tags* (symbol list or symbol) *music* (music)
: Tag the following *music* with *tags* and return the result, by adding the single symbol or symbol list *tags* to the `tags` property of *music*.

`\tagGroup` [void] - *tags* (symbol list)
: Define a tag group comprising the symbols in the symbol list *tags*. Tag groups must not overlap.

`\temporary` [music] - *music* (music)
: Make any `\override` in *music* replace an existing grob property value only temporarily, restoring the old value when a corresponding `\revert` is executed. This is achieved by clearing the '`pop-first`' property normally set on `\override`s.

  An `\override`/`\revert` sequence created by using `\temporary` and `\undo` on the same music containing overrides will cancel out perfectly or cause a warning.

  Non-property-related music is ignored, warnings are generated for any property-changing music that isn't an `\override`.

`\tieDashPattern` [music] - *dash-fraction* (number) *dash-period* (number)
: Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for ties.

`\time` [music] - *beat-structure* [number list] *fraction* (fraction, as pair)
: Set *fraction* as time signature, with optional number list *beat-structure* before it.

`\times` [music] - *fraction* (fraction, as pair) *music* (music)
: Scale *music* in time by *fraction*.

`\tocItem` [music] - *label* [symbol list or symbol] *text* (markup)
: Add a line to the table of contents, using the `tocItemMarkup` paper variable markup and assigning it to *label* if one is provided. If a hierarchy of labels is given, make the current item a child of the corresponding objects.

`\transpose` [music] - *from* (pitch) *to* (pitch) *music* (music)
: Transpose *music* from pitch *from* to pitch *to*.

`\transposedCueDuring` [music] - *what* (string) *dir* (direction) *pitch* (pitch) *main-music* (music)
: Insert notes from the part *what* into a voice called `cue`, using the transposition defined by *pitch*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

`\transposition` [music] - *pitch* (pitch)
: Set instrument transposition

`\tuplet` [music] - *ratio* (fraction, as pair) *tuplet-span* [duration] *music* (music)
: Scale the given *music* to tuplets. *ratio* is a fraction that specifies how many notes are played in place of the nominal value: it will be '`3/2`' for triplets, namely three notes being played in place of two. If the optional duration *tuplet-span* is specified, it is used instead of `tupletSpannerDuration` for grouping the tuplets. For example,

  ```
  \tuplet 3/2 4 { c8 c c c c c }
  ```

  will result in two groups of three tuplets, each group lasting for a quarter note.

`\tupletSpan` [music] - *tuplet-span* [duration]
: Set `tupletSpannerDuration`, the length into which `\tuplet` without an explicit '`tuplet-span`' argument of its own will group its tuplets, to the duration *tuplet-span*. To revert to the default of not subdividing the contents of a `\tuplet` command without explicit '`tuplet-span`', use

  ```
  \tupletSpan \default
  ```

`\tweak` [music] - *prop* (key list or symbol) *value* (any type) *music* (music)
: Add a tweak to the following *music*. Layout objects created by *music* get their property *prop* set to *value*. If *prop* has the form '`Grob.property`', like with

  ```
  \tweak Accidental.color #red cis'
  ```

an indirectly created grob ('`Accidental`' is caused by '`NoteHead`') can be tweaked; otherwise only directly created grobs are affected.

*prop* can contain additional elements in which case a nested property (inside of an alist) is tweaked.

If *music* is an '`event-chord`', every contained '`rhythmic-event`' is tweaked instead.

`\undo` [music] - *music* (music)
Convert `\override` and `\set` in *music* to `\revert` and `\unset`, respectively. Any reverts and unsets already in *music* cause a warning. Non-property-related music is ignored.

`\unfoldRepeats` [music] - *types* [symbol list or symbol] *music* (music)
Force `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`, if specified in the optional symbol-list *types*. The default for *types* is an empty list, which will force any of those commands in *music* to be interpreted as `\repeat unfold`. Possible entries are `volta`, `tremolo` or `percent`. Multiple entries are possible.

`\voices` [music] - *ids* (list of indexes or symbols) *music* (music)
Take the given key list of numbers (indicating the use of '`\voiceOne`'...) or symbols (indicating voice names, typically converted from strings by argument list processing) and assign the following `\\`-separated music to contexts according to that list. Named rather than numbered contexts can be used for continuing one voice (for the sake of spanners and lyrics), usually requiring a `\voiceOne`-style override at the beginning of the passage and a `\oneVoice` override at its end.

The default

```
<< ... \\ ... \\ ... >>
```

construct would correspond to

```
\voices 1,2,3 << ... \\ ... \\ ... >>
```
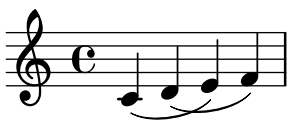
`\void` [void] - *arg* (any type)
Accept a scheme argument, return a void expression. Use this if you want to have a scheme expression evaluated because of its side-effects, but its value ignored.

`\withMusicProperty` [music] - *sym* (symbol) *val* (any type) *music* (music)
Set *sym* to *val* in *music*.

`\xNote` [music] - *note* (music)
Print *note* with a cross-shaped note head.

`\=` [post event] - *id* (index or symbol) *event* (post event)
This sets the `spanner-id` property of the following *event* to the given *id* (non-negative integer or symbol). This can be used to tell LilyPond how to connect overlapping or parallel slurs or phrasing slurs within a single `Voice`.

```
\fixed c' { c\=1( d\=2( e\=1) f\=2) }
```



## A.20 Identificatori delle modifiche di contesto

I seguenti comandi possono essere usati come modificatori di contesto all'interno di un blocco `\layout` o `\with`.

`\RemoveAllEmptyStaves`
: Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`, including those in the first system.
  - Sets grob property `remove-empty` in `Sezione ‘‘VerticalAxisGroup’’ in` *Guida al Funzionamento Interno* to `#t`.
  - Sets grob property `remove-first` in `Sezione ‘‘VerticalAxisGroup’’ in` *Guida al Funzionamento Interno* to `#t`.

`\RemoveEmptyStaves`
: Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`.
  - Sets grob property `remove-empty` in `Sezione ‘‘VerticalAxisGroup’’ in` *Guida al Funzionamento Interno* to `#t`.

## A.21 Tipi di predicati predefiniti

Predicates return `#t` when their argument is of the named type and `#f` if it isn't.

### R5RS primary predicates

Primary predicates can be applied to any expression. They can be used on their own as predicates for LilyPond functions. The predicates here are part of the Scheme standard R5RS.

| **Type predicate** | **Description** |
|---|---|
| `boolean?` | boolean |
| `char?` | character |
| `complex?` | complex number |
| `eof-object?` | end-of-file object |
| `input-port?` | input port |
| `integer?` | integer |
| `list?` | list *(use* `cheap-list?` *for faster processing)* |
| `null?` | null |
| `number?` | number |
| `output-port?` | output port |
| `pair?` | pair |
| `port?` | port |
| `procedure?` | procedure |
| `rational?` | rational number |
| `real?` | real number |
| `string?` | string |
| `symbol?` | symbol |
| `vector?` | vector |

### R5RS secondary predicates

Secondary predicates are only applicable to specific expressions (for example, to numbers). They will throw a type error when applied to expressions they are not intended for. The predicates here are part of the Scheme standard R5RS.

| **Type predicate** | **Description** |
|---|---|
| `char-alphabetic?` | alphabetic character |
| `char-lower-case?` | lower-case character |
| `char-numeric?` | numeric character |

| | |
|---|---|
| `char-upper-case?` | upper-case character |
| `char-whitespace?` | whitespace character |
| `even?` | even number |
| `exact?` | exact number |
| `inexact?` | inexact number |
| `negative?` | negative number |
| `odd?` | odd number |
| `positive?` | positive number |
| `zero?` | zero |

## Guile predicates

These predicates are defined by Guile but are not part of a Scheme standard.

| **Type predicate** | **Description** |
|---|---|
| `hash-table?` | hash table |

## LilyPond scheme predicates

These predicates are only available within LilyPond and defined in Scheme.

| **Type predicate** | **Description** |
|---|---|
| `boolean-or-symbol?` | boolean or symbol |
| `cheap-list?` | list *(use this instead of* `list?` *for faster processing)* |
| `color?` | color |
| `fraction?` | fraction, as pair |
| `grob-list?` | list of grobs |
| `index?` | non-negative integer |
| `integer-or-markup?` | integer or markup |
| `key?` | index or symbol |
| `key-list?` | list of indexes or symbols |
| `key-list-or-music?` | key list or music |
| `key-list-or-symbol?` | key list or symbol |
| `markup?` | markup |
| `markup-command-list?` | markup command list |
| `markup-list?` | markup list |
| `moment-pair?` | pair of moment objects |
| `number-list?` | number list |
| `number-or-grob?` | number or grob |
| `number-or-pair?` | number or pair |
| `number-or-string?` | number or string |
| `number-pair?` | pair of numbers |
| `number-pair-list?` | list of number pairs |
| `rational-or-procedure?` | an exact rational or procedure |
| `rhythmic-location?` | rhythmic location |
| `scale?` | non-negative rational, fraction, or moment |
| `scheme?` | any type |
| `string-or-music?` | string or music |
| `string-or-pair?` | string or pair |
| `string-or-symbol?` | string or symbol |
| `symbol-list?` | symbol list |
| `symbol-list-or-music?` | symbol list or music |
| `symbol-list-or-symbol?` | symbol list or symbol |

| | |
|---|---|
| `void?` | void |

## LilyPond exported predicates

These predicates are only available within LilyPond and usually defined in C++.

| **Type predicate** | **Description** |
|---|---|
| `ly:book?` | book |
| `ly:box?` | box |
| `ly:context?` | context |
| `ly:context-def?` | context definition |
| `ly:context-mod?` | context modification |
| `ly:dimension?` | dimension, in staff space |
| `ly:dir?` | direction |
| `ly:dispatcher?` | dispatcher |
| `ly:duration?` | duration |
| `ly:event?` | post event |
| `ly:font-metric?` | font metric |
| `ly:grob?` | graphical (layout) object |
| `ly:grob-array?` | array of grobs |
| `ly:grob-properties?` | grob properties |
| `ly:input-location?` | input location |
| `ly:item?` | item |
| `ly:iterator?` | iterator |
| `ly:lily-lexer?` | lily-lexer |
| `ly:lily-parser?` | lily-parser |
| `ly:listener?` | listener |
| `ly:moment?` | moment |
| `ly:music?` | music |
| `ly:music-function?` | music function |
| `ly:music-list?` | list of music objects |
| `ly:music-output?` | music output |
| `ly:otf-font?` | OpenType font |
| `ly:output-def?` | output definition |
| `ly:page-marker?` | page marker |
| `ly:pango-font?` | pango font |
| `ly:paper-book?` | paper book |
| `ly:paper-system?` | paper-system Prob |
| `ly:pitch?` | pitch |
| `ly:prob?` | property object |
| `ly:score?` | score |
| `ly:skyline?` | skyline |
| `ly:skyline-pair?` | pair of skylines |
| `ly:source-file?` | source file |
| `ly:spanner?` | spanner |
| `ly:spring?` | spring |
| `ly:stencil?` | stencil |
| `ly:stream-event?` | stream event |
| `ly:transform?` | coordinate transform |
| `ly:translator?` | translator |
| `ly:translator-group?` | translator group |
| `ly:undead?` | undead container |

`ly:unpure-pure-container?` unpure/pure container

## A.22 Funzioni Scheme

`ly:add-context-mod` *contextmods modification* [Funzione]
Adds the given context *modification* to the list *contextmods* of context modifications.

`ly:add-file-name-alist` *alist* [Funzione]
Add mappings for error messages from *alist*.

`ly:add-interface` *iface desc props* [Funzione]
Add a new grob interface. *iface* is the interface name, *desc* is the interface description, and *props* is the list of user-settable properties for the interface.

`ly:add-listener` *callback disp cl* [Funzione]
Add the single-argument procedure *callback* as listener to the dispatcher *disp*. Whenever *disp* hears an event of class *cl*, it calls *callback* with it.

`ly:add-option` *sym val description* [Funzione]
Add a program option *sym*. *val* is the default value and *description* is a string description.

`ly:all-grob-interfaces` [Funzione]
Return the hash table with all grob interface descriptions.

`ly:all-options` [Funzione]
Get all option settings in an alist.

`ly:all-stencil-expressions` [Funzione]
Return all symbols recognized as stencil expressions.

`ly:angle` *x y* [Funzione]
Calculates angle in degrees of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:assoc-get` *key alist default-value strict-checking* [Funzione]
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* [Funzione]
Set *grob* the parent of *grob-element* on all axes of *grob*.

`ly:basic-progress` *str rest* [Funzione]
A Scheme callable function to issue a basic progress message *str*. The message is formatted with `format` and *rest*.

`ly:beam-score-count` [Funzione]
count number of beam scores.

`ly:book?` *x* [Funzione]
Is *x* a `Book` object?

`ly:book-add-bookpart!` *book-smob book-part* [Funzione]
Add *book-part* to *book-smob* book part list.

`ly:book-add-score!` *book-smob score* [Funzione]
Add *score* to *book-smob* score list.

`ly:book-book-parts` *book* [Funzione]
Return book parts in *book*.

`ly:book-header` *book* [Funzione]
Return header in *book*.

`ly:book-paper` *book* [Funzione]
Return paper in *book*.

`ly:book-process` *book-smob default-paper default-layout output* [Funzione]
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* [Funzione]
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* [Funzione]
Return scores in *book*.

`ly:book-set-header!` *book module* [Funzione]
Set the book header.

`ly:box?` *x* [Funzione]
Is *x* a `Box` object?

`ly:bp` *num* [Funzione]
*num* bigpoints (1/72th inch).

`ly:bracket` *a iv t p* [Funzione]
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* [Funzione]
Send the stream event *ev* to the dispatcher *disp*.

`ly:camel-case->lisp-identifier` *name-sym* [Funzione]
Convert `FooBar_Bla` to `foo-bar-bla` style symbol.

`ly:chain-assoc-get` *key achain default-value strict-checking* [Funzione]
Return value for *key* from a list of alists *achain*. If no entry is found, return *default-value* or `#f` if *default-value* is not specified. With *strict-checking* set to `#t`, a programming_error is output in such cases.

`ly:check-expected-warnings` [Funzione]
Check whether all expected warnings have really been triggered.

`ly:cm` *num* [Funzione]
*num* cm.

`ly:command-line-code` [Funzione]
The Scheme code specified on command-line with `-e`.

`ly:command-line-options` [Funzione]
The Scheme options specified on command-line with `-d`.

`ly:connect-dispatchers` *to from* [Funzione]
Make the dispatcher *to* listen to events from *from*.

`ly:context?` *x* [Funzione]
Is *x* a `Context` object?

`ly:context-current-moment` *context* [Funzione]
Return the current moment of *context*.

`ly:context-def?` *x* [Funzione]
Is *x* a `Context_def` object?

`ly:context-def-lookup` *def sym val* [Funzione]
Return the value of *sym* in context definition *def* (e.g., `\Voice`). If no value is found, return *val* or `'()` if *val* is undefined. *sym* can be any of '`default-child`', '`consists`', '`description`', '`aliases`', '`accepts`', '`property-ops`', '`context-name`', '`group-type`'.

`ly:context-def-modify` *def mod* [Funzione]
Return the result of applying the context-mod *mod* to the context definition *def*. Does not change *def*.

`ly:context-event-source` *context* [Funzione]
Return `event-source` of context *context*.

`ly:context-events-below` *context* [Funzione]
Return a `stream-distributor` that distributes all events from *context* and all its subcontexts.

`ly:context-find` *context name* [Funzione]
Find a parent of *context* that has name or alias *name*. Return `#f` if not found.

`ly:context-grob-definition` *context name* [Funzione]
Return the definition of *name* (a symbol) within *context* as an alist.

`ly:context-id` *context* [Funzione]
Return the ID string of *context*, i.e., for `\context Voice = "one" ...` return the string `one`.

`ly:context-matched-pop-property` *context grob cell* [Funzione]
This undoes a particular `\override`, `\once \override` or `\once \revert` when given the specific alist pair to undo.

`ly:context-mod?` *x* [Funzione]
Is *x* a `Context_mod` object?

`ly:context-mod-apply!` *context mod* [Funzione]
Apply the context modification *mod* to *context*.

`ly:context-name` *context* [Funzione]
Return the name of *context*, i.e., for `\context Voice = "one" ...` return the symbol `Voice`.

`ly:context-now` *context* [Funzione]
Return `now-moment` of context *context*.

`ly:context-parent` *context* [Funzione]
Return the parent of *context*, `#f` if none.

`ly:context-property` *context sym def* [Funzione]
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* [Funzione]
Return the context above *context* where *name* is defined.

`ly:context-pushpop-property` *context grob eltprop val* [Funzione]
Do `\temporary` `\override` or `\revert` operation in *context*. The grob definition *grob* is extended with *eltprop* (if *val* is specified) or reverted (if unspecified).

`ly:context-set-property!` *context name val* [Funzione]
Set value of property *name* in context *context* to *val*.

`ly:context-unset-property` *context name* [Funzione]
Unset value of property *name* in context *context*.

`ly:debug` *str rest* [Funzione]
A Scheme callable function to issue a debug message *str*. The message is formatted with `format` and *rest*.

`ly:default-scale` [Funzione]
Get the global default scale.

`ly:dimension?` *d* [Funzione]
Is *d* a dimension? Used to distinguish length variables from normal numbers.

`ly:dir?` *s* [Funzione]
Is *s* a direction? Valid directions are `-1`, `0`, or `1`, where `-1` represents left or down, `1` represents right or up, and `0` represents a neutral direction.

`ly:directed` *direction magnitude* [Funzione]
Calculates an `(x . y)` pair with optional *magnitude* (defaulting to `1.0`) and *direction* specified either as an angle in degrees or a coordinate pair giving the direction. If *magnitude* is a pair, the respective coordinates are scaled independently, useful for ellipse drawings.

`ly:disconnect-dispatchers` *to from* [Funzione]
Stop the dispatcher *to* listening to events from *from*.

`ly:dispatcher?` *x* [Funzione]
Is *x* a `Dispatcher` object?

`ly:duration?` *x* [Funzione]
Is *x* a `Duration` object?

`ly:duration<?` *p1 p2* [Funzione]
Is *p1* shorter than *p2*?

`ly:duration->string` *dur* [Funzione]
Convert *dur* to a string.

`ly:duration-dot-count` *dur* [Funzione]
Extract the dot count from *dur*.

`ly:duration-factor` *dur* [Funzione]
Extract the compression factor from *dur*. Return it as a pair.

`ly:duration-length` *dur* [Funzione]
The length of the duration as a `moment`.

`ly:duration-log` *dur* [Funzione]
Extract the duration log from *dur*.

`ly:duration-scale` *dur* [Funzione]
Extract the compression factor from *dur*. Return it as a rational.

`ly:effective-prefix` [Funzione]
Return effective prefix.

`ly:encode-string-for-pdf` *str* [Funzione]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).

`ly:engraver-announce-end-grob` *engraver grob cause* [Funzione]
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* [Funzione]
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* [Funzione]
A Scheme callable function to issue the error *str*. The error is formatted with `format` and *rest*.

`ly:event?` *obj* [Funzione]
Is *obj* a proper (non-rhythmic) event object?

`ly:event-deep-copy` *m* [Funzione]
Copy *m* and all sub expressions of *m*.

`ly:event-property` *sev sym val* [Funzione]
Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or `'()` if *val* is not specified.

`ly:event-set-property!` *ev sym val* [Funzione]
Set property *sym* in event *ev* to *val*.

`ly:expand-environment` *str* [Funzione]
Expand `$VAR` and `${VAR}` in *str*.

`ly:expect-warning` *str rest* [Funzione]
A Scheme callable function to register a warning to be expected and subsequently suppressed. If the warning is not encountered, a warning about the missing warning will be shown. The message should be translated with `(_ ...)` and changing parameters given after the format string.

`ly:extract-subfont-from-collection` *collection-file-name idx subfont-file-name* [Funzione]
Extract the subfont of index *idx* in TrueType collection (TTC) or OpenType/CFF collection (OTC) file *collection_file_name* and write it to file *subfont_file_name*.

`ly:find-file` *name* [Funzione]
Return the absolute file name of *name*, or `#f` if not found.

`ly:font-config-add-directory` *dir* [Funzione]
Add directory *dir* to FontConfig.

`ly:font-config-add-font` *font* [Funzione]
Add font *font* to FontConfig.

`ly:font-config-display-fonts` [Funzione]
Dump a list of all fonts visible to FontConfig.

`ly:font-config-get-font-file` *name* [Funzione]
Get the file for font *name*.

`ly:font-design-size` *font* [Funzione]
Given the font metric *font*, return the design size, relative to the current output-scale.

`ly:font-file-name` *font* [Funzione]
Given the font metric *font*, return the corresponding file name.

`ly:font-get-glyph` *font name* [Funzione]
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* [Funzione]
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* [Funzione]
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* [Funzione]
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* [Funzione]
Given the font metric *font*, return the magnification, relative to the current output-scale.

`ly:font-metric?` *x* [Funzione]
Is *x* a `Font_metric` object?

`ly:font-name` *font* [Funzione]
Given the font metric *font*, return the corresponding name.

`ly:font-sub-fonts` *font* [Funzione]
Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

`ly:format` *str rest* [Funzione]
LilyPond specific format, supporting `~a` and `~[0-9]f`. Basic support for `~s` is also provided.

`ly:format-output` *context* [Funzione]
Given a global context in its final state, process it and return the `Music_output` object in its final state.

`ly:generic-bound-extent` *grob common* [Funzione]
Determine the extent of *grob* relative to *common* along the X axis, finding its extent as a bound when it a has `bound-alignment-interfaces` property list set and otherwise the full extent.

`ly:get-all-function-documentation` [Funzione]
Get a hash table with all LilyPond Scheme extension functions.

`ly:get-all-translators` [Funzione]
Return a list of all translator objects that may be instantiated.

`ly:get-cff-offset` *font-file-name idx* [Funzione]
Get the offset of 'CFF' table for *font_file_name*, returning it as an integer. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:get-context-mods` *contextmod* [Funzione]
Returns the list of context modifications stored in *contextmod*.

`ly:get-font-format` *font-file-name idx* [Funzione]
Get the font format for *font_file_name*, returning it as a symbol. The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:get-option` *var* [Funzione]
Get a global option setting.

`ly:get-spacing-spec` *from-scm to-scm* [Funzione]
Return the spacing spec going between the two given grobs, *from_scm* and *to_scm*.

`ly:get-undead` *undead* [Funzione]
Get back object from *undead*.

`ly:gettext` *original* [Funzione]
A Scheme wrapper function for `gettext`.

`ly:grob?` *x* [Funzione]
Is *x* a `Grob` object?

`ly:grob-alist-chain` *grob global* [Funzione]
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* [Funzione]
Is *x* a `Grob_array` object?

`ly:grob-array->list` *grob-arr* [Funzione]
Return the elements of *grob-arr* as a Scheme list.

`ly:grob-array-length` *grob-arr* [Funzione]
Return the length of *grob-arr*.

`ly:grob-array-ref` *grob-arr index* [Funzione]
Retrieve the *index*th element of *grob-arr*.

`ly:grob-basic-properties` *grob* [Funzione]
Get the immutable properties of *grob*.

`ly:grob-chain-callback` *grob proc sym* [Funzione]
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* [Funzione]
Find the common refpoint of *grob* and *other* for *axis*.

`ly:grob-common-refpoint-of-array` *grob others axis* [Funzione]
Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

`ly:grob-default-font` *grob* [Funzione]
Return the default font for grob *grob*.

`ly:grob-extent` *grob refp axis* [Funzione]
Get the extent in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-get-vertical-axis-group-index` *grob* [Funzione]
Get the index of the vertical axis group the grob *grob* belongs to; return `-1` if none is found.

`ly:grob-interfaces` *grob* [Funzione]
Return the interfaces list of grob *grob*.

`ly:grob-layout` *grob* [Funzione]
Get `\layout` definition from grob *grob*.

`ly:grob-object` *grob sym* [Funzione]
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* [Funzione]
Return the unbroken original grob of *grob*.

`ly:grob-parent` *grob axis* [Funzione]
Get the parent of *grob*. *axis* is 0 for the X-axis, 1 for the Y-axis.

`ly:grob-pq<?` *a b* [Funzione]
Compare two grob priority queue entries. This is an internal function.

`ly:grob-properties?` *x* [Funzione]
Is *x* a `Grob_properties` object?

`ly:grob-property` *grob sym val* [Funzione]
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* [Funzione]
Return the value for property *sym* of *grob*, but do not process callbacks.

`ly:grob-pure-height` *grob refp beg end val* [Funzione]
Return the pure height of *grob* given refpoint *refp*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-pure-property` *grob sym beg end val* [Funzione]
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or `'()` if *val* is not specified.

`ly:grob-relative-coordinate` *grob refp axis* [Funzione]
Get the coordinate in *axis* direction of *grob* relative to the grob *refp*.

`ly:grob-robust-relative-extent` *grob refp axis* [Funzione]
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* [Funzione]
Compare two grobs by script priority. For internal use.

`ly:grob-set-nested-property!` *grob symlist val* [Funzione]
Set nested property *symlist* in grob *grob* to value *val*.

`ly:grob-set-object!` *grob sym val* [Funzione]
Set *sym* in grob *grob* to value *val*.

`ly:grob-set-parent!` *grob axis parent-grob* [Funzione]
Set *parent-grob* the parent of grob *grob* in axis *axis*.

`ly:grob-set-property!` *grob sym val* [Funzione]
Set *sym* in grob *grob* to value *val*.

`ly:grob-spanned-rank-interval` *grob* [Funzione]
Returns a pair with the `rank` of the furthest left column and the `rank` of the furthest right column spanned by `grob`.

`ly:grob-staff-position` *sg* [Funzione]
Return the Y-position of *sg* relative to the staff.

`ly:grob-suicide!` *grob* [Funzione]
Kill *grob*.

`ly:grob-system` *grob* [Funzione]
Return the system grob of *grob*.

`ly:grob-translate-axis!` *grob d a* [Funzione]
Translate *grob* on axis *a* over distance *d*.

`ly:grob-vertical<?` *a b* [Funzione]
Does *a* lie above *b* on the page?

`ly:gulp-file` *name size* [Funzione]
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:has-glyph-names?` *font-file-name idx* [Funzione]
Does the font for *font_file_name* have glyph names? The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.

`ly:hash-table-keys` *tab* [Funzione]
Return a list of keys in *tab*.

`ly:inch` *num* [Funzione]
*num* inches.

`ly:input-both-locations` *sip* [Funzione]
Return input location in *sip* as `(file-name first-line first-column last-line last-column)`.

`ly:input-file-line-char-column` *sip* [Funzione]
Return input location in *sip* as `(file-name line char column)`.

`ly:input-location?` *x* [Funzione]
Is *x* a `Input` object?

`ly:input-message` *sip msg rest* [Funzione]
Print *msg* as a GNU compliant error message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:input-warning` *sip msg rest* [Funzione]
Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to `format`'s argument, using *rest*.

`ly:interpret-music-expression` *mus ctx* [Funzione]
Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.

`ly:intlog2` *d* [Funzione]
The 2-logarithm of $1/d$.

`ly:item?` *g* [Funzione]
Is *g* an `Item` object?

`ly:item-break-dir` *it* [Funzione]
The break status direction of item *it*. `-1` means end of line, `0` unbroken, and `1` beginning of line.

`ly:item-get-column` *it* [Funzione]
Return the `PaperColumn` or `NonMusicalPaperColumn` associated with this `Item`.

`ly:iterator?` *x* [Funzione]
Is *x* a `Music_iterator` object?

`ly:length` *x y* [Funzione]
Calculates magnitude of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.

`ly:lily-lexer?` *x* [Funzione]
Is *x* a `Lily_lexer` object?

`ly:lily-parser?` *x* [Funzione]
Is *x* a `Lily_parser` object?

`ly:line-interface::line` *grob startx starty endx endy* [Funzione]
Make a line using layout information from grob *grob*.

`ly:listened-event-class?` *disp cl* [Funzione]
Does *disp* listen to any event type in the list *cl*?

`ly:listened-event-types` *disp* [Funzione]
Return a list of all event types that *disp* listens to.

`ly:listener?` *x* [Funzione]
Is *x* a `Listener` object?

`ly:make-book` *paper header scores* [Funzione]
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.

`ly:make-book-part` *scores* [Funzione]
Make a `\bookpart` containing `\scores`.

`ly:make-context-mod` *mod-list* [Funzione]
Creates a context modification, optionally initialized via the list of modifications *mod-list*.

`ly:make-dispatcher` [Funzione]
Return a newly created dispatcher.

`ly:make-duration` *length dotcount num den* [Funzione]
*length* is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.

The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.

A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.

`ly:make-global-context` *output-def* [Funzione]
Set up a global interpretation context, using the output block *output-def*. The context is returned.

`ly:make-global-translator` *global* [Funzione]
Create a translator group and connect it to the global context *global*. The translator group is returned.

`ly:make-grob-properties` *alist* [Funzione]
This packages the given property list *alist* in a grob property container stored in a context property with the name of a grob.

`ly:make-moment` *m g gn gd* [Funzione]
Create the moment with rational main timing *m*, and optional grace timing *g*.

A *moment* is a point in musical time. It consists of a pair of rationals (*m*, *g*), where *m* is the timing for the main notes, and *g* the timing for grace notes. In absence of grace notes, *g* is zero.

For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.

`ly:make-music` *props* [Funzione]
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* [Funzione]
Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*'s cdr is a list containing either `ly:music?` predicates or other type predicates. Its car is the syntax function to call.

`ly:make-music-relative!` *music pitch* [Funzione]
Make *music* relative to *pitch*, return final pitch.

`ly:make-output-def` [Funzione]
Make an output definition.

`ly:make-page-label-marker` *label* [Funzione]
Return page marker with label *label*.

`ly:make-page-permission-marker` *symbol permission* [Funzione]
Return page marker with page breaking and turning permissions.

`ly:make-pango-description-string` *chain size* [Funzione]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.

`ly:make-paper-outputter` *port alist default-callback* [Funzione]
Create an outputter dumping to `port`. `alist` should map symbols to procedures. See `output-ps.scm` for an example. If `default_callback` is given, it is called for unsupported expressions

`ly:make-pitch` *octave note alter* [Funzione]
*octave* is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.

`ly:make-prob` *type init rest* [Funzione]
Create a `Prob` object.

`ly:make-rotation` *angle center* [Funzione]
Make a transform rotating by *angle* in degrees. If *center* is given as a pair of coordinates, it is the center of the rotation, otherwise the rotation is around (0 . 0).

`ly:make-scale` *steps* [Funzione]
Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.

`ly:make-scaling` *scale scaley* [Funzione]
Create a scaling transform from argument *scale* and optionally *scaley*. When both arguments are given, they must be real and give the scale in *x* and *y* direction. If only *scale* is given, it may also be complex to indicate a scaled rotation in the manner of complex number rotations, or a pair of reals for specifying different scales in *x* and *y* direction like with the first calling convention.

`ly:make-score` *music* [Funzione]
Return score with *music* encapsulated in it.

`ly:make-spring` *ideal min-dist* [Funzione]
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.

`ly:make-stencil` *expr xext yext* [Funzione]
Stencils are device independent output expressions. They carry two pieces of information:

1. A specification of how to print this object. This specification is processed by the output backends, for example `scm/output-ps.scm`.
2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use `empty-interval` as its value), it is taken to be empty.

`ly:make-stream-event` *cl proplist* [Funzione]
Create a stream event of class *cl* with the given mutable property list.

`ly:make-transform` *xx yx xy yy x0 y0* [Funzione]
Create a transform. Without options, it is an identity transform. Given four arguments *xx*, *yx*, *xy*, and *yy*, it is a linear transform, given six arguments (with *x0* and *y0* last), it is an affine transform. Transforms can be called as functions on other transforms (concatening them) or on points given either as complex number or real number pair. See also `ly:make-rotation`, `ly:make-scaling`, and `ly:make-translation`.

`ly:make-translation` *x y* [Funzione]
Make a transform translating by *x* and *y*. If only *x* is given, it can also be a complex number or a pair of numbers indicating the offset to use.

`ly:make-undead` *object* [Funzione]
This packages *object* in a manner that keeps it from triggering "Parsed object should be dead" messages.

`ly:make-unpure-pure-container` *unpure pure* [Funzione]
Make an unpure-pure container. *unpure* should be an unpure expression, and *pure* should be a pure expression. If *pure* is omitted, the value of *unpure* will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.

`ly:message` *str rest* [Funzione]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.

`ly:minimal-breaking` *pb* [Funzione]
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* [Funzione]
*num* mm.

`ly:module->alist` *mod* [Funzione]
Dump the contents of module *mod* as an alist.

`ly:module-copy` *dest src* [Funzione]
Copy all bindings from module *src* into *dest*.

`ly:modules-lookup` *modules sym def* [Funzione]
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* [Funzione]
Is *x* a `Moment` object?

`ly:moment<?` *a b* [Funzione]
Compare two moments.

`ly:moment-add` *a b* [Funzione]
Add two moments.

`ly:moment-div` *a b* [Funzione]
Divide two moments.

`ly:moment-grace` *mom* [Funzione]
Extract grace timing as a rational number from *mom*.

`ly:moment-grace-denominator` *mom* [Funzione]
Extract denominator from grace timing.

`ly:moment-grace-numerator` *mom* [Funzione]
Extract numerator from grace timing.

`ly:moment-main` *mom* [Funzione]
Extract main timing as a rational number from *mom*.

`ly:moment-main-denominator` *mom* [Funzione]
Extract denominator from main timing.

`ly:moment-main-numerator` *mom* [Funzione]
Extract numerator from main timing.

`ly:moment-mod` *a b* [Funzione]
Modulo of two moments.

`ly:moment-mul` *a b* [Funzione]
Multiply two moments.

`ly:moment-sub` *a b* [Funzione]
Subtract two moments.

`ly:music?` *obj* [Funzione]
Is *obj* a music object?

`ly:music-compress` *m factor* [Funzione]
Compress music object *m* by scale *factor*.

`ly:music-deep-copy` *m origin* [Funzione]
Copy *m* and all sub expressions of *m*. *m* may be an arbitrary type; cons cells and music are copied recursively. If *origin* is given, it is used as the origin for one level of music by calling `ly:set-origin!` on the copy.

`ly:music-duration-compress` *mus fact* [Funzione]
Compress *mus* by factor *fact*, which is a `Moment`.

`ly:music-duration-length` *mus* [Funzione]
Extract the duration field from *mus* and return the length.

`ly:music-function?` *x* [Funzione]
Is *x* a `Music_function` object?

`ly:music-function-extract` *x* [Funzione]
Return the Scheme function inside *x*.

`ly:music-function-signature` *x* [Funzione]
Return the function signature inside *x*.

`ly:music-length` *mus* [Funzione]
Get the length of music expression *mus* and return it as a `Moment` object.

`ly:music-list?` *lst* [Funzione]
Is *lst* a list of music objects?

`ly:music-mutable-properties` *mus* [Funzione]
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* [Funzione]
Is *x* a `Music_output` object?

`ly:music-property` *mus sym val* [Funzione]
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* [Funzione]
Set property *sym* in music expression *mus* to *val*.

`ly:music-transpose` *m p* [Funzione]
Transpose *m* such that central C is mapped to *p*. Return *m*.

`ly:note-column-accidentals` *note-column* [Funzione]
Return the `AccidentalPlacement` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-column-dot-column` *note-column* [Funzione]
Return the `DotColumn` grob from *note-column* if any, or `SCM_EOL` otherwise.

`ly:note-head::stem-attachment` *font-metric glyph-name* [Funzione]
Get attachment in *font-metric* for attaching a stem to notehead *glyph-name*.

`ly:number->string` *s* [Funzione]
Convert *s* to a string without generating many decimals.

`ly:one-line-auto-height-breaking` *pb* [Funzione]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.

`ly:one-line-breaking` *pb* [Funzione]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.

`ly:one-page-breaking` *pb* [Funzione]
Put each score on a single page. The paper-height settings are modified so each score fits on one page, and the height of the page matches the height of the full score.

`ly:optimal-breaking` *pb* [Funzione]
Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in bother vertical and horizontal spacing.

`ly:option-usage` *port* [Funzione]
Print `ly:set-option` usage. Optional *port* argumentfor the destination defaults to current output port.

`ly:otf->cff` *otf-file-name idx* [Funzione]
Convert the contents of an OTF file to a CFF file, returning it as a string. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.

`ly:otf-font?` *font* [Funzione]
Is *font* an OpenType font?

`ly:otf-font-glyph-info` *font glyph* [Funzione]
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* [Funzione]
Extract a table *tag* from *font*. Return empty string for non-existent *tag*.

`ly:otf-glyph-count` *font* [Funzione]
Return the number of glyphs in *font*.

`ly:otf-glyph-list` *font* [Funzione]
Return a list of glyph names for *font*.

`ly:output-def?` *x* [Funzione]
Is *x* a `Output_def` object?

`ly:output-def-clone` *def* [Funzione]
Clone output definition *def*.

`ly:output-def-lookup` *def sym val* [Funzione]
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* [Funzione]
Return the parent output definition of *def*.

`ly:output-def-scope` *def* [Funzione]
Return the variable scope inside *def*.

`ly:output-def-set-variable!` *def sym val* [Funzione]
Set an output definition *def* variable *sym* to *val*.

`ly:output-description` *output-def* [Funzione]
Return the description of translators in *output-def*.

`ly:output-find-context-def` *output-def context-name* [Funzione]
Return an alist of all context defs (matching *context-name* if given) in *output-def*.

`ly:output-formats` [Funzione]
Formats passed to `--format` as a list of strings, used for the output.

`ly:outputter-close` *outputter* [Funzione]
Close port of *outputter*.

`ly:outputter-dump-stencil` *outputter stencil* [Funzione]
Dump stencil *expr* onto *outputter*.

`ly:outputter-dump-string` *outputter str* [Funzione]
Dump *str* onto *outputter*.

`ly:outputter-output-scheme` *outputter expr* [Funzione]
Output *expr* to the paper outputter.

`ly:outputter-port` *outputter* [Funzione]
Return output port for *outputter*.

`ly:page-marker?` *x* [Funzione]
Is *x* a `Page_marker` object?

`ly:page-turn-breaking` *pb* [Funzione]
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* [Funzione]
Is *f* a pango font?

`ly:pango-font-physical-fonts` *f* [Funzione]
Return alist of `(ps-name file-name font-index)` lists for Pango font *f*.

`ly:paper-book?` *x* [Funzione]
Is *x* a `Paper_book` object?

`ly:paper-book-header` *pb* [Funzione]
Return the header definition (`\header`) in `Paper_book` object *pb*.

`ly:paper-book-pages` *pb* [Funzione]
Return pages in `Paper_book` object *pb*.

`ly:paper-book-paper` *pb* [Funzione]
Return the paper output definition (`\paper`) in `Paper_book` object *pb*.

`ly:paper-book-performances` *pb* [Funzione]
Return performances in `Paper_book` object *pb*.

`ly:paper-book-scopes` *pb* [Funzione]
Return scopes in `Paper_book` object *pb*.

`ly:paper-book-systems` *pb* [Funzione]
Return systems in `Paper_book` object *pb*.

`ly:paper-column::break-align-width` *col align-syms* [Funzione]
Determine the extent along the X-axis of a grob used for break-alignment organized by column *col*. The grob is specified by *align-syms*, which contains either a single `break-align-symbol` or a list of such symbols.

`ly:paper-column::print` [Funzione]
Optional stencil for `PaperColumn` or`NonMusicalPaperColumn`. Draws the `rank number` of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.

`ly:paper-fonts` *def* [Funzione]
Return a list containing the fonts from output definition *def* (e.g., `\paper`).

`ly:paper-get-font` *def chain* [Funzione]
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* [Funzione]
Return the value of variable *sym* in output definition *def* as a double.

`ly:paper-outputscale` *def* [Funzione]
Return the output-scale for output definition *def*.

`ly:paper-score-paper-systems` *paper-score* [Funzione]
Return vector of `paper_system` objects from *paper-score*.

`ly:paper-system?` *obj* [Funzione]
Is *obj* a C++ `Prob` object of type `paper-system`?

`ly:paper-system-minimum-distance` *sys1 sys2* [Funzione]
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* [Funzione]
Parse a single `.ly` file. Upon failure, throw `ly-file-failed` key.

`ly:parse-string-expression` *parser-smob ly-code filename line* [Funzione]
Parse the string *ly-code* with *parser-smob*. Return the contained music expression. *filename* and *line* are optional source indicators.

`ly:parsed-undead-list!` [Funzione]
Return the list of objects that have been found live that should have been dead, and clear that list.

`ly:parser-clear-error` *parser* [Funzione]
Clear error flag for *parser*, defaulting to current parser.

`ly:parser-clone` *closures location* [Funzione]
Return a clone of current parser. An association list of port positions to closures can be specified in *closures* in order to have `$` and `#` interpreted in their original lexical environment. If *location* is a valid location, it becomes the source of all music expressions inside.

`ly:parser-define!` *symbol val* [Funzione]
Bind *symbol* to *val* in current parser's module.

`ly:parser-error` *msg input* [Funzione]
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.

`ly:parser-has-error?` *parser* [Funzione]
Does *parser* (defaulting to current parser) have an error flag?

`ly:parser-include-string` *ly-code* [Funzione]
Include the string *ly-code* into the input stream for current parser. Can only be used in immediate Scheme expressions (`$` instead of `#`).

`ly:parser-lexer` *parser* [Funzione]
Return the lexer for *parser*, defaulting to current parser

`ly:parser-lookup` *symbol* [Funzione]
Look up *symbol* in current parser's module. Return `'()` if not defined.

`ly:parser-output-name` *parser* [Funzione]
Return the base name of the output file. If `parser` is left off, use currently active parser.

`ly:parser-parse-string` *parser-smob ly-code* [Funzione]
Parse the string *ly-code* with *parser-smob*. Upon failure, throw `ly-file-failed` key.

`ly:parser-set-note-names` *names* [Funzione]
Replace current note names in parser. *names* is an alist of symbols. This only has effect if the current mode is notes.

`ly:performance-headers` *performance* [Funzione]
Return the list of headers with the innermost first.

`ly:performance-write` *performance filename name* [Funzione]
Write *performance* to *filename* storing *name* as the name of the performance in the file metadata.

`ly:pitch?` *x* [Funzione]
Is *x* a `Pitch` object?

`ly:pitch<?` *p1 p2* [Funzione]
Is *p1* lexicographically smaller than *p2*?

`ly:pitch-alteration` *pp* [Funzione]
Extract the alteration from pitch *pp*.

`ly:pitch-diff` *pitch root* [Funzione]
Return pitch *delta* such that *root* transposed by *delta* equals *pitch*.

`ly:pitch-negate` *p* [Funzione]
Negate *p*.

`ly:pitch-notename` *pp* [Funzione]
Extract the note name from pitch *pp*.

`ly:pitch-octave` *pp* [Funzione]
Extract the octave from pitch *pp*.

`ly:pitch-quartertones` *pp* [Funzione]
Calculate the number of quarter tones of *pp* from middle C.

`ly:pitch-semitones` *pp* [Funzione]
Calculate the number of semitones of *pp* from middle C.

`ly:pitch-steps` *p* [Funzione]
Number of steps counted from middle C of the pitch *p*.

`ly:pitch-tones` *pp* [Funzione]
Calculate the number of tones of *pp* from middle C as a rational number.

`ly:pitch-transpose` *p delta* [Funzione]
Transpose *p* by the amount *delta*, where *delta* is relative to middle C.

`ly:pointer-group-interface::add-grob` *grob sym grob-element* [Funzione]
Add *grob-element* to *grob*'s *sym* grob array.

`ly:position-on-line?` *sg spos* [Funzione]
Return whether *spos* is on a line of the staff associated with the grob *sg* (even on an extender line).

`ly:prob?` *x* [Funzione]
Is *x* a `Prob` object?

`ly:prob-immutable-properties` *prob* [Funzione]
Retrieve an alist of immutable properties.

`ly:prob-mutable-properties` *prob* [Funzione]
Retrieve an alist of mutable properties.

`ly:prob-property` *prob sym val* [Funzione]
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* [Funzione]
Is boolean prop *sym* of *sym* set?

`ly:prob-set-property!` *obj sym value* [Funzione]
Set property *sym* of *obj* to *value*.

`ly:prob-type?` *obj type* [Funzione]
Is *obj* the specified prob-type?

`ly:programming-error` *str rest* [Funzione]
A Scheme callable function to issue the internal warning *str*. The message is formatted with `format` and *rest*.

`ly:progress` *str rest* [Funzione]
A Scheme callable function to print progress *str*. The message is formatted with `format` and *rest*.

`ly:property-lookup-stats` *sym* [Funzione]
Return hash table with a property access corresponding to *sym*. Choices are `prob`, `grob`, and `context`.

`ly:protects` [Funzione]
Return hash of protected objects.

`ly:pt` *num* [Funzione]
*num* printer points.

`ly:pure-call` *data grob start end rest* [Funzione]
Convert property *data* (unpure-pure container or procedure) to value in a pure context defined by *grob*, *start*, *end*, and possibly *rest* arguments.

`ly:randomize-rand-seed` [Funzione]
Randomize C random generator.

`ly:register-stencil-expression` *symbol* [Funzione]
Add *symbol* as head of a stencil expression.

`ly:register-translator` *creator name description* [Funzione]
Register a translator *creator* (usually a descriptive alist or a function/closure returning one when given a context argument) with the given symbol *name* and the given *description* alist.

`ly:relative-group-extent` *elements common axis* [Funzione]
Determine the extent of *elements* relative to *common* in the *axis* direction.

`ly:rename-file` *oldname newname* [Funzione]
Rename *oldname* to *newname*. In contrast to Guile's rename-file, this replaces the destination if it already exists. On Windows, fall back to copying the file contents if *newname* cannot be deleted.

`ly:reset-all-fonts` [Funzione]
Forget all about previously loaded fonts.

`ly:round-filled-box` *xext yext blot* [Funzione]
Make a `Stencil` object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.

`ly:round-filled-polygon` *points blot extroversion* [Funzione]
Make a `Stencil` object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*. Optional*extroversion* shifts the outline outward, with thedefault of `-1.0` keeping the outer boundary ofthe outline just inside of the polygon.

`ly:run-translator` *mus output-def* [Funzione]
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* [Funzione]
Is *x* a `Score` object?

`ly:score-add-output-def!` *score def* [Funzione]
Add an output definition *def* to *score*.

`ly:score-embedded-format` *score layout* [Funzione]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

`ly:score-error?` *score* [Funzione]
Was there an error in the score?

`ly:score-header` *score* [Funzione]
Return score header.

`ly:score-music` *score* [Funzione]
Return score music.

`ly:score-output-defs` *score* [Funzione]
All output definitions in a score.

`ly:score-set-header!` *score module* [Funzione]
Set the score header.

`ly:separation-item::print` [Funzione]
Optional stencil for `PaperColumn` or`NonMusicalPaperColumn`. Draws the `horizontal-skylines` of each `PaperColumn`, showing the shapes used to determine the minimum distances between `PaperColumns` at the note-spacing step, before staves have been spaced (vertically) on the page.

`ly:set-default-scale` *scale* [Funzione]
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* [Funzione]
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* [Funzione]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.

`ly:set-option` *var val* [Funzione]
Set a program option.

`ly:set-origin!` *m origin* [Funzione]
This sets the origin given in *origin* to *m*. *m* will typically be a music expression or a list of music. List structures are searched recursively, but recursion stops at the changed music expressions themselves. *origin* is generally of type `ly:input-location?`, defaulting to `(*location*)`. Other valid values for `origin` are a music expression which is then used as the source of location information, or `#f` or `'()` in which case no action is performed. The return value is *m* itself.

`ly:set-property-cache-callback` *cb* [Funzione]
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:skyline?` *x* [Funzione]
Is *x* a `Skyline` object?

`ly:skyline-empty?` *sky* [Funzione]
Return whether *sky* is empty.

`ly:skyline-pair?` *x* [Funzione]
Is *x* a `Skyline_pair` object?

`ly:slur-score-count` [Funzione]
count number of slur scores.

`ly:smob-protects` [Funzione]
Return LilyPond's internal smob protection list.

`ly:solve-spring-rod-problem` *springs rods length ragged* [Funzione]
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* [Funzione]
Is *x* a `Source_file` object?

`ly:source-files` *parser-smob* [Funzione]
A list of input files that have been opened up to here, including the files that have been closed already. a PARSER may optionally be specified.

`ly:spanner?` *g* [Funzione]
Is *g* a spanner object?

`ly:spanner-bound` *spanner dir* [Funzione]
Get one of the bounds of *spanner*. *dir* is `-1` for left, and `1` for right.

`ly:spanner-broken-into` *spanner* [Funzione]
Return broken-into list for *spanner*.

`ly:spanner-set-bound!` *spanner dir item* [Funzione]
Set grob *item* as bound in direction *dir* for *spanner*.

`ly:spawn` *command rest* [Funzione]
Simple interface to g_spawn_sync *str*. The error is formatted with `format` and *rest*.

`ly:spring?` *x* [Funzione]
Is *x* a `Spring` object?

`ly:spring-set-inverse-compress-strength!` *spring strength* [Funzione]
Set the inverse compress *strength* of *spring*.

`ly:spring-set-inverse-stretch-strength!` *spring strength* [Funzione]
Set the inverse stretch *strength* of *spring*.

`ly:staff-symbol-line-thickness` *grob* [Funzione]
Returns the current staff-line thickness in the staff associated with *grob*, expressed as a multiple of the current staff-space height.

`ly:staff-symbol-staff-radius` *grob* [Funzione]
Returns the radius of the staff associated with *grob*.

`ly:staff-symbol-staff-space` *grob* [Funzione]
Returns the current staff-space height in the staff associated with *grob*, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

`ly:start-environment` [Funzione]
Return the environment (a list of strings) that was in effect at program start.

`ly:stderr-redirect` *file-name mode* [Funzione]
Redirect stderr to *file-name*, opened with *mode*.

`ly:stencil?` *x* [Funzione]
Is *x* a `Stencil` object?

`ly:stencil-add` *args* [Funzione]
Combine stencils. Takes any number of arguments.

`ly:stencil-aligned-to` *stil axis dir* [Funzione]
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* [Funzione]
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`.

`ly:stencil-empty?` *stil axis* [Funzione]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.

`ly:stencil-expr` *stil* [Funzione]
Return the expression of *stil*.

`ly:stencil-extent` *stil axis* [Funzione]
Return a pair of numbers signifying the extent of *stil* in *axis* direction (`0` or `1` for x and y axis, respectively).

`ly:stencil-in-color` *stc r g b a* [Funzione]
Put *stc* in a different color.Accepts either three values for *r*,*g*, *b* and an optional valuefor *a*, or a single CSS-like string.

`ly:stencil-outline` *stil outline* [Funzione]
Return a stencil with the stencil expression (inking) of stencil *stil* but with outline and dimensions from stencil *outline*.

`ly:stencil-rotate` *stil angle x y* [Funzione]
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* [Funzione]
Return a stencil *stil* rotated *angle* degrees around point (*x*, *y*), given in absolute coordinates.

`ly:stencil-scale` *stil x y* [Funzione]
Scale stencil *stil* using the horizontal and vertical scaling factors *x* and *y*. Negative values will flip or mirror *stil* without changing its origin; this may result in collisions unless it is repositioned.

`ly:stencil-stack` *first axis direction second padding mindist* [Funzione]
Construct a stencil by stacking *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be `'()` or `#f`. As opposed to `ly:stencil-combine-at-edge`, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.

If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, *padding* and *mindist* do not apply.

`ly:stencil-translate` *stil offset* [Funzione]
Return a *stil*, but translated by *offset* (a pair of numbers).

`ly:stencil-translate-axis` *stil amount axis* [Funzione]
Return a copy of *stil* but translated by *amount* in *axis* direction.

`ly:stream-event?` *obj* [Funzione]
Is *obj* a `Stream_event` object?

`ly:string-percent-encode` *str* [Funzione]
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* [Funzione]
Replace string *a* by string *b* in string *s*.

`ly:system-font-load` *name* [Funzione]
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` [Funzione]
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.

*layout* is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.

`ly:transform?` *x* [Funzione]
Is *x* a `Transform` object?

`ly:transform->list` *transform* [Funzione]
Convert a transform matrix to a list of six values. Values are *xx*, *yx*, *xy*, *yy*, *x0*, *y0*.

`ly:translate-cpp-warning-scheme` *str* [Funzione]
Translates a string in C++ printf format and modifies it to use it for scheme formatting.

`ly:translator?` *x* [Funzione]
Is *x* a `Translator` object?

`ly:translator-context` *trans* [Funzione]
Return the context of the translator object *trans*.

`ly:translator-description` *creator* [Funzione]
Return an alist of properties of translator definition *creator*.

`ly:translator-group?` *x* [Funzione]
Is *x* a `Translator_group` object?

`ly:translator-name` *creator* [Funzione]
Return the type name of the translator definition *creator*. The name is a symbol.

`ly:transpose-key-alist` *l pit* [Funzione]
Make a new key alist of *l* transposed by pitch *pit*.

`ly:truncate-list!` *lst i* [Funzione]
Take at most the first *i* of list *lst*.

`ly:ttf->pfa` *ttf-file-name idx* [Funzione]
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* [Funzione]
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

`ly:type1->pfa` *type1-file-name* [Funzione]
Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in PFA format, pass through it.

`ly:undead?` *x* [Funzione]
Is *x* a `Undead` object?

`ly:unit` [Funzione]
Return the unit used for lengths as a string.

`ly:unpure-call` *data grob rest* [Funzione]
Convert property *data* (unpure-pure container or procedure) to value in an unpure context defined by *grob* and possibly *rest* arguments.

`ly:unpure-pure-container?` *x* [Funzione]
Is *x* a `Unpure_pure_container` object?

`ly:unpure-pure-container-pure-part` *pc* [Funzione]
Return the pure part of *pc*.

`ly:unpure-pure-container-unpure-part` *pc* [Funzione]
Return the unpure part of *pc*.

`ly:usage` [Funzione]
Print usage message.

`ly:verbose-output?` [Funzione]
Was verbose output requested, i.e. loglevel at least `DEBUG`?
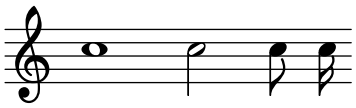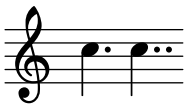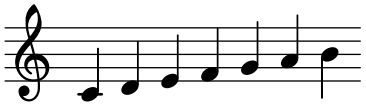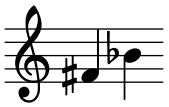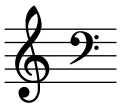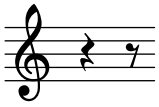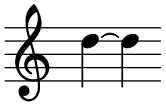
`ly:version` [Funzione]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.
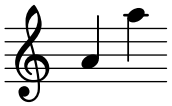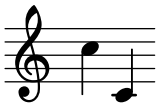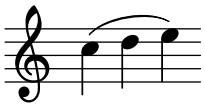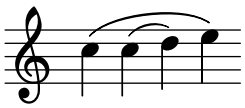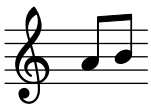
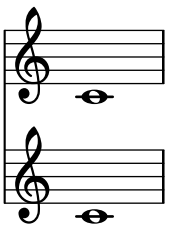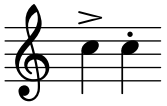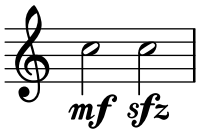
`ly:warning` *str rest* [Funzione]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.

`ly:warning-located` *location str rest* [Funzione]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with `format` and *rest*.
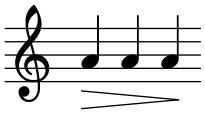
`ly:wide-char->utf-8` *wc* [Funzione]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.

# Appendice B Schema riassuntivo
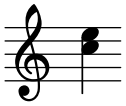
| Sintassi | Descrizione | Esempio |
|---|---|---|
| `1 2 8 16` | durate |  |
| `c4. c4..` | punti di aumentazione |  |
| `c d e f g a b` | scala |  |
| `fis bes` | alterazione |  |
| `\clef treble \clef bass` | chiavi |  |
| `\time 3/4 \time 4/4` | indicazione di tempo |  |
| `r4 r8` | pausa |  |
| `d ~ d` | legatura di valore |  |
| `\key es \major` | armatura di chiave |  |

| | | |
|---|---|---|
| *note*`'` | alzare l'ottava |  |
| *note*`,` | abbassare l'ottava |  |
| `c( d e)` | legatura di portamento |  |
| `c\( c( d) e\)` | legatura di frase |  |
| `a8[ b]` | travatura |  |
| `<< \new Staff ... >>` | più righi |  |
| `c-> c-.` | articolazioni |  |
| `c2\mf c\sfz` | dinamiche |  |
| `a\< a a\!` | crescendo |  |
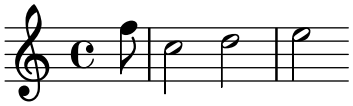
`a\> a a\!` decrescendo
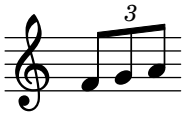


`< >` accordo
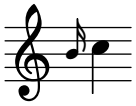


`\partial 8` anacrusi



`\tuplet 3/2 {f g a}` terzine




`\grace` abbellimenti
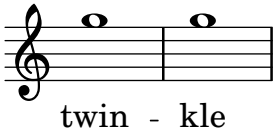


`\lyricmode { twinkle }` inserimento del testo twinkle vocale
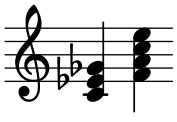
`\new Lyrics` stampa del testo vocale

twinkle

`twin -- kle` trattino nel testo vocale
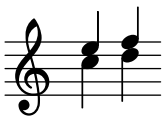



`\chordmode { c:dim f:maj7 }` accordi



`\new ChordNames` mostrare i nomi degli accordi

C° F△

`<<{e f} \\ {c d}>>` polifonia



| | |
|---|---|
| `s4 s8 s16` | pause spaziatrici |

# Appendice C GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
`https://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/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.

# Appendice D Indice dei comandi di LilyPond

Questo indice elenca tutti i comandi e le parole chiave di LilyPond con dei collegamenti alle sezioni del manuale che descrivono il loro uso.


## !

! ..... 6
\! ..... 126

## "

"|" ..... 113

## %

% ..... 481, 485
%{ ... %} ..... 481, 485

## ’

' ..... 2

## (

( ..... 135
\( ..... 138

## )

) ..... 135
\) ..... 138

## ,

, ..... 2

## —

- ..... 124
-! ..... 764
-+ ..... 765
-- ..... 764
-. ..... 764
-> ..... 764
-^ ..... 764
-_ ..... 764

## .

. ..... 47

## /

/ ..... 423
/+ ..... 424

## :

: ..... 168

## <

<...> ..... 170
<> ..... 170, 333
< ..... 170
\< ..... 126

## =

\= ..... 817
= ..... 10

## >

> ..... 170
\> ..... 126

## ?

? ..... 6

## [

[ ..... 97
\[ ..... 442

## ]

] ..... 97
\] ..... 442

## ^

^ ..... 423

## _

_ ..... 275

## |

| ..... 113

## ~

~ ..... 55

# A

\abs-fontsize ..... 248, 707
\absolute ..... 806
\accent ..... 124, 764
\accentus ..... 454, 766
\accepts ..... 607, 609, 610
\acciaccatura ..... 116, 806
\accidentalStyle ..... 28, 806
AccidentalSuggestion ..... 125
add-grace-property ..... 119
\addChordShape ..... 382, 806
\addInstrumentDefinition ..... 806
additionalPitchPrefix ..... 428
\addlyrics ..... 269, 271, 272
\addQuote ..... 213, 806
\aeolian ..... 22
\afterGrace ..... 117, 806
afterGraceFraction ..... 771
\aikenHeads ..... 42
\aikenHeadsMinor ..... 43
\alias ..... 607
alignAboveContext ..... 207, 611
alignBelowContext ..... 207, 290, 611
\allowPageTurn ..... 560, 807
\allowVoltaHook ..... 807
\alterBroken ..... 656, 807
\alternative ..... 153
\ambitusAfter ..... 807
AmbitusLine ..... 37
annotate-spacing ..... 589
\appendToTag ..... 518, 807
\applyContext ..... 597, 807
\applyMusic ..... 807
\applyOutput ..... 807
\appoggiatura ..... 116, 807
\arabicStringNumbers ..... 348
\arpeggio ..... 147
\arpeggioArrowDown ..... 147
\arpeggioArrowUp ..... 147
\arpeggioBracket ..... 147
\arpeggioNormal ..... 147
\arpeggioParenthesis ..... 147
\arpeggioParenthesisDashed ..... 147
\arrow-head ..... 256, 732
\articulate ..... 537
articulation-event ..... 216
\ascendens ..... 456, 462
\assertBeamQuant ..... 807
\assertBeamSlope ..... 807
associatedVoice ..... 269, 302
\auctum ..... 456, 462
aug ..... 421
\augmentum ..... 462
auto-first-page-number ..... 549
\auto-footnote ..... 754
autoBeaming ..... 88, 593
\autoBeamOff ..... 85, 338
\autoBeamOn ..... 85
\autoBreaksOff ..... 554
\autoBreaksOn ..... 554
\autoChange ..... 335, 807
\autoLineBreaksOff ..... 554
\autoLineBreaksOn ..... 554
automaticBars ..... 644
\autoPageBreaksOff ..... 557
\autoPageBreaksOn ..... 557

# B

\backslashed-digit ..... 754
Balloon_engraver ..... 233
\balloonGrobText ..... 233, 807
\balloonLengthOff ..... 233
\balloonLengthOn ..... 233
\balloonText ..... 233, 807
banjo-c-tuning ..... 397
banjo-modal-tuning ..... 397
banjo-open-d-tuning ..... 397
banjo-open-dm-tuning ..... 397
\bar ..... 101, 107, 807
barCheckSynchronize ..... 113
BarNumber ..... 108
\barNumberCheck ..... 114, 807
barNumberVisibility ..... 108
bartype ..... 107
base-shortest-duration ..... 578
baseMoment ..... 88, 93
\bassFigureExtendersOff ..... 435
\bassFigureExtendersOn ..... 435
\bassFigureStaffAlignmentDown ..... 437
\bassFigureStaffAlignmentNeutral ..... 437
\bassFigureStaffAlignmentUp ..... 437
\beam ..... 732
\beamExceptions ..... 88, 808
beatStructure ..... 88, 93
\bendAfter ..... 141, 808
binding-offset ..... 546
\blackTriangleMarkup ..... 428
blank-after-score-page-penalty ..... 548
blank-last-page-penalty ..... 548
blank-page-penalty ..... 548
\bold ..... 248, 707
\book ..... 481, 484
\bookOutputName ..... 483, 808
\bookOutputSuffix ..... 483, 808
\bookpart ..... 482, 484, 558
bookTitleMarkup ..... 494
bottom-margin ..... 542
\box ..... 255, 707
\bracket ..... 133, 732
bracket ..... 340
\bracket ..... 255
\break ..... 554
break-align-symbols ..... 649
break-visibility ..... 640
breakable ..... 86
breakbefore ..... 491
\breathe ..... 139, 808
BreathingSign ..... 140
\breve ..... 46, 59

## C

\cadenzaOff ... 76
\cadenzaOn ... 76
\caesura ... 454
\caps ... 708
\cavum ... 456, 462
\center-align ... 251, 717
\center-column ... 253, 717
\change ... 333
\char ... 754
check-consistency ... 546
choral ... 32
choral-cautionary ... 32
chordChanges ... 386, 426
\chordmode ... 6, 14, 379, 629
chordNameExceptions ... 429
chordNameLowercaseMinor ... 427
ChordNames ... 379
chordNameSeparator ... 428, 431
chordNoteNamer ... 428
chordPrefixSpacer ... 429
\chordRepeats ... 352, 808
chordRootNamer ... 427
\chords ... 425, 629
\circle ... 255, 733
\circulus ... 454, 766
\clef ... 17, 808
clip-regions ... 521
\cm ... 632
\coda ... 124, 766
color ... 229
\column ... 253, 717
\column-lines ... 761
\combine ... 256, 718
common-shortest-duration ... 578
Completion_heads_engraver ... 81
Completion_rest_engraver ... 81
\compound-meter ... 740
\compoundMeter ... 80, 808
\compressMMRests ... 62, 64, 808
\concat ... 718
\consists ... 600, 607
\context ... 594, 603
context-spec-music ... 182
controlpitch ... 10
countPercentRepeats ... 165
\cr ... 126
\cresc ... 128
crescendo-event ... 216
crescendoSpanner ... 132
crescendoText ... 132
\crescHairpin ... 128
\crescTextCresc ... 128
cross ... 39
\crossStaff ... 338, 808
\cueClef ... 217, 808
\cueClefUnset ... 808
\cueDuring ... 217, 808
\cueDuringWithClef ... 217, 808
CueVoice ... 217
currentBarNumber ... 108, 122
\customTabClef ... 740

## D

\dashBang ... 124
\dashDash ... 124
\dashDot ... 124
\dashHat ... 124
\dashLarger ... 124
\dashPlus ... 124
\dashUnderscore ... 124
\deadNote ... 40, 808
\deadNotesOff ... 40
\deadNotesOn ... 40
\decr ... 126
\decresc ... 128
decrescendoSpanner ... 132
decrescendoText ... 132
default-staff-staff-spacing ... 561
default ... 28, 30
\default ... 114, 500
defaultBarType ... 107
\defaultchild ... 611
\defaultTimeSignature ... 68
\defineBarLine ... 105, 808
\deminutum ... 456, 462
\denies ... 607, 609, 610
\descendens ... 456, 462
dim ... 421
\dim ... 128
\dimHairpin ... 128
\dimTextDecr ... 128
\dimTextDecresc ... 128
\dimTextDim ... 128
\dir-column ... 718
\discant ... 749
\displayLilyMusic ... 537, 809
\displayMusic ... 809
\displayScheme ... 809
\divisioMaior ... 454
\divisioMaxima ... 454
\divisioMinima ... 454
dodecaphonic ... 34
dodecaphonic-first ... 34
dodecaphonic-no-repeat ... 34
\dorian ... 22
\dotsDown ... 47
\dotsNeutral ... 47
\dotsUp ... 47
\doubleflat ... 741
\doublesharp ... 741
\downbow ... 124, 344, 765
\downmordent ... 124, 764
\downprall ... 124, 764
\draw-circle ... 256, 733
\draw-dashed-line ... 733
\draw-dotted-line ... 734
\draw-hline ... 734
\draw-line ... 256, 734
\draw-squiggle-line ... 735
\dropNote ... 809
\drummode ... 194, 398, 629
drumPitchNames ... 403
drumPitchTable ... 403
\drums ... 398, 629
DrumStaff ... 194
drumStyleTable ... 402

\dwn ..... 474
\dynamic ..... 133, 708
dynamic-event ..... 216
\dynamicDown ..... 128
DynamicLineSpanner ..... 128, 131
\dynamicNeutral ..... 128
\dynamicUp ..... 128

## E

\easyHeadsOff ..... 40
\easyHeadsOn ..... 40
\ellipse ..... 735
\endcr ..... 126
\enddecr ..... 126
\endSpanners ..... 638, 809
\episemFinis ..... 454
\episemInitium ..... 454
\epsfile ..... 256, 736
\espressivo ..... 124, 128, 764
\etc ..... 661
\eventChords ..... 809
explicitClefVisibility ..... 642
explicitKeySignatureVisibility ..... 642
extra-offset ..... 561
\eyeglasses ..... 755
Ez_numbers_engraver ..... 41

## F

\f ..... 126
\featherDurations ..... 100, 809
\fermata ..... 124, 741, 765
\ff ..... 126
\fff ..... 126
\ffff ..... 126
\fffff ..... 126
figuredBassAlterationDirection ..... 436
figuredBassPlusDirection ..... 436
\figuremode ..... 433, 629
\figures ..... 433, 629
\fill-line ..... 253, 719
\fill-with-pattern ..... 719
\filled-box ..... 256, 736
\finalis ..... 454
\finger ..... 226, 708, 809
fingeringOrientations ..... 227
first-page-number ..... 549
\first-visible ..... 755
\fixed ..... 2, 809
\flageolet ..... 124, 406, 765
\flat ..... 741
\flexa ..... 462
followVoice ..... 337
font-interface ..... 226, 261
font-size ..... 222, 226
\fontCaps ..... 708
fontSize ..... 222
\fontsize ..... 248, 708
\footnote ..... 499, 755, 809
Forbid_line_break_engraver ..... 53
forget ..... 35
four-string-banjo ..... 397
\fp ..... 126
\fraction ..... 755
\freeBass ..... 750
\frenchChords ..... 427
\fret-diagram ..... 368, 746
fret-diagram-interface ..... 375
\fret-diagram-terse ..... 371, 746
\fret-diagram-verbose ..... 373, 747
FretBoards ..... 378
\fromproperty ..... 755
\funkHeads ..... 42
\funkHeadsMinor ..... 43

## G

\general-align ..... 252, 720
\germanChords ..... 427
\glissando ..... 142
\glissandoMap ..... 143
\grace ..... 116, 809
GregorianTranscriptionStaff ..... 194
Grid_line_span_engraver ..... 234
Grid_point_engraver ..... 234
gridInterval ..... 234
grob-interface ..... 770
\grobdescriptions ..... 809
grow-direction ..... 100

## H

\halfopen ..... 124, 765
\halign ..... 251, 721
\harmonic ..... 40, 345, 355
\harmonicByFret ..... 355, 809
\harmonicByRatio ..... 355, 809
\harmonicNote ..... 810
\harmonicsOff ..... 345
\harmonicsOn ..... 345, 810
\harp-pedal ..... 748
\hbracket ..... 255, 736
\hcenter-in ..... 722
\header ..... 484
\hide ..... 639, 810
\hideKeySignature ..... 409
\hideNotes ..... 229
\hideSplitTiedTabNotes ..... 354
\hideStaffSwitch ..... 337
horizontal-shift ..... 547
Horizontal_bracket_engraver ..... 236
HorizontalBracketText ..... 237
\hspace ..... 722
\huge ..... 222, 251, 709

## I

\ictus ..... 454, 766
\iij ..... 457
\IIJ ..... 457
\ij ..... 457
\IJ ..... 457
\improvisationOff ..... 45, 83
\improvisationOn ..... 45, 83
\in ..... 632
\incipit ..... 466, 810
\inclinatum ..... 456, 462
\include ..... 485, 511
indent ..... 211, 547, 582
\inherit-acceptability ..... 609, 810
inner-margin ..... 546
\inStaffSegno ..... 156, 810
\instrumentSwitch ..... 810
\inversion ..... 14, 810
\invertChords ..... 810
\ionian ..... 22
\italianChords ..... 427
\italic ..... 248, 709

## J

\justified-lines ..... 260, 761
\justify ..... 254, 724
\justify-field ..... 723
\justify-line ..... 723
\justify-string ..... 724

## K

\keepWithTag ..... 515, 810
\key ..... 22, 43, 810
\kievanOff ..... 463
\kievanOn ..... 463
KievanStaff ..... 462
KievanVoice ..... 462
\killCues ..... 221, 810

## L

\label ..... 509, 810
\laissezVibrer ..... 56
\language ..... 810
\languageRestore ..... 810
\languageSaveAndChange ..... 810
\large ..... 222, 251, 709
\larger ..... 248, 251, 709
last-bottom-spacing ..... 544
\layout ..... 484, 550, 592, 603
layout-set-staff-size ..... 552
\left-align ..... 251, 725
\left-brace ..... 756
\left-column ..... 725
left-margin ..... 545
\lheel ..... 124, 765
\line ..... 725
line-width ..... 545, 582
\linea ..... 456, 462
\lineprall ..... 124, 764
\locrian ..... 22
\longa ..... 46, 59
\longfermata ..... 124, 765
\lookup ..... 756
\lower ..... 252, 726
\ltoe ..... 124, 765
ly:minimal-breaking ..... 559
ly:one-line-auto-height-breaking ..... 559
ly:one-line-breaking ..... 559
ly:one-page-breaking ..... 559
ly:optimal-breaking ..... 559
ly:page-turn-breaking ..... 560
\lydian ..... 22
\lyricmode ..... 268, 269, 630
\lyrics ..... 630
\lyricsto ..... 269, 271, 272

## M

m ..... 421
magnification->font-size ..... 222, 552
\magnify ..... 248, 709
\magnifyMusic ..... 222, 811
\magnifyStaff ..... 553, 811
magstep ..... 222, 552, 632
maj ..... 421
\major ..... 22
majorSevenSymbol ..... 427, 430
make-dynamic-script ..... 133
make-pango-font-tree ..... 264
\makeClusters ..... 175, 811
\makeDefaultStringTuning ..... 811
\map-markup-commands ..... 761
\marcato ..... 124, 764
\mark ..... 114, 242, 811
Mark_engraver ..... 244
\markalphabet ..... 756
\markLengthOff ..... 73, 243
\markLengthOn ..... 73, 243
\markletter ..... 757
\markup ..... 242, 245, 247, 630
markup-markup-spacing ..... 544
markup-system-spacing ..... 544
\markuplist ..... 245, 260
\markupMap ..... 811
max-systems-per-page ..... 547
\maxima ..... 46, 59
Measure_grouping_engraver ..... 94
measureLength ..... 88, 122
measurePosition ..... 75, 122
\medium ..... 710
\melisma ..... 276
\melismaEnd ..... 276
MensuralStaff ..... 194, 444
MensuralVoice ..... 444
\mergeDifferentlyDottedOff ..... 179
\mergeDifferentlyDottedOn ..... 179
\mergeDifferentlyHeadedOff ..... 179
\mergeDifferentlyHeadedOn ..... 179
\mf ..... 126
\midi ..... 484, 592
midiBalance ..... 535
midiChannelMapping ..... 533
midiChorusLevel ..... 535
midiDrumPitches ..... 403
midiExpression ..... 535

midiPanPosition ... 535
midiReverbLevel ... 535
min-systems-per-page ... 547
minimum-Y-extent ... 561
minimumFret ... 351, 390
minimumPageTurnLength ... 560
minimumRepeatLengthForPageTurn ... 560
\minor ... 22
minorChordModifier ... 429
mixed ... 340
\mixolydian ... 22
\mm ... 632
\modalInversion ... 16, 811
\modalTranspose ... 15, 811
mode ... 771
modern ... 30
modern-cautionary ... 31
modern-voice ... 31
modern-voice-cautionary ... 31
\mordent ... 124, 764
\mp ... 126
MultiMeasureRestScript ... 64
MultiMeasureRestText ... 64
\musicglyph ... 115, 741
\musicMap ... 811
musicQuotes ... 771

## N

\name ... 607
\natural ... 742
neo-modern ... 33
neo-modern-cautionary ... 33
neo-modern-voice ... 33
neo-modern-voice-cautionary ... 33
\new ... 594
\newSpacingSection ... 579
no-reset ... 34
\noBeam ... 97
\noBreak ... 554
nonstaff-nonstaff-spacing ... 561
nonstaff-relatedstaff-spacing ... 561
nonstaff-unrelatedstaff-spacing ... 561
\noPageBreak ... 557, 811
\noPageTurn ... 560, 811
\normal-size-sub ... 710
\normal-size-super ... 250, 710
\normal-text ... 711
\normalsize ... 222, 251, 711
\note ... 742
\note-by-number ... 742
note-event ... 216
Note_heads_engraver ... 81
\notemode ... 630
\null ... 252, 757
NullVoice ... 297
\number ... 711
\numericTimeSignature ... 68

## O

\octaveCheck ... 10, 811
\offset ... 622, 811
\omit ... 638, 811
\on-the-fly ... 497, 757
\once ... 617, 619, 623, 657, 812
\oneVoice ... 175
\open ... 124, 344, 765
\oriscus ... 456, 462
\ottava ... 24, 812
outer-margin ... 546
output-count ... 771
output-def ... 770
output-suffix ... 771
outside-staff-horizontal-padding ... 576
outside-staff-padding ... 576
outside-staff-priority ... 576
\oval ... 736
\overlay ... 726
\override ... 617, 621, 757
\override-lines ... 761
\overrideProperty ... 622, 812
\overrideTimeSignatureSettings ... 68, 812
\overtie ... 711

## P

\p ... 126
\pad-around ... 255, 726
\pad-markup ... 255, 727
\pad-to-box ... 255, 727
\pad-x ... 255, 727
page-breaking ... 548
page-breaking-system-system-spacing ... 548
page-count ... 548
\page-link ... 757
page-number-type ... 549
\page-ref ... 509, 758
page-spacing-weight ... 549
\pageBreak ... 557, 812
\pageTurn ... 560, 812
\palmMute ... 812
\palmMuteOn ... 812
\paper ... 484, 540
paper-height ... 542
paper-width ... 545
\parallelMusic ... 191, 812
\parenthesize ... 231, 737, 812
\partCombine ... 185, 297, 813
\partCombineApart ... 187
\partCombineAutomatic ... 187
\partCombineChords ... 187
\partCombineDown ... 813
\partCombineForce ... 813
partCombineListener ... 771
\partCombineSoloI ... 187
\partCombineSoloII ... 187
\partCombineUnisono ... 187
\partCombineUp ... 813
\partial ... 75, 153, 155, 813
\path ... 737
\pattern ... 758
pedalSustainStyle ... 340
percent ... 164

\pes ... 462
\phrasingSlurDashed ... 138
\phrasingSlurDashPattern ... 139, 813
\phrasingSlurDotted ... 138
\phrasingSlurDown ... 138
\phrasingSlurHalfDashed ... 138
\phrasingSlurHalfSolid ... 138
\phrasingSlurNeutral ... 138
\phrasingSlurSolid ... 138
\phrasingSlurUp ... 138
\phrygian ... 22
piano ... 32
piano-cautionary ... 32
PianoStaff ... 332, 335
Pitch_squash_engraver ... 83
\pitchedTrill ... 151, 813
pitchnames ... 771
\pointAndClickOff ... 813
\pointAndClickOn ... 813
\pointAndClickTypes ... 813
\portato ... 124, 764
\postscript ... 256, 738
\pp ... 126
\ppp ... 126
\pppp ... 126
\ppppp ... 126
\prall ... 124, 764
\pralldown ... 124, 764
\prallmordent ... 124, 764
\prallprall ... 124, 764
\prallup ... 124, 764
predefinedDiagramTable ... 387
\predefinedFretboardsOff ... 389
\predefinedFretboardsOn ... 389
print-all-headers ... 549
print-first-page-number ... 549
print-page-number ... 549
\property-recursive ... 758
\propertyOverride ... 813
\propertyRevert ... 813
\propertySet ... 813
\propertyTweak ... 814
\propertyUnset ... 814
\pt ... 632
\pushToTag ... 518, 814
\put-adjacent ... 727

## Q

\quilisma ... 456, 462
quotedCueEventTypes ... 216
quotedEventTypes ... 216
\quoteDuring ... 213, 217, 814

## R

r ... 59
R ... 62
ragged-bottom ... 542
ragged-last ... 546, 582
ragged-last-bottom ... 542
ragged-right ... 546, 582
\raise ... 252, 728
\raiseNote ... 814
\reduceChords ... 83, 814
\relative ... 2, 6, 14, 336, 814
\remove ... 600
remove-grace-property ... 119
\RemoveAllEmptyStaves ... 208, 818
\RemoveEmptyStaves ... 208, 818
\removeWithTag ... 515, 814
\repeat ... 153
\repeat percent ... 164
\repeat tremolo ... 167
repeatCommands ... 160
repeatCountVisibility ... 166
\repeatTie ... 56, 155, 293
\replace ... 712
\resetRelativeOctave ... 5, 814
\responsum ... 457
\rest ... 59, 743
\rest-by-number ... 742
rest-event ... 216
restNumberThreshold ... 64
restrainOpenStrings ... 351
\retrograde ... 15, 814
\reverseturn ... 124, 764
\revert ... 618
\revertTimeSignatureSettings ... 69, 814
\rfz ... 126
rgb-color ... 230
\rheel ... 124, 765
RhythmicStaff ... 194
\right-align ... 251, 728
\right-brace ... 758
\right-column ... 728
right-margin ... 545
\rightHandFinger ... 391, 814
\roman ... 712
\romanStringNumbers ... 344, 348
\rotate ... 729
\rounded-box ... 255, 739
\rtoe ... 124, 765

## S

s ... 61
\sacredHarpHeads ... 42
\sacredHarpHeadsMinor ... 43
\sans ... 712
\scale ... 739
\scaleDurations ... 54, 78, 814
\score ... 480, 484, 743
\score-lines ... 761
score-markup-spacing ... 544
score-system-spacing ... 544
scoreTitleMarkup ... 494
\segno ... 124, 766
self-alignment-X ... 561

\semicirculus ..... 454, 766
\semiflat ..... 744
\semiGermanChords ..... 427
\semisharp ..... 745
\sesquiflat ..... 745
\sesquisharp ..... 745
\set ..... 88, 615, 621
set-global-fonts ..... 265
set-global-staff-size ..... 552
set-octavation ..... 24
\settingsFrom ..... 814
\sf ..... 126
\sff ..... 126
\sfz ..... 126
\shape ..... 653, 815
\sharp ..... 745
\shiftDurations ..... 815
\shiftOff ..... 179
\shiftOn ..... 179
\shiftOnn ..... 179
\shiftOnnn ..... 179
short-indent ..... 211, 547
\shortfermata ..... 124, 765
show-available-fonts ..... 264
showFirstLength ..... 522, 771
\showKeySignature ..... 409
showLastLength ..... 522, 771
\showStaffSwitch ..... 337
\signumcongruentiae ..... 124, 766
\simple ..... 713
\single ..... 503, 624, 815
\skip ..... 61, 292, 815
skipTypesetting ..... 522
slashChordSeparator ..... 428
\slashed-digit ..... 758
\slashedGrace ..... 116, 815
\slashSeparator ..... 549
slur-event ..... 216
\slurDashed ..... 135
\slurDashPattern ..... 136, 815
\slurDotted ..... 135
\slurDown ..... 135
\slurHalfDashed ..... 136
\slurHalfSolid ..... 136
\slurNeutral ..... 135
\slurSolid ..... 135
\slurUp ..... 136
\small ..... 222, 251, 713
\smallCaps ..... 713
\smaller ..... 248, 251, 713
\snappizzicato ..... 124, 765
\sostenutoOff ..... 340
\sostenutoOn ..... 340
\sourcefileline ..... 485
\sourcefilename ..... 485
\southernHarmonyHeads ..... 42
\southernHarmonyHeadsMinor ..... 43
\sp ..... 126
spacing ..... 578
\spacingTweaks ..... 815
Span_stem_engraver ..... 338
\spp ..... 126
\staccatissimo ..... 124, 764
\staccato ..... 124, 764
staff-affinity ..... 561
\staff-space ..... 632
staff-staff-spacing ..... 561
Staff.midiInstrument ..... 536
Staff_collecting_engraver ..... 244
Staff_symbol_engraver ..... 208
staffgroup-staff-spacing ..... 561
start-repeat ..... 160
startAcciaccaturaMusic ..... 119
startAppoggiaturaMusic ..... 119
startGraceMusic ..... 119
\startGroup ..... 236
\startStaff ..... 201, 204
\startTrillSpan ..... 150
\stdBass ..... 750
\stdBassIV ..... 751
\stdBassV ..... 752
\stdBassVI ..... 753
stem-spacing-correction ..... 578
\stemDown ..... 232
stemLeftBeamCount ..... 98
\stemNeutral ..... 232
stemRightBeamCount ..... 98
\stemUp ..... 232
\stencil ..... 759
stopAcciaccaturaMusic ..... 119
stopAppoggiaturaMusic ..... 119
stopGraceMusic ..... 119
\stopGroup ..... 236
\stopped ..... 124, 765
\stopStaff ..... 201, 204, 208
\stopTrillSpan ..... 150
\storePredefinedDiagram ..... 382, 387, 815
strictBeatBeaming ..... 94
stringNumberOrientations ..... 227
\stringTuning ..... 365, 815
stringTunings ..... 364, 378
strokeFingerOrientations ..... 227, 392
\stropha ..... 456, 462
\strut ..... 759
\styledNoteHeads ..... 815
\sub ..... 250, 714
subdivideBeams ..... 93
suggestAccidentals ..... 125, 450
\super ..... 250, 714
sus ..... 423
\sustainOff ..... 340
\sustainOn ..... 340
system-count ..... 547
system-separator-markup ..... 549
system-system-spacing ..... 544
systems-per-page ..... 547

## T

\tabChordRepeats ..... 352, 815
\tabChordRepetition ..... 815
\tabFullNotation ..... 351
\table ..... 761
\table-of-contents ..... 511, 762
TabStaff ..... 194, 350
TabVoice ..... 350
\tag ..... 515, 815
\tagGroup ..... 517, 815
\taor ..... 409

teaching ..... 34
\teeny ..... 222, 251, 714
\tempo ..... 71
\temporary ..... 623, 657, 816
\tenuto ..... 124, 764
text ..... 340, 714
\textLengthOff ..... 64, 66, 240
\textLengthOn ..... 64, 66, 131, 240
\textSpannerDown ..... 241
\textSpannerNeutral ..... 241
\textSpannerUp ..... 241
\thumb ..... 124, 227
\tie ..... 715
TieColumn ..... 58
\tied-lyric ..... 745
\tieDashed ..... 56
\tieDashPattern ..... 56, 816
\tieDotted ..... 56
\tieDown ..... 56
\tieHalfDashed ..... 56
\tieHalfSolid ..... 56
\tieNeutral ..... 56
\tieSolid ..... 56
\tieUp ..... 56
tieWaitForNote ..... 58
\time ..... 67, 88, 816
\times ..... 816
timeSignatureFraction ..... 78
\tiny ..... 222, 251, 715
\tocItem ..... 511, 816
\tocItemWithDotsMarkup ..... 510
top-margin ..... 542
top-markup-spacing ..... 544
top-system-spacing ..... 544
toplevel-bookparts ..... 771
toplevel-scores ..... 771
\translate ..... 252, 729
\translate-scaled ..... 252, 729
\transparent ..... 759
\transpose ..... 6, 11, 14, 816
\transposedCueDuring ..... 220, 816
\transposition ..... 27, 213, 816
\treCorde ..... 340
tremolo ..... 167
\triangle ..... 256, 739
\trill ..... 124, 150, 764
\tuplet ..... 49, 78, 816
\tupletDown ..... 49
\tupletNeutral ..... 49
TupletNumber ..... 50
\tupletSpan ..... 50, 816
tupletSpannerDuration ..... 50
\tupletUp ..... 49
\turn ..... 124, 764
\tweak ..... 619, 622, 816
two-sided ..... 546
\type ..... 607
\typewriter ..... 715

# U

\unaCorda ..... 340
\underline ..... 248, 715
\undertie ..... 716
\undo ..... 624, 817
unfold ..... 162
\unfoldRepeats ..... 532, 817
\unHideNotes ..... 229
\unset ..... 616
\upbow ..... 124, 344, 765
\upmordent ..... 124, 764
\upprall ..... 124, 764
\upright ..... 716

# V

\varcoda ..... 124, 766
variabile globale ..... 771
VaticanaStaff ..... 194, 452
VaticanaVoice ..... 452
\vcenter ..... 729
\verbatim-file ..... 759
\version ..... 485
\versus ..... 457
VerticalAxisGroup ..... 561
\verylongfermata ..... 124, 765
\virga ..... 456, 462
\virgula ..... 454
Voice ..... 175
voice ..... 28, 30
\voiceFour ..... 175
\voiceFourStyle ..... 179
\voiceNeutralStyle ..... 179
\voiceOne ..... 175
\voiceOneStyle ..... 179
\voices ..... 178, 817
\voiceThree ..... 175
\voiceThreeStyle ..... 179
\voiceTwo ..... 175
\voiceTwoStyle ..... 179
\void ..... 537, 817
Volta_engraver ..... 158
\vspace ..... 730

# W

\walkerHeads ..... 42
\walkerHeadsMinor ..... 43
whichBar ..... 107
\whiteout ..... 759
\whiteTriangleMarkup ..... 428
\with ..... 600, 605
\with-color ..... 229, 760
\with-dimensions ..... 760
\with-dimensions-from ..... 760
\with-link ..... 761
\with-outline ..... 761
\with-url ..... 740
\withMusicProperty ..... 817
\woodwind-diagram ..... 748
\wordwrap ..... 254, 731
\wordwrap-field ..... 730
\wordwrap-internal ..... 762
\wordwrap-lines ..... 260, 762

\wordwrap-string ..... 731
\wordwrap-string-internal ..... 763

# X

x11-color ..... 229, 231
X-offset ..... 561
\xNote ..... 39, 817
\xNotesOff ..... 39
\xNotesOn ..... 39

# Appendice E Indice di LilyPond

Oltre a tutti i comandi e le parole chiave di LilyPond, questo indice elenca i termini musicali e le espressioni che si riferiscono a ognuno di essi, corredati di collegamenti alle relative sezioni del manuale.

Gli elementi inclinati dell'indice puntano a posizioni (soprattutto le sezioni 'Vedi anche') che contengono collegamenti esterni ad altri file della documentazione di LilyPond, come la Guida al funzionamento interno o il Glossario.


## !

! ..... 6
\! ..... 126

## "

"|" ..... 113

## %

% ..... 481, 485
%{ ... %} ..... 481, 485

## ’

' ..... 2

## (

( ..... 135
\( ..... 138

## )

) ..... 135
\) ..... 138

## ,

, ..... 2

## —

- ..... 124
-! ..... 764
-+ ..... 765
-- ..... 764
-. ..... 764
-> ..... 764
-^ ..... 764
-_ ..... 764

## .

. ..... 47

## /

/ ..... 423
/+ ..... 424

## :

: ..... 168

## <

<...> ..... 170
<> ..... 170, 333
< ..... 170
\< ..... 126

## =

\= ..... 817
= ..... 10

## >

> ..... 170
\> ..... 126

## ?

? ..... 6

## [

[ ..... 97
\[ ..... 442

## ]

] ..... 97
\] ..... 442

## ^

^ ..... 423

## _

_ ..... 275

## |

| ..... 113

## ~

~ ..... 55

# 1

15ma ... 24
15mb ... 24

# 8

8va ... 24
8vb ... 24

# A

a capo ... 554
a capo, gestione con una voce apposita ... 556
a capo, glissando ... 145
a capo, testo ... 254
*a due* ... 190
abbellimento ... 116, 124, 409
abbellimento al termine di una nota ... 117
abbellimento, e spaziatura rigorosa ... 119
abbellimento, e testo cantato ... 301
abbellimento, modifica delle impostazioni di formattazione ... 118, 119
abbellimento, modifica manuale ... 118, 119
abbellimento, sincronizzazione ... 120
`\abs-fontsize` ... 248, 707
`\absolute` ... 806
*accent* ... 126
`\accent` ... 124, 764
accento ... 124
`\accentus` ... 454, 766
`\accepts` ... 607, 609, 610
*acciaccatura* ... 120
acciaccatura ... 116
`\acciaccatura` ... 116, 806
acciaccatura su più note ... 120
*Accidental* ... 8, 35
*accidental* ... 449, 454, 464
*accidental-interface* ... 8
*accidental-suggestion-interface* ... 35
*Accidental_engraver* ... 8, 35, 450
*AccidentalCautionary* ... 8
*AccidentalPlacement* ... 35
`\accidentalStyle` ... 28, 806
*AccidentalSuggestion* ... 35, 450
`AccidentalSuggestion` ... 125
accollatura ... 195
accompagnamento, ritmo, mostrare ... 83, 84
accordatura non occidentale ... 472
accordatura per banjo ... 397
accordatura per liuto ... 398
accordatura personalizzata ... 365
accordatura predefinita, per strumenti con tasti ... 364
accordi ripetuti, soppressione ... 386
accordi, diagrammi automatici ... 388
accordi, inversioni ... 423
accordi, ripetizione ... 352
accordi, ripetizioni con volte, sotto ... 430
accordo ... 170, 419
accordo alterato ... 422
accordo che attraversa i righi ... 338
accordo di settima ... 420
accordo di settima maggiore, formattazione ... 430
accordo esteso ... 422
accordo jazz ... 427
accordo trasversale ai righi ... 338
accordo vuoto ... 119, 170, 333
accordo, alterazioni in ... 35
accordo, altezza relativa ... 171
accordo, diagramma per strumenti a tasto ... 368
accordo, diteggiatura ... 227
accordo, e legatura di valore ... 55
accordo, e ottava relativa ... 5
accordo, fondamentale ... 420
accordo, glissando, in intavolatura ... 362
accordo, modalità per ... 419
accordo, modificare una nota dell' ... 620
accordo, nome ... 419, 424
accordo, nome alternativo ... 427
accordo, nome, personalizzazione ... 427
accordo, nomi, eccezioni ... 429
accordo, power chord ... 395
accordo, qualità ... 420
accordo, ripetizione ... 171
accordo, soppressione delle ripetizioni ... 426
accordo, sopprimere ripetizioni ... 386
accordo, suddivisione sui righi con `\autoChange` ... 337
`add-grace-property` ... 119
`\addChordShape` ... 382, 806
adding white background, to text ... 759
`\addInstrumentDefinition` ... 806
`additionalPitchPrefix` ... 428
`\addlyrics` ... 269, 271, 272
`\addQuote` ... 213, 806
`\aeolian` ... 22
aerofono, strumento ... 406
`\afterGrace` ... 117, 806
`afterGraceFraction` ... 771
aggiungere diagramma dei tasti personalizzato ... 381
*Aggiungere e togliere gli incisori* ... 82
aggiunta, negli accordi ... 422
agogo ... 766
Aiken, testa di nota ... 42
`\aikenHeads` ... 42
`\aikenHeadsMinor` ... 43
aiuto, nuvoletta ... 233
*al niente* ... 132
al niente, forcella ... 130
`\alias` ... 607
`alignAboveContext` ... 207, 611
`alignBelowContext` ... 207, 290, 611
alist ... 769
*All layout objects* ... 224, 615, 622, 646, 769
allineamento del basso continuo ... 438
allineamento orizzontale del testo ... 251
allineamento orizzontale, testo vocale ... 287
allineamento su un oggetto ... 649
allineamento sulla cadenza ... 121
allineamento verticale del testo ... 252
allineamento verticale, dinamiche ... 131
allineamento verticale, segno testuale ... 131
allineamento, numeri di battuta ... 112
allineamento, testo, comandi ... 254
allineare il markup ... 251
allineare il testo ... 251
*Allineare il testo alla melodia* ... 271, 279
`\allowPageTurn` ... 560, 807
`\allowVoltaHook` ... 807
alterazione ... 6, 28

alterazione automatica ... 28
alterazione di cortesia ... 6
alterazione di sicurezza ... 6
alterazione di un quarto di tono ... 8
alterazione gregoriana ... 453
alterazione mensurale ... 449
alterazione moderna ... 31
alterazione nella notazione di Kiev ... 464
alterazione su più voci ... 31
alterazione tra parentesi ... 6
alterazione, basso continuo, posizione ... 436
alterazione, cadenze ... 76
alterazione, di cortesia ... 6
alterazione, di sicurezza ... 6
alterazione, e legatura di valore ... 7
alterazione, e note simultanee ... 35
alterazione, musica ficta ... 450
alterazione, musica in tempo libero ... 76
alterazione, negli accordi ... 35
alterazione, quarto di tono ... 8
alterazione, stile *modern-cautionary* ... 31
alterazione, stile moderno ... 30
alterazione, tra parentesi ... 6
alterazioni, nascondere, su note legate all'inizio del sistema ... 7
`\alterBroken` ... 656, 807
`\alternative` ... 153
altezza ... 1
altezza isolata ... 47
altezza naturale ... 6
altezza predefinita ... 47
altezza relativa, accordi ... 171
altezza, trasposizione di ... 11
*Altezze e armature di chiave* ... 6, 8, 22, 24, 476
altezze, trasposizione "intelligente" ... 13
alto, chiave di ... 17, 702
*Altre fonti di informazione* ... 179, 472, 473, 513, 524, 531, 532, 537, 613, 646
*Altri usi delle modifiche con \tweak* ... 332, 333
Amazing Grace, esempio per cornamusa ... 409
ambito delle altezze ... 36
*ambitus* ... 38, 268
*Ambitus* ... 38
ambitus ... 36
ambitus, distanza rispetto alla linea ... 37
ambitus, per voce ... 36
ambitus, su più voci ... 37
*ambitus-interface* ... 38
*Ambitus_engraver* ... 38
*AmbitusAccidental* ... 38
`\ambitusAfter` ... 807
*AmbitusLine* ... 38
`AmbitusLine` ... 37
*AmbitusNoteHead* ... 38
anacrusi ... 75
anacrusi, in una ripetizione ... 155
*anacrusis* ... 76
analisi musicologica ... 236
analisi, parentesi, con etichetta ... 237
analizzatore sintattico ... 770
*Ancient notation* ... 444, 449, 455, 456
angolazione, con travature, modifica ... 86
annidamento dei righi ... 198
annidamento, ripetizione ... 160
*Annidare le espressioni musicali* ... 205, 208, 611
`annotate-spacing` ... 589
annotazione ... 247
annotazione su pausa multipla ... 64
antica, chiave ... 17
aperto ... 124
apice ... 250
*Appare un rigo in più* ... 155, 164, 612
`\appendToTag` ... 518, 807
`\applyContext` ... 597, 807
`\applyMusic` ... 807
`\applyOutput` ... 807
*appoggiatura* ... 120
appoggiatura ... 116
`\appoggiatura` ... 116, 807
araba, esempio di musica ... 477
araba, improvvisazione ... 476
araba, modello di musica ... 477
araba, musica ... 473
`\arabicStringNumbers` ... 348
arabo, nomi delle note in ... 474
arcata in giù ... 124, 344
arcata in su ... 124, 344
arcata, indicazione di ... 344
archi da orchestra ... 344
archi, scrivere per ... 344
armatura di chiave ... 6, 22
armatura di chiave gregoriana ... 453
armatura di chiave mensurale ... 449
armatura di chiave, impedire i segni di bequadro ... 23
armatura di chiave, non tradizionale ... 24
armatura di chiave, visibilità dopo un cambio esplicito ... 642
armature di chiave arabe ... 474
armonici artificiali ... 345
armonici naturali ... 345
armonici su strumento con tasti ... 393
armonici, in intavolatura ... 357
armonici, intavolatura ... 355
armonico ... 124
armonico, testa di nota ... 39
arpa ... 342
arpa, diagramma del pedale ... 342
arpa, pedale ... 342
*arpeggio* ... 149
*Arpeggio* ... 149, 622
arpeggio ... 147
`\arpeggio` ... 147
arpeggio attraverso il rigo, stile della parentesi ... 150
arpeggio e legature di valore ... 58
arpeggio spezzato ... 147
arpeggio, attraverso i righi ... 148
arpeggio, attraverso le voci ... 149
arpeggio, parentesi, trasversale ai righi ... 339
arpeggio, simbolo speciale ... 147
`\arpeggioArrowDown` ... 147
`\arpeggioArrowUp` ... 147
`\arpeggioBracket` ... 147
`\arpeggioNormal` ... 147
`\arpeggioParenthesis` ... 147
`\arpeggioParenthesisDashed` ... 147
`\arrow-head` ... 256, 732
articolazione ... 124
articolazione "espressivo" ... 128
articolazione gregoriana ... 454
articolazione, valori predefiniti, modifica ... 124

*Articolazioni e dinamiche* ... 132
`\articulate` ... 537
articulate, script ... 536, 537
articulate.ly ... 536, 537
`articulation-event` ... 216
artificiali, armonici ... 345
`\ascendens` ... 456, 462
`\assertBeamQuant` ... 807
`\assertBeamSlope` ... 807
associatedVoice ... 271
`associatedVoice` ... 269, 302
assoluta, ottava ... 1
assoluto ... 1
`\auctum` ... 456, 462
audio ... 525
`aug` ... 421
`\augmentum` ... 462
`auto-first-page-number` ... 549
`\auto-footnote` ... 754
*Auto_beam_engraver* ... 88, 96
`autoBeaming` ... 88, 593
`\autoBeamOff` ... 85, 338
`\autoBeamOn` ... 85
`\autoBreaksOff` ... 554
`\autoBreaksOn` ... 554
`\autoChange` ... 335, 807
`autochange`, e musica relativa ... 336
*AutoChangeMusic* ... 337
`\autoLineBreaksOff` ... 554
`\autoLineBreaksOn` ... 554
`automaticBars` ... 644
`\autoPageBreaksOff` ... 557
`\autoPageBreaksOn` ... 557
*Axis_group_engraver* ... 567

# B

Bézier, curva, punti di controllo ... 652
*Backend* ... 612, 615, 619
backslashed digit ... 754
`\backslashed-digit` ... 754
*balloon-interface* ... 234
*Balloon_engraver* ... 234
`Balloon_engraver` ... 233
`\balloonGrobText` ... 233, 807
`\balloonLengthOff` ... 233
`\balloonLengthOn` ... 233
`\balloonText` ... 233, 807
*BalloonTextItem* ... 234
banjo, accordatura ... 397
banjo, diagramma dei tasti ... 370
banjo, intavolatura ... 347, 396
banjo, tablatura ... 347
`banjo-c-tuning` ... 397
`banjo-modal-tuning` ... 397
`banjo-open-d-tuning` ... 397
`banjo-open-dm-tuning` ... 397
`\bar` ... 101, 107, 807
*Bar_engraver* ... 427
*Bar_number_engraver* ... 113
`barCheckSynchronize` ... 113
baritono, chiave di ... 17, 702
*BarLine* ... 108
*BarNumber* ... 113
`BarNumber` ... 108
`\barNumberCheck` ... 114, 807
`barNumberVisibility` ... 108
barré, indicazione ... 369
barra, gambo con ... 118
barrata, testa di nota ... 39
Bartók, pizzicato ... 346
`bartype` ... 107
`base-shortest-duration` ... 578
`baseMoment` ... 88, 93
*BassFigure* ... 436, 438
*BassFigureAlignment* ... 436, 438
*BassFigureBracket* ... 436, 438
*BassFigureContinuation* ... 436, 438
`\bassFigureExtendersOff` ... 435
`\bassFigureExtendersOn` ... 435
*BassFigureLine* ... 436, 438
`\bassFigureStaffAlignmentDown` ... 437
`\bassFigureStaffAlignmentNeutral` ... 437
`\bassFigureStaffAlignmentUp` ... 437
basso ... 766
basso acustico ... 766
basso cifrato ... 433
basso continuo ... 433
basso continuo, allineamento ... 438
basso continuo, alterazione, posizione ... 436
basso continuo, linee di estensione ... 436
basso numerato ... 433
basso, chiave di ... 17, 702
batteria ... 398, 400
battiti per minuto ... 71
battito, disposizione delle travature ... 94
battito, raggruppamento ... 94
battuta del ritornello ... 101
battuta in levare ... 75
battuta, controlli ... 113
battuta, numero ... 108
battuta, numero, a intervalli regolari ... 109
battuta, numero, nella prima misura ... 109
battuta, numero, stile del ... 110
battuta, stanghetta manuale ... 101
battuta, stanghette ... 101
*bayati* ... 476
*Beam* ... 88, 96, 100, 335, 364
`\beam` ... 732
*beam-interface* ... 88, 96, 100
*Beam_engraver* ... 88, 100
*BeamEvent* ... 88, 100
`\beamExceptions` ... 88, 808
*BeamForbidEvent* ... 88, 96
`beatStructure` ... 88, 93
bemolle ... 6
bemolle, doppio ... 6
`\bendAfter` ... 141, 808
bequadro, impedirne l'inserimento nelle armature di chiave ... 23
bequadro, supplementare, impedire ... 7
*Bibliografia* ... 427, 431
binari ferroviari ... 141
`binding-offset` ... 546
bisbiglando ... 342
Bison ... 770
`\blackTriangleMarkup` ... 428
`blank-after-score-page-penalty` ... 548
`blank-last-page-penalty` ... 548

`blank-page-penalty` ... 548
blocco di commento ... 481, 485
blocco MIDI ... 526
BNF ... 770
`\bold` ... 248, 707
bongo ... 766
`\book` ... 481, 484
`\bookOutputName` ... 483, 808
`\bookOutputSuffix` ... 483, 808
`\bookpart` ... 482, 484, 558
`bookTitleMarkup` ... 494
`bottom-margin` ... 542
`\box` ... 255, 707
*brace* ... 198
*bracket* ... 198
`\bracket` ... 133, 732
`bracket` ... 340
`\bracket` ... 255
brano a più voci ... 307
`\break` ... 554
`break-align-symbols` ... 649
*break-alignment-interface* ... 787, 801
`break-visibility` ... 640
`breakable` ... 86
`breakbefore` ... 491
`\breathe` ... 139, 808
*Breathing_sign_engraver* ... 141
*BreathingEvent* ... 141
*BreathingSign* ... 141, 456
`BreathingSign` ... 140
*breve* ... 48, 61
`\breve` ... 46, 59
breve, nota, stile alternativo ... 48

## C

cabasa ... 766
*cadenza* ... 77, 122
cadenza ... 76, 121
cadenza, allineamento su ... 121
cadenza, alterazioni ... 76
cadenza, interruzioni di linea ... 77
cadenza, interruzioni di pagina ... 77
cadenza, numeri di battuta ... 76
cadenza, stanghette ... 76
cadenza, travature ... 76
`\cadenzaOff` ... 76
`\cadenzaOn` ... 76
*caesura* ... 141, 454
`\caesura` ... 454
callback ... 769
*Callback functions* ... 647
cambiare i tipi di carattere ... 248
cambiare il numero di punti di aumentazione ... 48
cambiare il separatore dell'accordo ... 431
cambiare il simbolo del rigo ... 633
cambiare il simbolo di respiro ... 140
cambiare il tempo senza il segno di metronomo ... 74
cambiare manualmente il rigo ... 333
cambio automatico di rigo ... 335
canali MIDI ... 533
cantante, nome del ... 300
cantico ... 318, 324
canto ... 318, 324
canzioniere ... 426
*Canzoni* ... 269, 306
capotasto ... 373
`\caps` ... 708
carattere non-ASCII ... 519
carattere riservato, stampare ... 247
carattere speciale ... 519
carattere speciale, in modalità markup ... 247
carattere tipografico, famiglia di ... 769
carta, formato ... 540
`\cavum` ... 456, 462
`\center-align` ... 251, 717
`\center-column` ... 253, 717
centering column of text ... 717
centrare il testo sulla pagina ... 253
cesura ... 141
`\change` ... 333
changing direction of text column ... 718
`\char` ... 754
`check-consistency` ... 546
chiamata, segno di ... 114
chiave ... 6, 17
chiave antica ... 17
chiave con notina (citazione in corpo piccolo) ... 18
chiave della notazione di Kiev ... 463
chiave delle percussioni ... 399
chiave di alto ... 702
chiave di baritono ... 17, 702
chiave di basso ... 17, 702
chiave di contralto ... 17
chiave di Do ... 17
Chiave di Fa ... 702
chiave di Fa ... 17
chiave di Kiev ... 445
chiave di mezzosoprano ... 17, 702
chiave di Petrucci ... 702
chiave di Sol ... 17, 702
chiave di Sol tenore ... 702
chiave di soprano ... 17, 702
chiave di subbasso ... 17
chiave di tenore ... 17, 702
chiave di tenore per coro ... 18
chiave di varbaritono ... 17
chiave di violino ... 17, 702
chiave francese ... 17
chiave gregoriana ... 452
chiave mensurale ... 445, 702
chiave mensurale nera ... 445
chiave per musica antica ... 445
chiave Petrucci ... 445
chiave traspositrice ... 18
chiave trasposta, visibilità della ... 644
chiave, intavolatura ... 366
chiave, moderntab ... 366
chiave, modifica delle proprietà ... 20
chiave, stile di ... 445
chiave, stili di ... 702
chiave, tablatura ... 366
chiave, visibilità della trasposizione ... 644
chiave, visibilità dopo un cambio esplicito ... 642
chiavi dell'intavolatura ... 702
chiavi della musica antica ... 702
chiavi delle percussioni ... 702
chitarra, accordi, tabella ... 83
chitarra, intavolatura ... 347

chitarra, ritmo di accompagnamento, mostrare ... 83, 84
chitarra, tablatura ... 347
chitarra, testa di nota ... 39
chiuso ... 124
*ChoirStaff* ... 198, 200, 308
`choral` ... 32
choral, stile delle alterazioni ... 32
`choral-cautionary` ... 32
choral-cautionary, stile delle alterazioni ... 32
*chord* ... 171, 420, 426
*Chord_name_engraver* ... 427
`chordChanges` ... 386, 426
`\chordmode` ... 6, 14, 379, 629
*ChordName* ... 427
`chordNameExceptions` ... 429
`chordNameLowercaseMinor` ... 427
*ChordNames* ... 209, 427
`ChordNames` ... 379
`ChordNames`, con stanghette ... 430
`chordNameSeparator` ... 428, 431
`chordNoteNamer` ... 428
`chordPrefixSpacer` ... 429
`\chordRepeats` ... 352, 808
`chordRootNamer` ... 427
*Chords* ... 420, 421, 424, 426, 427, 431, 433, 436, 438
`\chords` ... 425, 629
Christian Harmony, testa di nota ... 42
*church mode* ... 24
`\circle` ... 255, 733
circling text ... 733
`\circulus` ... 454, 766
citare le voci ... 213
citazione di un contesto ... 594
citazione in corpo più piccolo, togliere ... 221
citazione in corpo piccolo, chiave ... 18
citazione, nel testo vocale ... 268
claves ... 766
*clef* ... 446, 453, 463
*Clef* ... 22, 446
`\clef` ... 17, 808
*clef-interface* ... 22, 446
*Clef_engraver* ... 22, 446
*ClefModifier* ... 22, 446
`clip-regions` ... 521
closure ... 769
*cluster* ... 175
cluster ... 175
cluster di note ... 175
*Cluster_spanner_engraver* ... 175
*ClusterSpanner* ... 175
*ClusterSpannerBeacon* ... 175
`\cm` ... 632
coda ... 115, 124
`\coda` ... 124, 766
coda (uncinata) ... 448
coda mensurale ... 448
coda, segno, sulla stanghetta ... 242
codette ... 98
collisione ... 179
collisione di note ... 179
collisione trasversale ai righi ... 334
collisione, ignorare ... 174
collisione, numero di battuta ... 113
collisione, voci trasversali ai righi ... 334
collisione,ignorare ... 184
colonna, testo ... 246, 253
`color` ... 229
colorare un oggetto ... 229, 639
colorare una nota ... 229
colorare una voce ... 179
colorata, nota ... 229
colorato, oggetto ... 229
colore ... 229
colore in un accordo ... 231
colore rgb ... 230
colore x11 ... 231
colori, elenco dei ... 678
coloring text ... 760
`\column` ... 253, 717
`\column-lines` ... 761
*Combinare le note negli accordi* ... 171
combinare le parti, modifica del testo ... 189
combinare musica contrassegnata con etichette ... 518
combinatore delle parti ... 185
combinatore delle parti, modifica del testo ... 189
combinazione automatica delle parti ... 185
`\combine` ... 256, 718
*Come funzionano i file di input di LilyPond* ... 486, 492
comma, intervallo ... 478
commento ... 481, 485
commento multirigo ... 481, 485
*Common Practice Period* ... 10, 473
`common-shortest-duration` ... 578
*Completion_heads_engraver* ... 82
`Completion_heads_engraver` ... 81
*Completion_rest_engraver* ... 82
`Completion_rest_engraver` ... 81
`\compound-meter` ... 740
`\compoundMeter` ... 80, 808
`\compressMMRests` ... 62, 64, 808
comprimere la musica ... 54
`\concat` ... 718
concatenating text ... 718
*concert pitch* ... 28
condensare le pause normali ... 67
conduzione, segni ... 94
conga ... 766
`\consists` ... 600, 607
contatore della ripetizione con segno percentuale ... 165
contemporaneo, glissando ... 143
contenitore impuro, Scheme ... 658
contenitore puro, Scheme ... 658
*Contesti e incisori* ... 177, 592
contesto implicito ... 611
contesto, ciclo di vita ... 598
contesto, conservazione ... 598
contesto, creazione e citazione ... 594
contesto, definire nuovo ... 607
contesto, modifica proprietà di ... 602
contesto, ordine di disposizione ... 610
`\context` ... 594, 603
`\context` nel blocco `\layout` ... 603
`context-spec-music` ... 182
*ContextChange* ... 335
*Contexts* ... 563, 564, 568, 592
*Contexts and engravers* ... 592

conto della ripetizione con segno percentuale, visibilità ... 166
controlli del numero di battuta ... 113
controlli di battuta ... 113
controlling general text alignment ... 720
controllo dell'ottava ... 10
controllo della misura ... 113
controllo delle altezze ... 10
controllo di battuta, con ripetizioni ... 155
`controlpitch` ... 10
*Convenzioni per i nomi di oggetti e proprietà* ... 769, 770
copyright, segno ... 520
corda vuota, indicazione di ... 344
corda, numero di ... 348
cornamusa ... 409
cornamusa scozzese ... 409
cornamusa, esempio ... 409
coro, rigo per ... 195
corona ... 115, 124
corona, sulla stanghetta ... 242
*Correggere elementi della notazione sovrapposti* ... 334, 335
correzione dell'ottava ... 10
`countPercentRepeats` ... 165
cowbell ... 766
`\cr` ... 126
creating a table ... 761
creating empty text object ... 757
creating horizontal space, in text ... 722
creating text fraction ... 755
creating vertical space, in text ... 730, 759
creazione del rigo ... 194
`\cresc` ... 128
*crescendo* ... 132
crescendo ... 126
`crescendo-event` ... 216
`crescendoSpanner` ... 132
`crescendoText` ... 132
`\crescHairpin` ... 128
`\crescTextCresc` ... 128
`cross` ... 39
`\crossStaff` ... 338, 808
*cue-notes* ... 316
`\cueClef` ... 217, 808
`\cueClefUnset` ... 808
`\cueDuring` ... 217, 808
`\cueDuringWithClef` ... 217, 808
*CueVoice* ... 222, 317
`CueVoice` ... 217
cuica ... 766
`currentBarNumber` ... 108, 122
curva Bézier, punti di controllo ... 652
custodes ... 443
`\customTabClef` ... 740
*Custos* ... 444
*custos* ... 441, 444
custos ... 443

# D

D.S. al Fine ... 115
`\dashBang` ... 124
`\dashDash` ... 124
`\dashDot` ... 124
`\dashHat` ... 124
`\dashLarger` ... 124
`\dashPlus` ... 124
`\dashUnderscore` ... 124
dead note, percussioni ... 405
`\deadNote` ... 40, 808
`\deadNotesOff` ... 40
`\deadNotesOn` ... 40
decorazione del testo ... 255
`\decr` ... 126
`\decresc` ... 128
*decrescendo* ... 132
decrescendo ... 126
`decrescendoSpanner` ... 132
`decrescendoText` ... 132
`default-staff-staff-spacing` ... 561
*Default_bar_line_engraver* ... 81
`default` ... 28, 30
`\default` ... 114, 500
`defaultBarType` ... 107
`\defaultchild` ... 611
`\defaultTimeSignature` ... 68
`\defineBarLine` ... 105, 808
*Definire esplicitamente le voci* ... 177, 179
definire tastiera predefinita ... 384
definire una stanghetta ... 105
definizione di output ... 592
definizioni di contesto per il MIDI ... 531
delimitatore di inizio del sistema ... 195
delimitatore di inizio del sistema, annidato ... 198
`\deminutum` ... 456, 462
`\denies` ... 607, 609, 610
`\descendens` ... 456, 462
destra, mano, posizionamento diteggiatura ... 392
diagramma degli accordi ... 368, 378
diagramma degli accordi per chitarra ... 83
diagramma degli accordi, automatico ... 388
diagramma dei tasti ... 368, 378
diagramma dei tasti personalizzato ... 375
diagramma dei tasti, aggiungere diteggiatura ... 390
diagramma dei tasti, aggiungere personalizzato ... 381
diagramma dei tasti, automatico ... 388
diagramma dei tasti, cambiare orientamento ... 375
diagramma dei tasti, insieme ai nomi degli accordi ... 379
diagramma dei tasti, mandolino ... 378
diagramma dei tasti, personalizzato ... 376
diagramma dei tasti, trasposizione ... 380
diagramma dei tasti, ukulele ... 378
diagramma del pedale dell'arpa ... 342
diagramma per legni, elenchi delle chiavi ... 418
diagramma per legni, elenco ... 414
diagrammi dei tasti personalizzati ... 368
diagrammi per legni, modifica ... 417
diamante, testa di nota ... 39
diesis ... 6
diesis, doppio ... 6
*Difficult tweaks* ... 658
`dim` ... 421

`\dim` ... 128
dimensione ... 632
*Dimensione degli oggetti* ... 208
dimensione del testo ... 248
dimensione del tipo di carattere ... 248
dimensione del tipo di carattere (notazione) ... 222
dimensione standard del tipo di carattere (notazione) ... 226
`\dimHairpin` ... 128
diminuendo ... 126
`\dimTextDecr` ... 128
`\dimTextDecresc` ... 128
`\dimTextDim` ... 128
dinamica, indicazione, nuova ... 133
dinamiche ... 126
dinamiche assolute ... 126
dinamiche centrate nella musica per pianoforte ... 332
dinamiche del testo, stile ... 132
dinamiche editoriali ... 133
dinamiche nel MIDI ... 527
dinamiche personalizzate nel MIDI ... 527
dinamiche, allineamento verticale ... 131
dinamiche, centrarle nella musica per pianoforte ... 332
dinamiche, estensore del testo, personalizzazione ... 241
dinamiche, linea di estensione, nascondere ... 132
dinamiche, parentesi ... 133
dinamiche, posizionamento verticale ... 128
dinamiche, testo, stile ... 132
`\dir-column` ... 718
direzione, automatica, del gambo, rispetto alla linea centrale ... 232
direzione, predefinita, del gambo, rispetto alla linea centrale ... 232
diritto d'autore ... 520
`\discant` ... 749
discanto, simbolo, fisarmonica ... 341
disegnare il simbolo del rigo ... 633
disegnare legature di valore e di portamento ... 653
disegnare manualmente le legature di valore ... 58
disegnare un oggetto grafico ... 255
*Displaying music expressions* ... 538, 621
`\displayLilyMusic` ... 537, 809
`\displayMusic` ... 809
`\displayScheme` ... 809
disposizione automatica delle travature, proprietà delle indicazioni di tempo ... 68
disposizione delle travature che segue strettamente il battito ... 94
disposizione delle travature, proprietà predefinite delle indicazioni di tempo ... 68
distanza assoluta ... 632
distanza proporzionale ... 632
distanza, tra i righi ... 561
dita, cambio ... 226
diteggiatura ... 226
diteggiatura per accordo ... 227
diteggiatura, aggiungere al diagramma dei tasti ... 390
diteggiatura, dentro al rigo ... 228
diteggiatura, e pausa multipla ... 67
diteggiatura, grafico ... 408
diteggiatura, mano destra, per strumento a tasti ... 391
diteggiatura, mano destra, posizionamento ... 392
diteggiatura, orientamento ... 227
diteggiatura, simboli per legni ... 407
*divisio* ... 454
divisio ... 454
`\divisioMaior` ... 454
`\divisioMaxima` ... 454
`\divisioMinima` ... 454
divisione delle note ... 81
divisione delle pause ... 81
divisione, voce ... 309
divisiones ... 454
Do, chiave di ... 17
`dodecaphonic` ... 34
*dodecaphonic*, stile delle alterazioni ... 34
`dodecaphonic-first` ... 34
dodecaphonic-first, stile delle alterazioni ... 34
`dodecaphonic-no-repeat` ... 34
dodecaphonic-no-repeat, stile delle alterazioni ... 34
*doit* ... 142
doppie, legature di portamento, per accordi legati ... 137
doppio bemolle ... 6
doppio diesis ... 6
doppio punto, nota ... 47
`\dorian` ... 22
dorico ... 22
*DotColumn* ... 49
*Dots* ... 49
`\dotsDown` ... 47
`\dotsNeutral` ... 47
`\dotsUp` ... 47
*double flat* ... 8
*double sharp* ... 8
*Double_percent_repeat_engraver* ... 166
`\doubleflat` ... 741
*DoublePercentEvent* ... 166
*DoublePercentRepeat* ... 166
*DoublePercentRepeatCounter* ... 166
*DoubleRepeatSlash* ... 166
`\doublesharp` ... 741
`doubleSlurs` ... 137
`\downbow` ... 124, 344, 765
`\downmordent` ... 124, 764
`\downprall` ... 124, 764
`\draw-circle` ... 256, 733
`\draw-dashed-line` ... 733
`\draw-dotted-line` ... 734
`\draw-hline` ... 734
`\draw-line` ... 256, 734
`\draw-squiggle-line` ... 735
drawing beam, within text ... 732
drawing box, with rounded corners ... 736
drawing box, with rounded corners, around text ... 739
drawing circle, within text ... 733
drawing dashed line, within text ... 733
drawing dotted line, within text ... 734
drawing ellipse, around text ... 735
drawing line, across a page ... 734
drawing line, within text ... 734
drawing oval, around text ... 736
drawing path ... 737
drawing solid box, within text ... 736
drawing squiggled line, within text ... 735
drawing triangle, within text ... 739

\dropNote ... 809
\drummode ... 194, 398, 629
drumPitchNames ... 403
drumPitchTable ... 403
\drums ... 398, 629
*DrumStaff* ... 195, 405
DrumStaff ... 194
drumStyleTable ... 402
*DrumVoice* ... 405
durata isolata ... 47
durata predefinita ... 47
durata, di una nota ... 46
durata, scalare ... 53
*Duration names notes and rests* ... 48
\dwn ... 474
\dynamic ... 133, 708
dynamic-event ... 216
*Dynamic_performer* ... 528, 530, 531
\dynamicDown ... 128
*DynamicLineSpanner* ... 128, 132
DynamicLineSpanner ... 128, 131
\dynamicNeutral ... 128
*Dynamics* ... 132
*DynamicText* ... 132
\dynamicUp ... 128

## E

\easyHeadsOff ... 40
\easyHeadsOn ... 40
eccezione, nomi dell'accordo ... 429
*Editorial annotations* ... 226, 228, 229, 231, 232, 233, 234, 236
editoriali, dinamiche ... 133
edizione medicea ... 441
edizione vaticana ... 441
effetti nel MIDI ... 535
elencare i tipi di carattere disponibili ... 264
elenco dei colori ... 678
elenco dei diagrammi per legni ... 414
elenco delle chiavi nei diagrammi per legni ... 418
\ellipse ... 735
Emmentaler, font ... 679
encapsulated postscript, output ... 523
enclosing text in box, with rounded corners ... 739
enclosing text within a box ... 707
\endcr ... 126
\enddecr ... 126
\endSpanners ... 638, 809
*Engravers and Performers* ... 592
eolio ... 22
*Episema* ... 455
*Episema_engraver* ... 455
*EpisemaEvent* ... 455
\episemFinis ... 454
\episemInitium ... 454
EPS, output ... 523
\epsfile ... 256, 736
equalizzazione MIDI ... 527
equalizzazione, strumento, cambiare predefinita nel MIDI ... 530
esempio di musica araba ... 477
*Esempio musicale* ... 182, 184, 332, 333
esempio per cornamusa ... 409
espandere la musica ... 54
espressione nel MIDI ... 535
espressione, markup ... 247
*Espressioni musicali* ... 481
espressivo ... 124, 128
\espressivo ... 124, 128, 764
estensione ... 36
estensore ... 279
estensore del testo ... 240
estensore del testo, dinamiche, personalizzazione ... 241
estensore del testo, formattazione ... 241
estensore spezzato, modifica ... 656
estensore, modifica ... 656
\etc ... 661
etichetta ... 515
etichette, gruppi di ... 517
\eventChords ... 809
evento segnaposto ... 170
explicitClefVisibility ... 642
explicitKeySignatureVisibility ... 642
*Expressive marks* ... 59, 126, 132, 135, 138, 139, 141, 142, 147, 149, 152, 242, 393
extra-offset ... 561
\eyeglasses ... 755
Ez_numbers_engraver ... 41

## F

\f ... 126
Fa, chiave di ... 17, 702
*fall* ... 142
famiglia di font ... 250
famiglia di tipi di carattere ... 250
famiglia di tipi di carattere, impostare ... 264
\featherDurations ... 100, 809
\fermata ... 124, 741, 765
fermata su pausa multipla ... 64
Ferneyhough, forcella ... 130
Feta, font ... 679
\ff ... 126
\fff ... 126
\ffff ... 126
\fffff ... 126
fiati ... 406
*fifth* ... 6
*figured bass* ... 433
*FiguredBass* ... 209, 436, 438
figuredBassAlterationDirection ... 436
figuredBassPlusDirection ... 436
\figuremode ... 433, 629
\figures ... 433, 629
\fill-line ... 253, 719
\fill-with-pattern ... 719
\filled-box ... 256, 736
finale alternativo ... 153
finale alternativo, con legature di valore ... 155
finale alternato, ripetizioni ... 162
finali alternativi, e testo vocale ... 291
finalis ... 454
\finalis ... 454
fine linea, posizionamento segni ... 244
fine ripetizione ... 160
\finger ... 226, 708, 809

*finger-interface* ... 613
*Fingering* ... 228, 350, 612, 613
*fingering-event* ... 228, 612
*Fingering_engraver* ... 228, 612, 614
*FingeringEvent* ... 228, 612
`fingeringOrientations` ... 227
`first-page-number` ... 549
`\first-visible` ... 755
fisarmonica ... 341
fisarmonica, simbolo di discanto ... 341
fisarmonica, simbolo di registro ... 341
`\fixed` ... 2, 809
*flag* ... 442, 448
`\flageolet` ... 124, 406, 765
flageolet, modifica dimensione ... 406
*flat* ... 8
`\flat` ... 741
Flex ... 770
`\flexa` ... 462
floor tom tom ... 766
*Fogli di stile* ... 519
`followVoice` ... 337
fondamentale dell'accordo ... 420
font ... 261, 769
font (notazione), dimensione standard ... 226
font Emmentaler ... 679
font Feta ... 679
font non testuali, nel markup ... 261
font Parmesan ... 679
font, cambiare ... 248
font, dimensione ... 248
font, famiglia ... 250
font, impostare ... 264
font, impostare la dimensione ... 552
font, modificarli per l'intero documento ... 264
font, ridimensionamento ... 222
*font-interface* ... 226, 613, 757
`font-interface` ... 226, 261
`font-size` ... 222, 226
`\fontCaps` ... 708
`\fontsize` ... 248, 708
`fontSize` ... 222
`\footnote` ... 499, 755, 809
*Footnote_engraver* ... 508
*FootnoteEvent* ... 508
*FootnoteItem* ... 508
*FootnoteSpanner* ... 508
*Forbid_line_break_engraver* ... 82
`Forbid_line_break_engraver` ... 53
forcella ... 126
forcella allargata (flared-hairpins) ... 130
forcella angolata ... 645
forcella continua (constante-hairpins) ... 130
forcella Ferneyhough ... 130
forcella, al niente ... 130
forcella, lunghezza minima ... 129
forcella, spostarre le estremità della ... 130
forcella, terminata sulle stanghette ... 129
`forget` ... 35
`forget`, stile delle alterazioni ... 35
forma di un accordo, per strumenti con tasti ... 382
formato carta ... 540
formato carta, orientamento ... 541
formato carta, orizzontale (landscape) ... 541
formato del segno di chiamata ... 115
formato pagina ... 540
formattare un estensore del testo ... 241
formattazione del gruppo irregolare ... 50
formattazione dell'accordo di settima maggiore ... 430
formattazione della pagina ... 582
formattazione della terzina ... 50
formattazione mensurale ... 198
formattazione, nel testo vocale ... 268
formattazione, notine ... 217
formattazione, oggetto ... 769
*Formatting text* ... 795, 804
`four-string-banjo` ... 397
`\fp` ... 126
`\fraction` ... 755
frammento ... 217
frammento musicale ... 521
frammento, citare ... 213
francese, chiave ... 17
frase, legatura di ... 138
fraseggio, nel testo vocale ... 276
`\freeBass` ... 750
`\frenchChords` ... 427
*Frenched score* ... 311
*Frenched staff* ... 208, 209
*Frenched staves* ... 311
`\fret-diagram` ... 368, 746
fret-diagram, diagramma dei tasti ... 368
*fret-diagram-interface* ... 375, 378, 383, 388, 391
`fret-diagram-interface` ... 375
`\fret-diagram-terse` ... 371, 746
fret-diagram-terse, diagramma dei tasti conciso ... 371
`\fret-diagram-verbose` ... 373, 747
fret-diagram-verbose, diagramma dei tasti ridondante ... 373
`FretBoards` ... 378
*Fretted strings* ... 350, 364, 368, 378, 388, 391, 393, 395, 396, 397
frigio ... 22
`\fromproperty` ... 755
Funk, testa di nota ... 42
`\funkHeads` ... 42
`\funkHeadsMinor` ... 43

# G

G, chiave di ... 702
G2, chiave di ... 702
gambo ... 232
gambo barrato ... 118
gambo che attraversa i righi ... 338
gambo invisibile ... 232
gambo orizzontale ... 356
gambo trasversale ai righi ... 338
gambo, direzione ... 232
gambo, direzione automatica rispetto alla linea centrale ... 232
gambo, direzione predefinita rispetto alla linea centrale ... 232
gambo, giù ... 232
gambo, in intavolatura ... 356
gambo, neutrale ... 232
gambo, su ... 232
`\general-align` ... 252, 720
`\germanChords` ... 427
gestione del tempo ... 122

GG, chiave di ... 702
ghost note ... 231
ghost note, percussioni ... 405
giustificato, testo ... 254
*Gli incisori* ... 82
glifo ... 769
glifo musicale ... 115
*glissando* ... 146
*Glissando* ... 147, 638
glissando ... 142
`\glissando` ... 142
glissando che vanno a capo ... 145
glissando contemporaneo ... 143
glissando, accordi in intavolatura ... 362
glissando, attraverso le ripetizioni ... 145
glissando, e ripetizioni ... 160
glissando, segni di tempo ... 144
`\glissandoMap` ... 143
glissato, in intavolatura ... 361
glyph ... 769
`\grace` ... 116, 809
*grace notes* ... 120
*Grace_auto_beam_engraver* ... 120
*Grace_beam_engraver* ... 120
*Grace_engraver* ... 120
*Grace_spacing_engraver* ... 120
*GraceMusic* ... 120
grado dell'accordo, alterazione ... 423
graffa verticale ... 195
graffa, varie dimensioni ... 261
grafica inclusa ... 256
grafica, inclusione ... 255
grafico della diteggiatura ... 408
grammatica di LilyPond ... 770
*grand staff* ... 198
*GrandStaff* ... 35, 198
*Graphical Object Interfaces* ... 770
gregoriana, alterazione ... 453
gregoriana, armatura di chiave ... 453
gregoriana, articolazione ... 454
gregoriana, musica, trascrizione moderna ... 324
gregoriano, legatura di neumi quadrati ... 456
*GregorianTranscriptionStaff* ... 195
`GregorianTranscriptionStaff` ... 194
*grid-line-interface* ... 236
*grid-point-interface* ... 236
*Grid_line_span_engraver* ... 236
`Grid_line_span_engraver` ... 234
*Grid_point_engraver* ... 236
`Grid_point_engraver` ... 234
`gridInterval` ... 234
*GridLine* ... 236
*GridPoint* ... 236
griglia ... 234
grob ... 612, 769
grob, proprietà ... 617
grob, sovrascrivere ... 639
grob, visibilità del ... 638
*grob-interface* ... 613, 769, 772
`grob-interface` ... 770
`\grobdescriptions` ... 809
`grow-direction` ... 100
gruppetto ... 124
gruppetto rovesciato ... 124
gruppo di righi ... 195
gruppo di righi, con parentesi quadra all'inizio ... 197
gruppo irregolare ... 49
gruppo irregolare con travature, interruzione di linea ... 53
gruppo irregolare, formattazione del ... 50
gruppo irregolare, inserirne multipli ... 50
gruppo irregolare, modifiche del numero del ... 50
gruppo irregolare, posizionamento della parentesi quadra ... 49
gruppo irregolare, raggruppamento ... 49
gruppo ritardato, ornamento ... 125
Guida al funzionamento interno ... 592
guide ... 443
guiro ... 766

# H

*hairpin* ... 132
*Hairpin* ... 132, 622, 623
half-open high hat ... 766
`\halfopen` ... 124, 765
`\halign` ... 251, 721
hammer on ... 363
handclap ... 766
Harmonia Sacra, testa di nota ... 42
`\harmonic` ... 40, 345, 355
`\harmonicByFret` ... 355, 809
`\harmonicByRatio` ... 355, 809
`\harmonicNote` ... 810
*harmonics* ... 346
`\harmonicsOff` ... 345
`\harmonicsOn` ... 345, 810
`\harp-pedal` ... 748
`\hbracket` ... 255, 736
`\hcenter-in` ... 722
`\header` ... 484
`\hide` ... 639, 810
`\hideKeySignature` ... 409
`\hideNotes` ... 229
`\hideSplitTiedTabNotes` ... 354
`\hideStaffSwitch` ... 337
high bongo ... 766
high conga ... 766
high hat ... 766
high timbale ... 766
*horizontal-bracket-interface* ... 238
*horizontal-bracket-text-interface* ... 238
`horizontal-shift` ... 547
*Horizontal_bracket_engraver* ... 238
`Horizontal_bracket_engraver` ... 236
*HorizontalBracket* ... 238
*HorizontalBracketText* ... 238
`HorizontalBracketText` ... 237
horizontally centering text ... 717
`\hspace` ... 722
hufnagel ... 441
`\huge` ... 222, 251, 709

# I


`\ictus` ... 454, 766
`\iij` ... 457
`\IIJ` ... 457
`\ij` ... 457
`\IJ` ... 457
immagine, inclusione ... 256
immutabile, oggetto ... 770
immutable ... 770
importing stencil, into text ... 759
*Impostare canzoni semplici* ... 267, 268
`\improvisationOff` ... 45, 83
`\improvisationOn` ... 45, 83
improvvisazione ... 45
improvvisazione araba ... 476
`\in` ... 632
`\incipit` ... 466, 810
incipit, aggiungere ... 466
incisore, includere nel contesto ... 607
`\inclinatum` ... 456, 462
inclinazione, estensore dell'ottava, modifica ... 26
`\include` ... 485, 511
includere le impostazioni ... 519
inclusione di file ... 511
incorniciatura del testo ... 255
`indent` ... 211, 547, 582
indicare la posizione e il barré per uno strumento con tasti ... 393
indicazione di arcata ... 344
indicazione di armonici, nell'intavolatura ... 355
indicazione di barré ... 369
indicazione di corda vuota ... 344
indicazione di diteggiatura per accordo ... 227
indicazione di No Chord in `ChordNames` ... 425
indicazione di tempo ... 67
indicazione di tempo composto ... 80
indicazione di tempo doppia ... 78
indicazione di tempo polimetrico ... 78
indicazione di tempo, impostazioni predefinite ... 68
indicazione di tempo, mensurale ... 446
indicazione di tempo, mostrare solo il numeratore ... 71
indicazione di tempo, multipla ... 601
indicazione di tempo, proprietà, ripristinare i valori predefiniti ... 69
indicazione di tempo, stile ... 68, 446
indicazione di tempo, visibilità ... 67
indicazione dinamica nuova ... 133
indicazione dinamica, più di un segno su una nota ... 127
indicazione manuale di ripetizione ... 160
indicazione metronomica ... 71
indicazione polimetrica ... 78
indicazione testuale ... 242
indicazioni di tempo arabe ... 476
indicazioni dinamiche multiple, su una nota ... 127
informazioni sul tempo e ripetizioni ... 160
ingrandimento del tipo di carattere ... 222
`\inherit-acceptability` ... 609, 810
inizializzazione del rigo ... 194
inizio del sistema ... 195
inizio ripetizione ... 160
inlining an Encapsulated PostScript image ... 736
`inner-margin` ... 546
inno ... 307, 318, 324
inno, e misure parziali ... 329
insegnamento (*teaching*), stile delle alterazioni ... 34
inserimento del testo ... 268
inserimento di gruppi irregolari multipli ... 50
inserting music, into text ... 743
inserting PostScript directly, into text ... 738
inserting URL link, into text ... 740
`\inStaffSegno` ... 156, 810
*instrument-specific-markup-interface* ... 419, 757
*InstrumentName* ... 213
`\instrumentSwitch` ... 810
intavolatura ... 194, 347
intavolatura e armonici ... 355
intavolatura per banjo ... 347, 396
intavolatura per chitarra ... 347
intavolatura per liuto ... 397
intavolatura personalizzata ... 364
intavolatura, accordature personalizzate ... 365
intavolatura, accordature predefinite ... 364
intavolatura, banjo ... 364
intavolatura, base ... 350
intavolatura, basso ... 364
intavolatura, basso elettrico ... 364
intavolatura, chiave ... 366
intavolatura, chiavi di ... 702
intavolatura, chitarra ... 364
intavolatura, contrabbasso ... 364
intavolatura, e armonici ... 357
intavolatura, e gambi ... 356
intavolatura, e polifonia ... 357
intavolatura, e slide ... 361
intavolatura, e travature ... 356
intavolatura, glissando di accordi ... 362
intavolatura, mandolino ... 364
intavolatura, martellato ... 363
intavolatura, microtono ... 367
intavolatura, predefinita ... 350
intavolatura, quarto di tono ... 367
intavolatura, strappato ... 363
intavolatura, ukulele ... 364
intavolatura, viola ... 364
intavolatura, violino ... 364
intavolatura, violoncello ... 364
interfaccia ... 770
interfaccia dell'oggetto grafico ... 770
interface, layout ... 612
*Interfaces for programmers* ... 646
*Intermediate substitution functions* ... 623, 657
Internals Reference ... 592
interruzione di linea ... 101
interruzione di linea manuale ... 554
interruzione di linea regolare ... 556
interruzione di linea, cadenze ... 77
interruzione di linea, musica in tempo libero ... 77
interruzione di linea, travature ... 86
interruzione di pagina ... 582
interruzione di pagina, cadenze ... 77
interruzione di pagina, musica in tempo libero ... 77
interruzione, di pagina o linea, gestione con una voce apposita ... 556
interruzione, nella musica in tempo libero ... 77
*interval* ... 6
intervallo intermedio ... 473
intervallo, comma ... 478

intestazione ... 486
\inversion ... 14, 810
inversione ... 14
inversione modale ... 16
inversioni degli accordi ... 423
\invertChords ... 810
invisibile, gambo ... 232
invisibile, nota ... 229
\ionian ... 22
ionio ... 22
*iraq* ... 476
isolata, altezza ... 47
isolata, durata ... 47
isolati, segni percentuali, nelle ripetizioni ... 166
\italianChords ... 427
\italic ... 248, 709
*item-interface* ... 613

## J

jazz, accordo ... 427
\justified-lines ... 260, 761
\justify ... 254, 724
\justify-field ... 723
\justify-line ... 723
\justify-string ... 724
justifying lines of text ... 761
justifying text ... 724

## K

\keepWithTag ... 515, 810
\key ... 22, 43, 810
*key signature* ... 449, 454
*key-signature-interface* ... 24
*Key_engraver* ... 24
*Key_performer* ... 24
*Keyboards* ... 333, 335, 337, 338, 340, 341, 342
*KeyCancellation* ... 24
*KeyChangeEvent* ... 24
*KeySignature* ... 24, 449, 454, 476
Kiev, chiave ... 445
Kiev, notazione di, alterazione ... 464
Kiev, notazione di, legatura ... 464
Kiev, notazione, chiave ... 463
*kievan notation* ... 463, 464
\kievanOff ... 463
\kievanOn ... 463
KievanStaff ... 462
KievanVoice ... 462
\killCues ... 221, 810
kirchenpausen ... 64
*kurd* ... 476

## L

*La partitura è una (singola) espressione musicale composta* ... 481
*La proprietà outside-staff-priority* ... 576
\label ... 509, 810
*laissez vibrer* ... 59
laissez vibrer ... 56
\laissezVibrer ... 56
*LaissezVibrerTie* ... 59
*LaissezVibrerTieColumn* ... 59
\language ... 810
\languageRestore ... 810
\languageSaveAndChange ... 810
\large ... 222, 251, 709
\larger ... 248, 251, 709
last-bottom-spacing ... 544
*Lavorare sui file di input* ... 481
\layout ... 484, 550, 592, 603
layout interface ... 612
layout, file ... 552
layout-set-staff-size ... 552
*Le voci contengono la musica* ... 179, 184
lead sheet ... 426
*ledger line* ... 204
*ledger-line-spanner-interface* ... 40
*Ledger_line_engraver* ... 40
*LedgerLineSpanner* ... 40
\left-align ... 251, 725
left-aligning text ... 725
\left-brace ... 756
\left-column ... 725
left-margin ... 545
legato discendente ... 363
legatura ... 442
legatura di frase ... 135, 138
legatura di frase puntata ... 138
legatura di frase tratteggiata ... 138
legatura di frase, definizione dei modelli di tratteggio ... 139
legatura di frase, metà continua e metà tratteggiata ... 138
legatura di Kiev ... 464
legatura di neumi quadrati ... 456
legatura di portamento ... 135
legatura di portamento continua ... 135
legatura di portamento multipla ... 135
legatura di portamento punteggiata ... 135
legatura di portamento simultanea ... 135
legatura di portamento tratteggiata ... 135
legatura di portamento, definizione dei modelli di tratteggio ... 137
legatura di portamento, definizione dei modelli di tratteggio per il fraseggio ... 139
legatura di portamento, definizione del modello di tratteggio ... 136
legatura di portamento, e ripetizioni ... 160
legatura di portamento, frase puntata ... 138
legatura di portamento, frase tratteggiata ... 138
legatura di portamento, fraseggio multiplo ... 138
legatura di portamento, fraseggio simultaneo ... 138
legatura di portamento, fraseggio, definizione dei modelli di tratteggio ... 139
legatura di portamento, metà tratteggiata e metà continua ... 136

legatura di portamento, modificare ..... 652
legatura di portamento,
posizionamento manuale ..... 135
legatura di portamento, sopra le note ..... 135
legatura di portamento, sotto le note ..... 135
legatura di portamento, testo all'interno ..... 137
legatura di portamento, tratto metà continuo
e metà tratteggiato ..... 138
legatura di valore ..... 55
legatura di valore punteggiata ..... 56
legatura di valore tratteggiata ..... 56
legatura di valore, aspetto ..... 56
legatura di valore, con arpeggi ..... 58
legatura di valore, disegno manuale ..... 58
legatura di valore, e alterazione ..... 7
legatura di valore, e parentesi della volta ..... 56
legatura di valore, finali alternativi ..... 155
legatura di valore, in un accordo ..... 55
legatura di valore, laissez vibrer ..... 56
legatura di valore, modificare ..... 652
legatura di valore, nel testo vocale ..... 275
legatura di valore, nelle ripetizioni ..... 155
legatura di valore, posizionamento ..... 56
legatura di valore, ripetizione ..... 56
legature di frase multiple ..... 138
legature di frase simultanee ..... 138
legature di portamento doppie
per accordi legati ..... 137
legature di portamento, stile ..... 135
legature mensurali bianche ..... 450
legni, simboli di diteggiatura ..... 407
lexer ..... 770
`\lheel` ..... 124, 765
lidio ..... 22
*ligature* ..... 441, 443, 451, 452, 462, 465
ligature, in text ..... 718
*LilyPond grammar* ..... 771
*line* ..... 204
`\line` ..... 725
*line-spanner-interface* ..... 638
`line-width` ..... 545, 582
`\linea` ..... 456, 462
linea del cambio rigo ..... 337
linea del rigo, fermare e avviare ..... 201
linea del rigo, modificare ..... 201
linea di commento ..... 481, 485
linea di commento, singola ..... 481, 485
linea di estensione, dinamiche, nascondere ..... 132
linea trasversale ai righi ..... 337
linea verticale tra i righi ..... 234
linea, cambio di rigo ..... 337
linea, fine, posizionamento segni ..... 244
linea, interruzione, gestione con
una voce apposita ..... 556
*LineBreakEvent* ..... 557
linee della griglia, modificare aspetto ..... 235
linee di estensione del basso continuo ..... 436
`\lineprall` ..... 124, 764
lingua, nomi delle altezze in un'altra ..... 8
lingua, nomi delle note in un'altra ..... 8
lista di associazioni ..... 769
liuto, accordatura ..... 398
liuto, intavolatura ..... 397
livello del chorus nel MIDI ..... 535
`\locrian` ..... 22
locrio ..... 22
*longa* ..... 48, 61
`\longa` ..... 46, 59
`\longfermata` ..... 124, 765
`\lookup` ..... 756
low bongo ..... 766
low conga ..... 766
low timbale ..... 766
`\lower` ..... 252, 726
lowering text ..... 726
`\ltoe` ..... 124, 765
lunghezza di una nota ..... 46
*Lunghezza e spessore degli oggetti* ..... 208, 632
lunghezza minima, forcella ..... 129
`ly:add-context-mod` ..... 821
`ly:add-file-name-alist` ..... 821
`ly:add-interface` ..... 821
`ly:add-listener` ..... 821
`ly:add-option` ..... 821
`ly:all-grob-interfaces` ..... 821
`ly:all-options` ..... 821
`ly:all-stencil-expressions` ..... 821
`ly:angle` ..... 821
`ly:assoc-get` ..... 821
`ly:axis-group-interface::add-element` ..... 821
`ly:basic-progress` ..... 821
`ly:beam-score-count` ..... 821
`ly:book-add-bookpart!` ..... 821
`ly:book-add-score!` ..... 821
`ly:book-book-parts` ..... 822
`ly:book-header` ..... 822
`ly:book-paper` ..... 822
`ly:book-process` ..... 822
`ly:book-process-to-systems` ..... 822
`ly:book-scores` ..... 822
`ly:book-set-header!` ..... 822
`ly:book?` ..... 821
`ly:box?` ..... 822
`ly:bp` ..... 822
`ly:bracket` ..... 822
`ly:broadcast` ..... 822
`ly:camel-case->lisp-identifier` ..... 822
`ly:chain-assoc-get` ..... 822
`ly:check-expected-warnings` ..... 822
`ly:cm` ..... 822
`ly:command-line-code` ..... 822
`ly:command-line-options` ..... 822
`ly:connect-dispatchers` ..... 823
`ly:context-current-moment` ..... 823
`ly:context-def-lookup` ..... 823
`ly:context-def-modify` ..... 823
`ly:context-def?` ..... 823
`ly:context-event-source` ..... 823
`ly:context-events-below` ..... 823
`ly:context-find` ..... 823
`ly:context-grob-definition` ..... 823
`ly:context-id` ..... 823
`ly:context-matched-pop-property` ..... 823
`ly:context-mod-apply!` ..... 823
`ly:context-mod?` ..... 823
`ly:context-name` ..... 823
`ly:context-now` ..... 823
`ly:context-parent` ..... 823
`ly:context-property` ..... 823
`ly:context-property-where-defined` ..... 824

ly:context-pushpop-property ..... 824
ly:context-set-property! ..... 824
ly:context-unset-property ..... 824
ly:context? ..... 823
ly:debug ..... 824
ly:default-scale ..... 824
ly:dimension? ..... 824
ly:dir? ..... 824
ly:directed ..... 824
ly:disconnect-dispatchers ..... 824
ly:dispatcher? ..... 824
ly:duration->string ..... 824
ly:duration-dot-count ..... 824
ly:duration-factor ..... 824
ly:duration-length ..... 824
ly:duration-log ..... 824
ly:duration-scale ..... 825
ly:duration<? ..... 824
ly:duration? ..... 824
ly:effective-prefix ..... 825
ly:encode-string-for-pdf ..... 825
ly:engraver-announce-end-grob ..... 825
ly:engraver-make-grob ..... 825
ly:error ..... 825
ly:event-deep-copy ..... 825
ly:event-property ..... 825
ly:event-set-property! ..... 825
ly:event? ..... 825
ly:expand-environment ..... 825
ly:expect-warning ..... 825
ly:extract-subfont-from-collection ..... 825
ly:find-file ..... 825
ly:font-config-add-directory ..... 825
ly:font-config-add-font ..... 825
ly:font-config-display-fonts ..... 826
ly:font-config-get-font-file ..... 826
ly:font-design-size ..... 826
ly:font-file-name ..... 826
ly:font-get-glyph ..... 826
ly:font-glyph-name-to-charcode ..... 826
ly:font-glyph-name-to-index ..... 826
ly:font-index-to-charcode ..... 826
ly:font-magnification ..... 826
ly:font-metric? ..... 826
ly:font-name ..... 826
ly:font-sub-fonts ..... 826
ly:format ..... 826
ly:format-output ..... 826
ly:generic-bound-extent ..... 827
ly:get-all-function-documentation ..... 827
ly:get-all-translators ..... 827
ly:get-cff-offset ..... 827
ly:get-context-mods ..... 827
ly:get-font-format ..... 827
ly:get-option ..... 827
ly:get-spacing-spec ..... 827
ly:get-undead ..... 827
ly:gettext ..... 827
ly:grob-alist-chain ..... 827
ly:grob-array->list ..... 827
ly:grob-array-length ..... 827
ly:grob-array-ref ..... 827
ly:grob-array? ..... 827
ly:grob-basic-properties ..... 827
ly:grob-chain-callback ..... 828
ly:grob-common-refpoint ..... 828
ly:grob-common-refpoint-of-array ..... 828
ly:grob-default-font ..... 828
ly:grob-extent ..... 828
ly:grob-get-vertical-axis-group-index ..... 828
ly:grob-interfaces ..... 828
ly:grob-layout ..... 828
ly:grob-object ..... 828
ly:grob-original ..... 828
ly:grob-parent ..... 828
ly:grob-pq<? ..... 828
ly:grob-properties? ..... 828
ly:grob-property ..... 828
ly:grob-property-data ..... 828
ly:grob-pure-height ..... 828
ly:grob-pure-property ..... 828
ly:grob-relative-coordinate ..... 828
ly:grob-robust-relative-extent ..... 829
ly:grob-script-priority-less ..... 829
ly:grob-set-nested-property! ..... 829
ly:grob-set-object! ..... 829
ly:grob-set-parent! ..... 829
ly:grob-set-property! ..... 829
ly:grob-spanned-rank-interval ..... 829
ly:grob-staff-position ..... 829
ly:grob-suicide! ..... 829
ly:grob-system ..... 829
ly:grob-translate-axis! ..... 829
ly:grob-vertical<? ..... 829
ly:grob? ..... 827
ly:gulp-file ..... 829
ly:has-glyph-names? ..... 829
ly:hash-table-keys ..... 829
ly:inch ..... 829
ly:input-both-locations ..... 829
ly:input-file-line-char-column ..... 829
ly:input-location? ..... 830
ly:input-message ..... 830
ly:input-warning ..... 830
ly:interpret-music-expression ..... 830
ly:intlog2 ..... 830
ly:item-break-dir ..... 830
ly:item-get-column ..... 830
ly:item? ..... 830
ly:iterator? ..... 830
ly:length ..... 830
ly:lily-lexer? ..... 830
ly:lily-parser? ..... 830
ly:line-interface::line ..... 830
ly:listened-event-class? ..... 830
ly:listened-event-types ..... 830
ly:listener? ..... 830
ly:make-book ..... 830
ly:make-book-part ..... 830
ly:make-context-mod ..... 831
ly:make-dispatcher ..... 831
ly:make-duration ..... 831
ly:make-global-context ..... 831
ly:make-global-translator ..... 831
ly:make-grob-properties ..... 831
ly:make-moment ..... 831
ly:make-music ..... 831
ly:make-music-function ..... 831
ly:make-music-relative! ..... 831
ly:make-output-def ..... 831

ly:make-page-label-marker ..... 831
ly:make-page-permission-marker ..... 832
ly:make-pango-description-string ..... 832
ly:make-paper-outputter ..... 832
ly:make-pitch ..... 832
ly:make-prob ..... 832
ly:make-rotation ..... 832
ly:make-scale ..... 832
ly:make-scaling ..... 832
ly:make-score ..... 832
ly:make-spring ..... 832
ly:make-stencil ..... 832
ly:make-stream-event ..... 832
ly:make-transform ..... 832
ly:make-translation ..... 833
ly:make-undead ..... 833
ly:make-unpure-pure-container ..... 833
ly:message ..... 833
ly:minimal-breaking ..... 833
ly:minimal-breaking ..... 559
ly:mm ..... 833
ly:module->alist ..... 833
ly:module-copy ..... 833
ly:modules-lookup ..... 833
ly:moment-add ..... 833
ly:moment-div ..... 833
ly:moment-grace ..... 833
ly:moment-grace-denominator ..... 833
ly:moment-grace-numerator ..... 833
ly:moment-main ..... 833
ly:moment-main-denominator ..... 834
ly:moment-main-numerator ..... 834
ly:moment-mod ..... 834
ly:moment-mul ..... 834
ly:moment-sub ..... 834
ly:moment<? ..... 833
ly:moment? ..... 833
ly:music-compress ..... 834
ly:music-deep-copy ..... 834
ly:music-duration-compress ..... 834
ly:music-duration-length ..... 834
ly:music-function-extract ..... 834
ly:music-function-signature ..... 834
ly:music-function? ..... 834
ly:music-length ..... 834
ly:music-list? ..... 834
ly:music-mutable-properties ..... 834
ly:music-output? ..... 834
ly:music-property ..... 834
ly:music-set-property! ..... 835
ly:music-transpose ..... 835
ly:music? ..... 834
ly:note-column-accidentals ..... 835
ly:note-column-dot-column ..... 835
ly:note-head::stem-attachment ..... 835
ly:number->string ..... 835
ly:one-line-auto-height-breaking ..... 835
ly:one-line-auto-height-breaking ..... 559
ly:one-line-breaking ..... 835
ly:one-line-breaking ..... 559
ly:one-page-breaking ..... 835
ly:one-page-breaking ..... 559
ly:optimal-breaking ..... 835
ly:optimal-breaking ..... 559
ly:option-usage ..... 835
ly:otf->cff ..... 835
ly:otf-font-glyph-info ..... 835
ly:otf-font-table-data ..... 835
ly:otf-font? ..... 835
ly:otf-glyph-count ..... 835
ly:otf-glyph-list ..... 836
ly:output-def-clone ..... 836
ly:output-def-lookup ..... 836
ly:output-def-parent ..... 836
ly:output-def-scope ..... 836
ly:output-def-set-variable! ..... 836
ly:output-def? ..... 836
ly:output-description ..... 836
ly:output-find-context-def ..... 836
ly:output-formats ..... 836
ly:outputter-close ..... 836
ly:outputter-dump-stencil ..... 836
ly:outputter-dump-string ..... 836
ly:outputter-output-scheme ..... 836
ly:outputter-port ..... 836
ly:page-marker? ..... 836
ly:page-turn-breaking ..... 836
ly:page-turn-breaking ..... 560
ly:pango-font-physical-fonts ..... 836
ly:pango-font? ..... 836
ly:paper-book-header ..... 837
ly:paper-book-pages ..... 837
ly:paper-book-paper ..... 837
ly:paper-book-performances ..... 837
ly:paper-book-scopes ..... 837
ly:paper-book-systems ..... 837
ly:paper-book? ..... 837
ly:paper-column::break-align-width ..... 837
ly:paper-column::print ..... 837
ly:paper-fonts ..... 837
ly:paper-get-font ..... 837
ly:paper-get-number ..... 837
ly:paper-outputscale ..... 837
ly:paper-score-paper-systems ..... 837
ly:paper-system-minimum-distance ..... 837
ly:paper-system? ..... 837
ly:parse-file ..... 837
ly:parse-string-expression ..... 838
ly:parsed-undead-list! ..... 838
ly:parser-clear-error ..... 838
ly:parser-clone ..... 838
ly:parser-define! ..... 838
ly:parser-error ..... 838
ly:parser-has-error? ..... 838
ly:parser-include-string ..... 838
ly:parser-lexer ..... 838
ly:parser-lookup ..... 838
ly:parser-output-name ..... 838
ly:parser-parse-string ..... 838
ly:parser-set-note-names ..... 838
ly:performance-headers ..... 838
ly:performance-write ..... 838
ly:pitch-alteration ..... 839
ly:pitch-diff ..... 839
ly:pitch-negate ..... 839
ly:pitch-notename ..... 839
ly:pitch-octave ..... 839
ly:pitch-quartertones ..... 839
ly:pitch-semitones ..... 839
ly:pitch-steps ..... 839

ly:pitch-tones ... 839
ly:pitch-transpose ... 839
ly:pitch<? ... 838
ly:pitch? ... 838
ly:pointer-group-interface::add-grob ... 839
ly:position-on-line? ... 839
ly:prob-immutable-properties ... 839
ly:prob-mutable-properties ... 839
ly:prob-property ... 839
ly:prob-property? ... 839
ly:prob-set-property! ... 839
ly:prob-type? ... 839
ly:prob? ... 839
ly:programming-error ... 840
ly:progress ... 840
ly:property-lookup-stats ... 840
ly:protects ... 840
ly:pt ... 840
ly:pure-call ... 840
ly:randomize-rand-seed ... 840
ly:register-stencil-expression ... 840
ly:register-translator ... 840
ly:relative-group-extent ... 840
ly:rename-file ... 840
ly:reset-all-fonts ... 840
ly:round-filled-box ... 840
ly:round-filled-polygon ... 840
ly:run-translator ... 840
ly:score-add-output-def! ... 841
ly:score-embedded-format ... 841
ly:score-error? ... 841
ly:score-header ... 841
ly:score-music ... 841
ly:score-output-defs ... 841
ly:score-set-header! ... 841
ly:score? ... 841
ly:separation-item::print ... 841
ly:set-default-scale ... 841
ly:set-grob-modification-callback ... 841
ly:set-middle-C! ... 841
ly:set-option ... 841
ly:set-origin! ... 841
ly:set-property-cache-callback ... 842
ly:skyline-empty? ... 842
ly:skyline-pair? ... 842
ly:skyline? ... 842
ly:slur-score-count ... 842
ly:smob-protects ... 842
ly:solve-spring-rod-problem ... 842
ly:source-file? ... 842
ly:source-files ... 842
ly:spanner-bound ... 842
ly:spanner-broken-into ... 842
ly:spanner-set-bound! ... 842
ly:spanner? ... 842
ly:spawn ... 842
ly:spring-set-inverse-compress-strength! ... 843
ly:spring-set-inverse-stretch-strength! ... 843
ly:spring? ... 842
ly:staff-symbol-line-thickness ... 843
ly:staff-symbol-staff-radius ... 843
ly:staff-symbol-staff-space ... 843
ly:start-environment ... 843
ly:stderr-redirect ... 843
ly:stencil-add ... 843
ly:stencil-aligned-to ... 843
ly:stencil-combine-at-edge ... 843
ly:stencil-empty? ... 843
ly:stencil-expr ... 843
ly:stencil-extent ... 843
ly:stencil-in-color ... 843
ly:stencil-outline ... 843
ly:stencil-rotate ... 844
ly:stencil-rotate-absolute ... 844
ly:stencil-scale ... 844
ly:stencil-stack ... 844
ly:stencil-translate ... 844
ly:stencil-translate-axis ... 844
ly:stencil? ... 843
ly:stream-event? ... 844
ly:string-percent-encode ... 844
ly:string-substitute ... 844
ly:system-font-load ... 844
ly:text-interface::interpret-markup ... 844
ly:transform->list ... 845
ly:transform? ... 844
ly:translate-cpp-warning-scheme ... 845
ly:translator-context ... 845
ly:translator-description ... 845
ly:translator-group? ... 845
ly:translator-name ... 845
ly:translator? ... 845
ly:transpose-key-alist ... 845
ly:truncate-list! ... 845
ly:ttf->pfa ... 845
ly:ttf-ps-name ... 845
ly:type1->pfa ... 845
ly:undead? ... 845
ly:unit ... 845
ly:unpure-call ... 845
ly:unpure-pure-container-pure-part ... 845
ly:unpure-pure-container-unpure-part ... 846
ly:unpure-pure-container? ... 845
ly:usage ... 846
ly:verbose-output? ... 846
ly:version ... 846
ly:warning ... 846
ly:warning-located ... 846
ly:wide-char->utf-8 ... 846
\lydian ... 22
*LyricCombineMusic* ... 275, 281
*LyricExtender* ... 279
*LyricHyphen* ... 279
\lyricmode ... 268, 269, 630
*Lyrics* ... 209, 271, 275, 281, 308, 778
\lyrics ... 630
\lyricsto ... 269, 271, 272
*LyricText* ... 269, 306, 318

# M


m ... 421
maggiore ... 22
magnification->font-size ... 222, 552
\magnify ... 248, 709
magnifying text ... 709
\magnifyMusic ... 222, 811
\magnifyStaff ... 553, 811
magstep ... 222, 552, 632
maj ... 421
\major ... 22
majorSevenSymbol ... 427, 430
*makam* ... 479
makam ... 478
*makamlar* ... 473, 479
makamlar ... 478
make-dynamic-script ... 133
make-pango-font-tree ... 264
\makeClusters ... 175, 811
\makeDefaultStringTuning ... 811
Manuali ... 1
\map-markup-commands ... 761
*maqam* ... 476
maqam ... 473
maracas ... 766
marcato ... 124
\marcato ... 124, 764
margine di rilegatura ... 546
margine, testo che va oltre ... 240
\mark ... 114, 242, 811
*Mark_engraver* ... 116, 245
Mark_engraver ... 244
\markalphabet ... 756
*MarkEvent* ... 116, 245
\markLengthOff ... 73, 243
\markLengthOn ... 73, 243
\markletter ... 757
markup ... 247
\markup ... 242, 245, 247, 630
markup condizionale ... 497
*Markup construction in Scheme* ... 134, 135
markup multilinea ... 253
markup, allineare ... 251
markup, centrare sulla pagina ... 253
markup, decorazione ... 255
markup, due colonne ... 246
markup, espressione ... 247
markup, incorniciatura ... 255
markup, multipagina ... 260
markup, nel testo ... 269
markup, notazione musicale dentro ... 257
markup, partitura dentro ... 259
markup, sintassi ... 247
markup, testo a capo ... 254
markup, testo giustificato ... 254
markup, testo, comando di allineamento ... 254
markup, testo, dentro le legature di portamento ... 137
markup, testo, padding ... 255
markup-markup-spacing ... 544
markup-system-spacing ... 544
\markuplist ... 245, 260
\markupMap ... 811
martellato ... 363
max-systems-per-page ... 547
*maxima* ... 48, 61
\maxima ... 46, 59
Measure_grouping_engraver ... 94
measureLength ... 88, 122
measurePosition ... 75, 122
Medicaea, Editio ... 441
\medium ... 710
*melisma* ... 276, 279
melisma ... 276, 279
\melisma ... 276
melisma, con travature ... 86
\melismaEnd ... 276
melodia alternativa, passare a ... 302
melodia, mostrare i ritmi ... 83
*mensural notation* ... 441, 442, 445, 446, 447, 448, 449
mensurale ... 441
mensurale, chiave ... 445, 702
mensurale, coda ... 448
mensurale, formattazione ... 198
mensurali, legature bianche ... 450
*MensuralStaff* ... 195
MensuralStaff ... 194, 444
MensuralVoice ... 444
Mensurstriche ... 466
\mergeDifferentlyDottedOff ... 179
\mergeDifferentlyDottedOn ... 179
\mergeDifferentlyHeadedOff ... 179
\mergeDifferentlyHeadedOn ... 179
merging text ... 718, 726
metadati MIDI ... 498
metadati PDF ... 498
*meter* ... 80
*Metodi di modifica* ... 53, 619, 621
*metronome* ... 75
*metronome mark* ... 75
*MetronomeMark* ... 75
metronomo, segno, sotto il rigo ... 73
metronomo, segno, testo personalizzato ... 74
mezzosoprano, chiave di ... 17, 702
\mf ... 126
microtono ... 9
microtono, intavolatura ... 367
\midi ... 484, 592
*MIDI* ... 528, 531
MIDI ... 27, 525
MIDI, bilanciamento stereo ... 535
MIDI, blocco ... 526
MIDI, canali ... 533
MIDI, definizioni di contesto ... 531
MIDI, dinamiche ... 527
MIDI, dinamiche personalizzate ... 527
MIDI, effetti ... 535
MIDI, equalizzazione ... 527
MIDI, espressione ... 535
MIDI, livello del chorus ... 535
MIDI, metadati ... 498
MIDI, modificare l'equalizzazione predefinita di uno strumento ... 530
MIDI, notazione non supportata ... 526
MIDI, notazione supportata ... 525
MIDI, panning, posizione ... 535
MIDI, riverbero ... 535
MIDI, strumento ... 536
MIDI, tracce ... 533

MIDI, trasposizione ... 27
MIDI, un canale per voce ... 534
MIDI, uso delle ripetizioni ... 532
MIDI, volume ... 527
`midiBalance` ... 535
`midiChannelMapping` ... 533
`midiChorusLevel` ... 535
`midiDrumPitches` ... 403
`midiExpression` ... 535
`midiPanPosition` ... 535
`midiReverbLevel` ... 535
`min-systems-per-page` ... 547
`minimum-Y-extent` ... 561
`minimumFret` ... 351, 390
`minimumPageTurnLength` ... 560
`minimumRepeatLengthForPageTurn` ... 560
`\minor` ... 22
`minorChordModifier` ... 429
minore ... 22
mirroring markup ... 739
misolidio ... 22
misura ... 67
misura parziale ... 75
misura parziale, negli inni ... 329
misura, numero ... 108
misura, numero, stile del ... 110
misura, raggruppamento ... 94
misura, ripetizione ... 164
misura, sottoraggruppamento ... 94
misura, stanghetta manuale ... 101
misura, stanghette ... 101
`mixed` ... 340
`\mixolydian` ... 22
`\mm` ... 632
modale, inversione ... 16
modale, trasposizione ... 15
`\modalInversion` ... 16, 811
modalità accordo ... 419
modalità markup, caratteri speciali ... 247
modalità markup, testo tra virgolette ... 247
`\modalTranspose` ... 15, 811
`mode` ... 771
*Modelli integrati* ... 307
*Modelli per gruppi vocali* ... 283, 285, 307, 308, 325, 328
*Modelli per quartetto d'archi* ... 344
modello di musica araba ... 477
`modern` ... 30
*modern*, stile delle alterazioni ... 30, 31
`modern-cautionary` ... 31
*modern-cautionary*, stile delle alterazioni ... 30, 31
`modern-voice` ... 31
`modern-voice-cautionary` ... 31
*modern-voice-cautionary*, stile delle alterazioni ... 31
moderntab, chiave ... 366
modifica automatica del rigo ... 335
*Modifica dell'output* ... 592, 646
modifica della dimensione del segno di flageolet ... 406
modifica di una proprietà ... 615
modifica manuale del rigo ... 333
modifica, ripristinare ... 618
modificare il nome di uno strumento ... 212
modificare in un solo istante ... 619
*Modificare le proprietà di contesto* ... 607
modificare un abbellimento ... 118, 119
modificatore, nell'accordio ... 420
modo ... 22
modo ecclesiastico ... 22
`\mordent` ... 124, 764
mordente ... 124
mordente inferiore ... 124
mordente inverso ... 124
mordente inverso lungo ... 124
mordente inverso, giù ... 124
mordente inverso, sù ... 124
mordente superiore ... 124
movimenti, molteplici ... 481
`\mp` ... 126
*multi-measure rest* ... 67
multi-measure rest, within text, by `log` and `dot-count` ... 742
multi-measure rest, within text, by string ... 743
multilinea, markup ... 253
multilinea, testo ... 253
*MultiMeasureRest* ... 67
*MultiMeasureRestNumber* ... 67
*MultiMeasureRestScript* ... 67
`MultiMeasureRestScript` ... 64
*MultiMeasureRestText* ... 67
`MultiMeasureRestText` ... 64
multipagina, testo ... 260
multipla, legatura di portamento ... 135
*Music classes* ... 216
*Music functions* ... 660
musica a quattro battute ... 556
musica antica, chiave per ... 445
musica antica, chiavi di ... 702
musica araba ... 473
musica ficta ... 450
musica gregoriana, trascrizione moderna ... 324
musica in tempo libero ... 76, 122
musica in tempo libero, alterazioni ... 76
musica in tempo libero, interruzioni di linea ... 77
musica in tempo libero, interruzioni di pagina ... 77
musica in tempo libero, numeri di battuta ... 76
musica in tempo libero, stanghette ... 76
musica in tempo libero, travature ... 76
musica mensurale, segno ... 446
musica mensurale, trascrizione di ... 198
musica ottomana ... 478
musica parallela ... 191
musica per pianoforte, centrare le dinamiche ... 332
musica per principianti ... 40
musica polifonica ... 179
musica relativa, e `autochange` ... 336
musica religiosa ... 318
musica rinascimentale ... 198
musica turca ... 478
musica, dentro il blocco markup ... 257
musicale, suggerimento ... 313
`\musicglyph` ... 115, 741
`\musicMap` ... 811
musicologia, analisi ... 236
`musicQuotes` ... 771
mutable ... 770
mute bongo ... 766
mute conga ... 766
mute timbale ... 766

# N


N.C. (No Chord), simbolo ..... 425
`\name` ..... 607
nascondere alterazioni, su note legate all'inizio del sistema ..... 7
nascondere i righi ..... 208
nascondere i righi antichi ..... 208
nascondere i righi ritmici ..... 208
nascondere i tetragrammi ..... 208
nascondere la linea di estensione delle dinamiche ..... 132
nascosta, nota ..... 229
`\natural` ..... 742
naturali, armonici ..... 345
`neo-modern` ..... 33
*neo-modern*, stile delle alterazioni ..... 33
`neo-modern-cautionary` ..... 33
*neo-modern-cautionary*, stile delle alterazioni ..... 33
`neo-modern-voice` ..... 33
*neo-modern-voice*, stile delle alterazioni ..... 33
`neo-modern-voice-cautionary` ..... 33
*neo-modern-voice-cautionary*, stile delle alterazioni ..... 33
neomensurale ..... 442
nera, chiave mensurale ..... 445
nessun accordo, simbolo ..... 425
neumi quadrati, legatura gregoriana ..... 456
`\new` ..... 594
*New markup list command definition* ..... 260
*New_fingering_engraver* ..... 228, 612
`\newSpacingSection` ..... 579
niente, al, forcella ..... 130
`no-reset` ..... 34
`no-reset`, stile delle alterazioni ..... 34
`\noBeam` ..... 97
`\noBreak` ..... 554
nome accordo, insieme ai diagrammi dei tasti ..... 379
nome alternativo di un accordo ..... 427
nome altezza, altre lingue ..... 8
nome del cantante ..... 300
nome del personaggio ..... 311
nome delle note in turco ..... 478
nome di un accordo ..... 419, 424
nome di un'altezza ..... 1
nome nota, altre lingue ..... 8
nome nota, olandese ..... 6
nome nota, predefinito ..... 6
nome strumento ..... 210, 536
nome strumento, abbreviato ..... 210
nome strumento, aggiunta in altri contesti ..... 212
nome strumento, centrare ..... 210
nome strumento, complesso ..... 210
nome strumento, modifica ..... 212
nomi delle note in arabo ..... 474
*NonMusicalPaperColumn* ..... 579
`nonstaff-nonstaff-spacing` ..... 561
`nonstaff-relatedstaff-spacing` ..... 561
`nonstaff-unrelatedstaff-spacing` ..... 561
`\noPageBreak` ..... 557, 811
`\noPageTurn` ..... 560, 811
`\normal-size-sub` ..... 710
`\normal-size-super` ..... 250, 710
`\normal-text` ..... 711
`\normalsize` ..... 222, 251, 711
nota a piè di pagina ..... 499
nota a piè di pagina basata su un evento ..... 500
nota a piè di pagina basate sul tempo ..... 502
nota a piè di pagina nel testo separato ..... 505
nota a piè di pagina, nelle espressioni musicali ..... 499
nota che attraversa i righi ..... 338
nota colorata ..... 229
nota colorata in un accordo ..... 231
nota di basso, per accordi ..... 423
nota doppiamente puntata ..... 47
nota fantasma ..... 231
nota fantasma, percussioni ..... 405
nota in corpo più piccolo ..... 213, 217
nota invisibile ..... 229
nota muta, percussioni ..... 405
nota nascosta ..... 229
nota più piccola ..... 217
nota puntata ..... 47
nota puntata, cambiare il numero di punti ..... 48
nota puntata, spostamento orizzontale ..... 183
nota smorzata su strumento con tasti ..... 393
nota spaziatrice ..... 61
nota tra parentesi ..... 231
nota trasparente ..... 229
nota trasversale ai righi ..... 338
nota trasversale al rigo ..... 333
nota, altezza predefinita ..... 47
nota, divisione ..... 81
nota, durata ..... 46
nota, durata predefinita ..... 47
nota, lunghezza ..... 46
nota, nome predefinito ..... 6
nota, spaziatura orizzontale ..... 579
nota, spostamento ..... 179
nota, spostamento orizzontale ..... 184
nota, trasposizione di ..... 11
notazione grafica ..... 256
notazione semplificata ..... 40
notazione semplificata, usando numeri ..... 41
notazione, dentro il blocco markup ..... 257
notazione, dimensione del tipo di carattere ..... 222
notazione, spiegare la ..... 233
`\note` ..... 742
note a forma variabile ..... 42
*note head* ..... 448, 463
*Note simultanee* ..... 184
note simultanee e alterazioni ..... 35
*note value* ..... 48
note, within text, by duration ..... 742
note, within text, by `log` and `dot-count` ..... 742
`\note-by-number` ..... 742
*note-collision-interface* ..... 791, 794, 796
*note-event* ..... 40, 42, 45
`note-event` ..... 216
*note-head-interface* ..... 40, 42, 45
*Note_head_line_engraver* ..... 338
*Note_heads_engraver* ..... 40, 42, 45, 82, 610
`Note_heads_engraver` ..... 81
*Note_spacing_engraver* ..... 229
*NoteCollision* ..... 184
*NoteColumn* ..... 184
*NoteHead* ..... 40, 42, 45
`\notemode` ..... 630
*NoteSpacing* ..... 229, 578, 579
notina ..... 213, 217

notina, chiave ... 18
notina, formattazione ... 217
notina, togliere ... 221
\null ... 252, 757
NullVoice ... 297
\number ... 711
numeri di battuta, alternativi, nelle ripetizioni ... 159
numeri di battuta, con lettere, nelle ripetizioni ... 159
numeri di battuta, controlli ... 113
numeri di strofa ... 298
\numericTimeSignature ... 68
numero del gruppo irregolare, modifiche del ... 50
numero del gruppo irregolare, non predefinito ... 51
numero della misura, e ripetizioni ... 160
numero di battuta ... 108, 122
numero di battuta, a intervalli regolari ... 109
numero di battuta, allineamento ... 112
numero di battuta, cadenze ... 76
numero di battuta, collisione ... 113
numero di battuta, disposizione a distanza regolare ... 108
numero di battuta, musica in tempo libero ... 76
numero di battuta, nella prima misura ... 109
numero di battuta, stile ... 110
numero di battuta, togliere ... 112
numero di chiamata, sotto il rigo ... 73
numero di corda ... 344, 348
numero di corda vs. numero di diteggiatura ... 348
numero di misura ... 108
numero di misura, stile ... 110
numero di pagina, in numeri romani ... 549
numero di pagina, indicare il primo ... 549
numero di pagina, numerazione automatica ... 549
numero di pagina, sopprimere ... 549
numero di ripetizione, modificare ... 160
numero, in notazione semplificata ... 41
nuova spaziatura nel corso di un brano ... 579
nuovo contesto ... 594
nuovo rigo ... 194
nuvoletta ... 233
nuvoletta di aiuto ... 233

## O

*octavation* ... 27
\octaveCheck ... 10, 811
offset ... 622
\offset ... 622, 811
*Oggetti e interfacce* ... 507, 769, 770
*Oggetti interni al rigo* ... 631
oggetti variabili ... 770
oggetto colorato ... 229
oggetto grafico ... 769
oggetto grafico incorporato ... 255
oggetto grafico, disegnare ... 255
oggetto grafico, includere ... 255
oggetto immutabile ... 770
oggetto Scheme ... 771
oggetto, colorare ... 639
oggetto, rotazione ... 645
oggetto, sovrascrivere ... 639
oggetto, visibilità del ... 638
\omit ... 638, 811
\on-the-fly ... 497, 757
\once ... 617, 619, 623, 657, 812
\oneVoice ... 175
\open ... 124, 344, 765
open bongo ... 766
open conga ... 766
open high hat ... 766
open timbale ... 766
operazione, inversione ... 14
operazione, inversione modale ... 16
operazione, modale ... 15
operazione, retrogradazione ... 15
operazione, trasposizione ... 15
*Optical spacing* ... 578
*Opzioni avanzate della linea di comando per LilyPond* ... 524
*Opzioni di base della linea di comando per LilyPond* ... 523
oratorio ... 307
orchestra, archi ... 344
ordine di stampa ... 639
ordine verticale degli script ... 125
*Organizzare i brani con le variabili* ... 193, 313, 513, 518, 519, 597
organo, segno del pedale ... 124
orientamento dei diagrammi dei tasti, modifica ... 375
orientamento delle diteggiature ... 227
\oriscus ... 456, 462
orizzontale, allineamento, testo vocale ... 287
orizzontale, spaziatura ... 577
ornamento ... 116
ornamento, gruppo ritardato ... 125
*ossia* ... 208
ossia ... 204, 209
ossia, posizionamento ... 207
ottava ... 24
\ottava ... 24, 812
ottava assoluta ... 1
ottava relativa ... 2
ottava relativa, e accordi ... 5
ottava relativa, e trasposizione ... 6
ottava, controllo ... 10
ottava, estensore, modifica dell'inclinazione ... 26
ottava, per una sola voce ... 25
ottava, testo ... 25
*ottava-bracket-interface* ... 27
*Ottava_spanner_engraver* ... 27
*OttavaBracket* ... 27
ottavazione ... 24
ottomana, musica ... 478
outer-margin ... 546
output, definizione ... 592
output-count ... 771
output-def ... 770
output-suffix ... 771
outside-staff-horizontal-padding ... 576
outside-staff-padding ... 576
outside-staff-priority ... 576
\oval ... 736
\overlay ... 726
\override ... 617, 621, 757
\override-lines ... 761
*OverrideProperty* ... 615
\overrideProperty ... 622, 812
\overrideTimeSignatureSettings ... 68, 812
overriding property within text markup ... 757

\overtie ..... 711
overtie-ing text ..... 711

# P

\p ..... 126
\pad-around ..... 255, 726
\pad-markup ..... 255, 727
\pad-to-box ..... 255, 727
\pad-x ..... 255, 727
padding ..... 614
padding intorno al testo ..... 255
padding text ..... 727
padding text horizontally ..... 727
page breaking, manual ..... 557
page-breaking ..... 548
page-breaking-system-system-spacing ..... 548
page-count ..... 548
\page-link ..... 757
page-number-type ..... 549
\page-ref ..... 509, 758
page-spacing-weight ..... 549
\pageBreak ..... 557, 812
\pageTurn ..... 560, 812
pagina, formato ..... 540
pagina, interruzione ..... 582
pagina, interruzione, gestione con una voce apposita ..... 556
pagina, orientamento ..... 541
\palmMute ..... 812
\palmMuteOn ..... 812
Pango ..... 261
panning, posizione nel MIDI ..... 535
\paper ..... 484, 540
paper-height ..... 542
paper-width ..... 545
parallela, musica ..... 191
\parallelMusic ..... 191, 812
parentesi ..... 236
parentesi del gruppo irregolare, visibilità ..... 52
parentesi dell'arpeggio trasversale ai righi ..... 339
parentesi dell'arpeggio, contrassegnare voci divise ..... 309
parentesi della volta ..... 160
parentesi della volta, accorciare ..... 157
parentesi della volta, con testo ..... 162
parentesi della volta, e legatura di valore ..... 56
parentesi della volta, in altri righi ..... 158
parentesi di analisi musicologica ..... 236
parentesi di analisi, con etichetta ..... 237
parentesi di raggruppamento delle note ..... 236
parentesi graffa, annidamento di ..... 198
parentesi orizzontale ..... 236
parentesi quadra ..... 231
parentesi quadra verticale ..... 195
parentesi quadra, all'inizio di un gruppo di righi ..... 197
parentesi quadra, annidamento di ..... 198
parentesi uncinata (o angolare) ..... 170
parentesi, all'inizio di un rigo singolo ..... 197
parentesi, fraseggio ..... 236
parentesi, stile nell'arpeggio attraverso il rigo ..... 150
parentesi, verticale, contrassegnare voci divise ..... 309
*parentheses-interface* ..... 232
*ParenthesesItem* ..... 232
*Parenthesis_engraver* ..... 232
\parenthesize ..... 231, 737, 812
parlato ..... 317
parlato, testa di nota ..... 39
Parmesan, font ..... 679
parser ..... 770
parser variable ..... 771
*part* ..... 190
\partCombine ..... 185, 297, 813
\partCombine e testo cantato ..... 297
\partCombine e testo vocale ..... 188
\partCombineApart ..... 187
\partCombineAutomatic ..... 187
\partCombineChords ..... 187
\partCombineDown ..... 813
\partCombineForce ..... 813
partCombineListener ..... 771
*PartCombineMusic* ..... 190
\partCombineSoloI ..... 187
\partCombineSoloII ..... 187
\partCombineUnisono ..... 187
\partCombineUp ..... 813
parte a due ..... 185
parte solista ..... 185
\partial ..... 75, 153, 155, 813
partitura polimetrica ..... 601
partitura senza i righi vuoti ..... 208
*Partitura vocale a quattro parti SATB* ..... 307, 308
partitura vocale, aggiungere suggerimenti ..... 315
partitura, dentro il blocco markup ..... 259
*Partiture e parti* ..... 513
parziale, misura ..... 75
parziale, misura, negli inni ..... 329
\path ..... 737
path, drawing ..... 737
\pattern ..... 758
pausa ..... 59
pausa antica ..... 448
pausa d'intero ..... 59, 62
pausa di breve ..... 59
pausa di longa ..... 59
pausa di maxima ..... 59
pausa ecclesiastica ..... 64
pausa intera per una misura intera ..... 62
pausa invisibile ..... 61
pausa multipla ..... 59, 62
pausa multipla con testo a margine ..... 64
pausa multipla, attaccare fermata ..... 64
pausa multipla, attaccare testo ..... 64
pausa multipla, contrazione ..... 63
pausa multipla, e diteggiature ..... 67
pausa multipla, espansione ..... 63
pausa multipla, numerazione ..... 64
pausa multipla, posizionamento ..... 65
pausa multipla, script ..... 64
pausa multipla, testo a margine ..... 66
pausa spaziatrice ..... 61
pausa, collisioni di ..... 67
pausa, divisione ..... 81
pausa, inserire le durate ..... 59
pausa, segno ..... 139
pausa, specificare la posizione verticale ..... 59
pausa, spostamento automatico ..... 179
pausa, stile ..... 60

pause, condensare ... 67
PDF, metadati ... 498
pedal high hat ... 766
pedale del pianoforte ... 340
pedale dell'arpa ... 342
pedale di risonanza ... 340
pedale di risonanza, stile del ... 340
pedale sostenuto ... 340
pedale tonale ... 340
pedale, indicazione, misto ... 340
pedale, indicazione, parentesi ... 340
pedale, indicazione, testo ... 340
pedale, segno ... 124
`pedalSustainStyle` ... 340
pedice ... 250
`percent` ... 164
*percent repeat* ... 166
*Percent_repeat_engraver* ... 166
*PercentRepeat* ... 166
*PercentRepeatCounter* ... 166
*PercentRepeatedMusic* ... 166
percentuale, ripetizione ... 164
*Percussion* ... 398, 399, 400, 402, 405
percussioni ... 398, 400
percussioni personalizzate ... 402
percussioni, chiave ... 399
percussioni, chiavi di ... 702
percussioni, dead note ... 405
percussioni, ghost note ... 405
percussioni, nota fantasma ... 405
percussioni, nota morta ... 405
percussioni, nota muta ... 405
percussioni, nota non suonata ... 405
percussioni, varie ... 766
personaggio, nome ... 311
personalizzazione del nome di un accordo ... 427
`\pes` ... 462
Petrucci ... 441, 442
Petrucci, chiave ... 445, 702
*PhrasingSlur* ... 139
`\phrasingSlurDashed` ... 138
`\phrasingSlurDashPattern` ... 139, 813
`\phrasingSlurDotted` ... 138
`\phrasingSlurDown` ... 138
`\phrasingSlurHalfDashed` ... 138
`\phrasingSlurHalfSolid` ... 138
`\phrasingSlurNeutral` ... 138
`\phrasingSlurSolid` ... 138
`\phrasingSlurUp` ... 138
`\phrygian` ... 22
piè di pagina ... 486
`piano` ... 32
*piano*, stile delle alterazioni ... 32
`piano-cautionary` ... 32
*piano-cautionary*, stile delle alterazioni ... 32
*Piano_pedal_engraver* ... 341
pianoforte, musica, centrare le dinamiche ... 332
pianoforte, pedale ... 340
pianoforte, rigo ... 332
pianoforte, rigo per ... 195
*PianoPedalBracket* ... 341
*PianoStaff* ... 35, 149, 198, 213, 308, 333
`PianoStaff` ... 332, 335
piatti, vari ... 766
piatto China ... 766
piatto crash ... 766
piatto ride ... 766
piatto splash ... 766
*Pitch names* ... 2, 6, 8, 10, 449
*Pitch_squash_engraver* ... 45, 85, 610, 782
`Pitch_squash_engraver` ... 83
`\pitchedTrill` ... 151, 813
*Pitches* ... 2, 6, 8, 10, 14, 22, 24, 27, 28, 35, 38, 40, 42, 45, 446, 476
*Pitches, Altezze* ... 11
`pitchnames` ... 771
pizzicato a schiocco ... 346
pizzicato alla Bartók ... 346
placing horizontal brackets, around text ... 736
placing parentheses, around text ... 737
placing vertical brackets, around text ... 732
`\pointAndClickOff` ... 813
`\pointAndClickOn` ... 813
`\pointAndClickTypes` ... 813
polifonia su un rigo singolo ... 175
polifonia, in intavolatura ... 357
polifonia, testo cantato condiviso ... 297
polifonia, voci ulteriori ... 182
polimetria ... 601
polimetria, indicazione ... 78
pollice ... 764
pollice, indicazione ... 227
pollice, segno del ... 124
*polymetric* ... 53, 80
*polymetric time signature* ... 80
*polyphony* ... 184
portamenti indeterminati verso il basso (cadute) e verso l'alto ... 141
*portato* ... 126
portato ... 124
`\portato` ... 124, 764
*Posizionamento degli oggetti* ... 124, 126, 240
posizionamento della parentesi quadra del gruppo irregolare ... 49
posizionamento verticale delle dinamiche ... 128
posizionamento, diteggiatura mano destra ... 392
posizionamento, testo vocale ... 281
posizione, alterazione del basso continuo ... 436
posizione, ossia ... 207
posizione, pausa multipla ... 65
posizione, testo vocale ... 207
postscript ... 256
`\postscript` ... 256, 738
*power chord* ... 396
power chord ... 395
`\pp` ... 126
`\ppp` ... 126
`\pppp` ... 126
`\ppppp` ... 126
`\prall` ... 124, 764
`\pralldown` ... 124, 764
`\prallmordent` ... 124, 764
`\prallprall` ... 124, 764
`\prallup` ... 124, 764
`predefinedDiagramTable` ... 387
`\predefinedFretboardsOff` ... 389
`\predefinedFretboardsOn` ... 389
predefinita, altezza ... 47
predefinito, offset ... 622
prima volta ... 153

principianti, musica ... 40
print-all-headers ... 549
print-first-page-number ... 549
print-page-number ... 549
prob ... 771
\property-recursive ... 758
\propertyOverride ... 813
\propertyRevert ... 813
*PropertySet* ... 615
\propertySet ... 813
\propertyTweak ... 814
\propertyUnset ... 814
proprietà ... 615
proprietà condivisa ... 770
*Proprietà degli oggetti di formattazione* ... 769
proprietà dei grob ... 617
proprietà dell'oggetto ... 771
proprietà di contesto predefinita, modifica ... 602
proprietà immutabile ... 770
*Proprietà presenti nelle interfacce* ... 770
proprietà variabili ... 770
\pt ... 632
pull off ... 363
puntata, nota, spostamento orizzontale ... 183
punte delle travature ... 98
punteggiatura, nel testo vocale ... 268
punti di controllo, modifica con \tweak ... 621
punto di aumentazione, cambiare numero ... 48
punto di controllo, curva Bézier ... 652
punto, nota ... 47
\pushToTag ... 518, 814
\put-adjacent ... 727
putting space around text ... 727

# Q

q, ripetizione accordo ... 171, 352
*quarter tone* ... 8
quarto di tono ... 6
quarto di tono, intavolatura ... 367
\quilisma ... 456, 462
quotedCueEventTypes ... 216
quotedEventTypes ... 216
\quoteDuring ... 213, 217, 814
*QuoteMusic* ... 216

# R

r ... 59
R ... 62
ragged-bottom ... 542
ragged-last ... 546, 582
ragged-last-bottom ... 542
ragged-right ... 546, 582
raggruppamento di gruppi irregolari ... 49
\raise ... 252, 728
\raiseNote ... 814
raising text ... 728
*rast* ... 476
Ratisbona, Editio ... 441
\reduceChords ... 83, 814
referencing page label, in text ... 761
referencing page number, in text ... 757, 758
registro della fisarmonica ... 341
*RehearsalMark* ... 116, 245
relativa, ottava ... 2
\relative ... 2, 6, 14, 336, 814
*RelativeOctaveCheck* ... 11
*RelativeOctaveMusic* ... 6
relativo ... 2
\remove ... 600
remove-grace-property ... 119
\RemoveAllEmptyStaves ... 208, 818
\RemoveEmptyStaves ... 208, 818
\removeWithTag ... 515, 814
*repeat* ... 160
\repeat ... 153
\repeat percent ... 164
\repeat tremolo ... 167
repeatCommands ... 160
repeatCountVisibility ... 166
*RepeatedMusic* ... 160, 162, 164
*Repeats* ... 160, 162, 164, 166, 169
*RepeatSlash* ... 166
*RepeatSlashEvent* ... 166
\repeatTie ... 56, 155, 293
\replace ... 712
\resetRelativeOctave ... 5, 814
respiro, segno ... 139
respiro, simbolo, modifica ... 140
\responsum ... 457
*Rest* ... 61
\rest ... 59, 743
rest, within text, by log and dot-count ... 742
rest, within text, by string ... 743
\rest-by-number ... 742
rest-event ... 216
*Rest_engraver* ... 82
*RestCollision* ... 184
restNumberThreshold ... 64
restrainOpenStrings ... 351
retrogradazione, trasformazione ... 15
\retrograde ... 15, 814
\reverseturn ... 124, 764
\revert ... 618
*RevertProperty* ... 615
\revertTimeSignatureSettings ... 69, 814
\rfz ... 126
rgb, colore ... 230
rgb-color ... 230
\rheel ... 124, 765
*Rhythmic_column_engraver* ... 610
*RhythmicStaff* ... 45, 85, 195
RhythmicStaff ... 194
*Rhythms* .. 49, 53, 54, 59, 61, 62, 67, 71, 76, 77, 80, 82, 85, 88, 96, 100, 101, 108, 113, 114, 116, 120, 122, 123
ricopiate, ripetizioni ... 162
ride bell ... 766
ridimensionamento dei righi ... 204
righi, gruppo di ... 195
\right-align ... 251, 728
right-aligning text ... 728
\right-brace ... 758
\right-column ... 728
right-margin ... 545
\rightHandFinger ... 391, 814
rigo del pianoforte ... 332
rigo Gregoriano per trascrizione ... 194
rigo multiplo ... 195

rigo per batteria ... 194
rigo per coro ... 195
rigo per percussioni ... 194
rigo per pianoforte ... 195
rigo ritmico ... 194
rigo temporaneo ... 204, 208
rigo vuoto ... 208
rigo, annidato ... 198
rigo, batteria ... 194
rigo, cambio automatico ... 335
rigo, cambio del, linea ... 337
rigo, cambio manuale ... 333
rigo, distanza ... 561
rigo, impostare la dimensione ... 552
rigo, nascondere ... 208
rigo, nuovo ... 194
rigo, percussioni ... 194
rigo, ridimensionamento del ... 204
rigo, segno di metronomo sotto ... 73
rigo, simbolo del ... 201
rigo, singolo ... 194
rigo, singolo, con parentesi ... 197
rigo, strumenti a tastiera ... 332
rigo, strumenti dotati di tasti ... 332
rilegatura ... 546
rimozione, negli accordi ... 423
rinascimentale, musica ... 198
ripetere il testo vocale, con finali alternativi ... 291
ripetere una legatura di valore ... 56
ripetizione alternata ... 162
ripetizione annidata ... 160
ripetizione con percentuale ... 164
ripetizione con volte, sotto gli accordi ... 430
ripetizione della misura ... 164
ripetizione e glissando ... 145
ripetizione normale ... 153
ripetizione ricopiata ... 162
ripetizione, alternativa ... 162
ripetizione, ambigua ... 160
ripetizione, breve ... 164
ripetizione, con anacrusi ... 155
ripetizione, con controlli di battuta ... 155
ripetizione, con finali alternativi ... 153
ripetizione, con legature di valore ... 155
ripetizione, con segno ... 156
ripetizione, contatore con segno percentuale ... 165
ripetizione, doppia, stile per la volta ... 158
ripetizione, e glissandi ... 160
ripetizione, e legatura di portamento ... 160
ripetizione, e numero della misura ... 160
ripetizione, e testo vocale ... 287
ripetizione, fine ... 160
ripetizione, informazioni sul tempo ... 160
ripetizione, inizio ... 160
ripetizione, manuale ... 160
ripetizione, numeri di battuta alternativi ... 159
ripetizione, numeri di battuta con lettere ... 159
ripetizione, percentuale, visibilità del conto ... 166
ripetizione, tremolo ... 167
ripetizione, unfold ... 162
ripetizione, usando `q` ... 352
ripetizione, uso di `q` ... 171
ripetizioni con segni percentuali isolati ... 166
ripetizioni nel MIDI ... 532
ripetuti, accordi, soppressione ... 426
ripristinare le proprietà predefinite di un'indicazione di tempo ... 69
ripristinare una modifica ... 618
riquadro intorno all'oggetto ... 632
risonanza, pedale di ... 340
ritmo di accompagnamento, mostrare ... 83
ritmo, mostrare la melodia ... 83
ritocco ... 619
ritornello ... 104
riverbero nel MIDI ... 535
`\roman` ... 712
`\romanStringNumbers` ... 344, 348
romboidale, testa di nota ... 345
`\rotate` ... 729
rotating text ... 729
rotazione di un oggetto ... 645
`\rounded-box` ... 255, 739
`\rtoe` ... 124, 765
rullante ... 766
rullante acustico ... 766
rullante elettrico ... 766

# S

`s` ... 61
Sacred Harp, testa di nota ... 42
`\sacredHarpHeads` ... 42
`\sacredHarpHeadsMinor` ... 43
*Salmi* ... 328
saltare le note nel teso vocale ... 292
`\sans` ... 712
SATB ... 307
scalable vector graphics, output ... 523
scalare la durata ... 53
`\scale` ... 739
`\scaleDurations` ... 54, 78, 814
scaling markup ... 739
scaling text ... 729
scambio di rigo ... 337
scelta della dimensione del tipo di carattere (notazione) ... 222
*Scheme tutorial* ... 592
Scheme, contenitore impuro ... 658
Scheme, contenitore puro ... 658
Scheme, oggetto ... 771
schiocco, pizzicato ... 346
*scordatura* ... 24
*Score* ... 123, 772, 776
`\score` ... 480, 484, 743
`\score-lines` ... 761
`score-markup-spacing` ... 544
`score-system-spacing` ... 544
`scoreTitleMarkup` ... 494
*Script* ... 124, 126, 455
script ... 124
script su pausa multipla ... 64
script, ordine verticale ... 125
*Script_engraver* ... 455
*ScriptEvent* ... 455
scritta ... 239
scrivere la musica in parallelo ... 191
seconda volta ... 153
segni di tempo, per glissando ... 144
segni per la conduzione ... 94
segno ... 103, 115, 124

\segno ..... 124, 766
segno del pedale ..... 124
segno del pedale dell'organo ..... 124
segno del pollice ..... 124
segno di bequadro ..... 6
segno di bequadro, supplementare, impedire ..... 7
segno di chiamata ..... 114
segno di chiamata manuale ..... 115
segno di chiamata personalizzato ..... 115
segno di chiamata, formato ..... 115
segno di chiamata, stile ..... 115
segno di coda, variante ..... 124
segno di metronomo con testo personalizzato ..... 74
segno di modifica dell'ottava ..... 2
segno di spunta ..... 140
segno di tremolo ..... 168
segno separatore del sistema ..... 200
segno testuale, allineamento verticale ..... 131
segno, a fine linea ..... 244
segno, con ripetizioni ..... 156
segno, di chiamata, formato ..... 115
segno, di chiamata, stile ..... 115
segno, numero di chiamata, sotto il rigo ..... 73
segno, su ogni rigo ..... 244
segno, sulla stanghetta ..... 242
segui voce ..... 337
*self-alignment-interface* ..... 613, 646
self-alignment-X ..... 561
*semai* ..... 477
semai, forma ..... 476
semi-bemolle ..... 6, 9
semi-diesis ..... 6, 9
semibemolle, simbolo, aspetto ..... 474
\semicirculus ..... 454, 766
\semiflat ..... 744
\semiGermanChords ..... 427
\semisharp ..... 745
*Sento le Voci* ..... 179, 400, 402
separatore dell'accordo, modifica ..... 431
sesqui-bemolle ..... 9
sesqui-diesis ..... 9
\sesquiflat ..... 745
\sesquisharp ..... 745
\set ..... 88, 615, 621
set-global-fonts ..... 265
set-global-staff-size ..... 552
set-octavation ..... 24
settima maggiore, accordo, formattazione ..... 430
setting extent of text object ..... 760
setting horizontal text alignment ..... 721
setting subscript, in standard font size ..... 710
setting superscript, in standard font size ..... 710
\settingsFrom ..... 814
\sf ..... 126
\sff ..... 126
\sfz ..... 126
\shape ..... 653, 815
*sharp* ..... 8
\sharp ..... 745
\shiftDurations ..... 815
\shiftOff ..... 179
\shiftOn ..... 179
\shiftOnn ..... 179
\shiftOnnn ..... 179
short-indent ..... 211, 547
\shortfermata ..... 124, 765
show-available-fonts ..... 264
showFirstLength ..... 522, 771
\showKeySignature ..... 409
showLastLength ..... 522, 771
\showStaffSwitch ..... 337
*side-position-interface* ..... 613, 646
sidestick ..... 766
\signumcongruentiae ..... 124, 766
*sikah* ..... 476
sillabe, durata automatica ..... 271
simboli di settima maggiore ..... 429
simbolo arabo per semibemolle ..... 474
simbolo del rigo ..... 201
simbolo del rigo, impostazione del ..... 633
simbolo di respiro, modifica ..... 140
simbolo non musicale ..... 256
simbolo speciale di arpeggio ..... 147
*simile* ..... 166
\simple ..... 713
simple text string ..... 713
simple text string, with tie characters ..... 745
simultanea, legatura di portamento ..... 135
*Simultaneous notes* ..... 171, 175, 179, 184, 190, 193
sincronizzazione degli abbellimenti ..... 120
\single ..... 503, 624, 815
sintassi di LilyPond ..... 770
sintassi di markup ..... 247
sistema ..... 195
sistema, delimitatore di inizio annidato ..... 198
sistema, segno separatore ..... 200
\skip ..... 61
\skip ..... 61, 292, 815
*SkipMusic* ..... 62
skipTypesetting ..... 522
*Slash_repeat_engraver* ..... 166
slashChordSeparator ..... 428
slashed digit ..... 758
\slashed-digit ..... 758
\slashedGrace ..... 116, 815
\slashSeparator ..... 549
slide, in intavolatura ..... 361
*slur* ..... 138
*Slur* ..... 138, 149
slur-event ..... 216
\slurDashed ..... 135
\slurDashPattern ..... 136, 815
\slurDotted ..... 135
\slurDown ..... 135
\slurHalfDashed ..... 136
\slurHalfSolid ..... 136
\slurNeutral ..... 135
\slurSolid ..... 135
\slurUp ..... 136
\small ..... 222, 251, 713
\smallCaps ..... 713
\smaller ..... 248, 251, 713
smob ..... 771
\snappizzicato ..... 124, 765
Sol tenore, chiave di ..... 702
Sol, chiave di ..... 17, 702
Solesmes ..... 441
soppressione degli accordi ripetuti ..... 426
sopprimere accordi ripetuti ..... 386
soprano, chiave di ..... 17, 702

sos. ... 340
sostenuto, pedale ... 340
*SostenutoEvent* ... 341
`\sostenutoOff` ... 340
`\sostenutoOn` ... 340
*SostenutoPedal* ... 341
*SostenutoPedalLineSpanner* ... 341
`\sourcefileline` ... 485
`\sourcefilename` ... 485
Southern Harmony, testa di nota ... 42
`\southernHarmonyHeads` ... 42
`\southernHarmonyHeadsMinor` ... 43
sovrascrivere un oggetto ... 639
`\sp` ... 126
*Spacing* .. 542, 544, 547, 550, 552, 554, 557, 558, 559, 561, 565, 567, 568, 576, 577, 579, 580, 582, 588, 589, 591
`spacing` ... 578
*spacing-spanner-interface* ... 801, 802
*SpacingSpanner* ... 577, 578, 579, 580
`\spacingTweaks` ... 815
`Span_stem_engraver` ... 338
*SpanBar* ... 108
spaziatura del testo vocale ... 285
spaziatura nuova nel corso di un brano ... 579
spaziatura orizzontale ... 577
spaziatura orizzontale, sovrascrivere la ... 658
*Spaziatura ottica* ... 579
spaziatura rigorosa e abbellimenti ... 119
spaziatura verticale ... 561, 582
spaziatura, visualizzazione della formattazione ... 589
spazio bianco ... 485
spazio, dentro i sistemi ... 561
spazio, nel testo vocale ... 268, 275
spazio, tra i righi ... 561
spezzato, arpeggio ... 147
spostamento automatico di una pausa ... 179
spostamento orizzontale di una nota ... 184
spostamento relativo (offset) ... 622
*Spostare gli oggetti* ... 251, 255
spostare una nota ... 179
spostare una voce ... 179
`\spp` ... 126
Sprechgesang ... 317
spunta, segno di ... 140
staccatissimo ... 124
`\staccatissimo` ... 124, 764
*staccato* ... 126
staccato ... 124
`\staccato` ... 124, 764
stacking text in a column ... 717
*Staff* .... 35, 38, 81, 195, 198, 209, 213, 238, 578, 772
*staff* ... 195, 204, 208
*Staff notation* .. 75, 195, 198, 200, 201, 204, 208, 209, 213, 216, 222
`staff-affinity` ... 561
`\staff-space` ... 632
`staff-staff-spacing` ... 561
*staff-symbol-interface* ... 204
`Staff.midiInstrument` ... 536
`Staff_collecting_engraver` ... 244
*Staff_symbol_engraver* ... 209
`Staff_symbol_engraver` ... 208
*StaffGroup* ... 113, 198, 200
`staffgroup-staff-spacing` ... 561
*StaffGrouper* ... 309, 563, 564, 567, 627
*StaffSpacing* ... 579
*StaffSymbol* ... 195, 204, 208
StaffSymbol ... 201
stampa del nome di un accordo ... 424
stampare carattere riservato ... 247
stampare carattere speciale ... 247
stampo ... 771
stampo, soppressione ... 638
stanghetta ... 101
stanghetta di chiusura ... 101
stanghetta doppia ... 101
stanghetta invisibile ... 101
stanghetta manuale ... 101
stanghetta predefinita, modifica ... 107
stanghetta, cadenze ... 76
stanghetta, definire ... 105
stanghetta, evitata dal testo vocale ... 287
stanghetta, forcella, terminare a ... 129
stanghetta, in `ChordNames` ... 430
stanghetta, musica in tempo libero ... 76
stanghetta, simboli sulla ... 242
stanghetta, soppressione ... 644
stanghetta, tra i righi ... 198
*StanzaNumber* ... 306
`start-repeat` ... 160
`startAcciaccaturaMusic` ... 119
`startAppoggiaturaMusic` ... 119
`startGraceMusic` ... 119
`\startGroup` ... 236
`\startStaff` ... 201, 204
`\startTrillSpan` ... 150
`\stdBass` ... 750
`\stdBassIV` ... 751
`\stdBassV` ... 752
`\stdBassVI` ... 753
*Stem* ... 233, 340, 623
*stem-interface* ... 233
`stem-spacing-correction` ... 578
*Stem_engraver* ... 100, 233
`\stemDown` ... 232
`stemLeftBeamCount` ... 98
`\stemNeutral` ... 232
`stemRightBeamCount` ... 98
`\stemUp` ... 232
stencil ... 771
`\stencil` ... 759
`stencil`, soppressione ... 638
stereo, bilanciamento nel MIDI ... 535
stile alternativo delle note brevi ... 48
stile del pedale di risonanza ... 340
stile del segno di chiamata ... 115
stile delle alterazioni *default* ... 30
stile delle alterazioni *modern* ... 30, 31
stile delle alterazioni *modern-cautionary* ... 30
stile delle alterazioni *neo-modern* ... 33
stile delle alterazioni *neo-modern-cautionary* ... 33
stile delle alterazioni per insegnamento (*teaching*) ... 34
stile delle alterazioni *voice* ... 30
stile delle alterazioni, choral ... 32
stile delle alterazioni, choral-cautionary ... 32
stile delle alterazioni, *dodecaphonic* ... 34
stile delle alterazioni, dodecaphonic-first ... 34
stile delle alterazioni, dodecaphonic-no-repeat ... 34
stile delle alterazioni, `forget` ... 35

stile delle alterazioni, *modern-voice-cautionary* .... 31
stile delle alterazioni, *neo-modern-voice* ..... 33
stile delle alterazioni, *neo-modern-voice-cautionary* ..... 33
stile delle alterazioni, `no-reset` ..... 34
stile delle alterazioni, *piano* ..... 32
stile delle alterazioni, *piano-cautionary* ..... 32
stile delle dinamiche del testo ..... 132
stile delle teste di nota ..... 701
stile di alterazione ..... 28
stile di alterazione predefinito ..... 28
stile di chiave ..... 445
stile di indicazione del pedale ..... 340
stile di voce ..... 179
stile moderno dell'alterazione ..... 30
stile, doppia ripetizione per la volta ..... 158
stile, legatura di portamento ..... 135
stile, pause ..... 60
stile, teste di nota ..... 39
`stopAcciaccaturaMusic` ..... 119
`stopAppoggiaturaMusic` ..... 119
`stopGraceMusic` ..... 119
`\stopGroup` ..... 236
`\stopped` ..... 124, 765
`\stopStaff` ..... 201, 204, 208
`\stopTrillSpan` ..... 150
`\storePredefinedDiagram` ..... 382, 387, 815
strappato ..... 363
strato ..... 639
`strictBeatBeaming` ..... 94
*StringNumber* ..... 350
`stringNumberOrientations` ..... 227
`\stringTuning` ..... 365, 815
`stringTunings` ..... 364, 378
strofa, numeri di ..... 298
*StrokeFinger* ..... 393
`strokeFingerOrientations` ..... 227, 392
`\stropha` ..... 456, 462
strumenti a tastiera, rigo ..... 332
strumenti dotati di tasti, rigo ..... 332
strumento a fiato ..... 406
strumento a tasti, diteggiatura della mano destra ..... 391
strumento aerofono ..... 406
strumento con tasti, accordature predefinite ..... 364
strumento con tasti, armonici ..... 393
strumento con tasti, forma di un accordo ..... 382
strumento con tasti, indicare la posizione e il barré ..... 393
strumento con tasti, nota smorzata ..... 393
strumento traspositore ..... 12, 27
strumento, nome ..... 210, 536
strumento, nome complesso ..... 210
`\strut` ..... 759
stumento, equalizzazione nel MIDI, cambiare predefinita ..... 530
`\styledNoteHeads` ..... 815
`\sub` ..... 250, 714
subbasso, chiave di ..... 17
`subdivideBeams` ..... 93
subscript text ..... 714
suddividere le travature ..... 93
suggerimento musicale ..... 313
suggerimento, nella partitura vocale ..... 315
`suggestAccidentals` ..... 125, 450
*Sul non annidamento di parentesi e legature di valore* ..... 138, 139
`\super` ..... 250, 714
superscript text ..... 714
`sus` ..... 423
*SustainEvent* ..... 341
`\sustainOff` ..... 340
`\sustainOn` ..... 340
*SustainPedal* ..... 341
*SustainPedalLineSpanner* ..... 341
SVG, output ..... 523
`system-count` ..... 547
`system-separator-markup` ..... 549
`system-system-spacing` ..... 544
`systems-per-page` ..... 547
*SystemStartBar* ..... 198, 200
*SystemStartBrace* ..... 198, 200
*SystemStartBracket* ..... 198, 200
*SystemStartSquare* ..... 198, 200

# T

*Tab_note_heads_engraver* ..... 368
`\tabChordRepeats` ..... 352, 815
`\tabChordRepetition` ..... 815
tabella alternativa, tastiera ..... 387
`\tabFullNotation` ..... 351
tablatura ..... 194, 347
tablatura e armonici ..... 355
tablatura per banjo ..... 347, 396
tablatura per chitarra ..... 347
tablatura per liuto ..... 397
tablatura personalizzata ..... 364
tablatura, accordature personalizzate ..... 365
tablatura, accordature predefinite ..... 364
tablatura, banjo ..... 364
tablatura, base ..... 350
tablatura, basso ..... 364
tablatura, basso elettrico ..... 364
tablatura, chiave ..... 366
tablatura, chiavi di ..... 702
tablatura, chitarra ..... 364
tablatura, contrabbasso ..... 364
tablatura, e armonici ..... 357
tablatura, e gambi ..... 356
tablatura, e polifonia ..... 357
tablatura, e slide ..... 361
tablatura, e travature ..... 356
tablatura, glissando di accordi ..... 362
tablatura, mandolino ..... 364
tablatura, martellato ..... 363
tablatura, microtono ..... 367
tablatura, predefinita ..... 350
tablatura, quarto di tono ..... 367
tablatura, strappato ..... 363
tablatura, ukulele ..... 364
tablatura, viola ..... 364
tablatura, violino ..... 364
tablatura, violoncello ..... 364
`\table` ..... 761
`\table-of-contents` ..... 511, 762
*TabNoteHead* ..... 364
*TabStaff* ..... 195, 364
`TabStaff` ..... 194, 350

*TabVoice* ... 364
`TabVoice` ... 350
tag ... 515
`\tag` ... 515, 815
`\tagGroup` ... 517, 815
tagliata, testa di nota ... 45
taglio addizionale ... 201
taglio addizionale, funzionamento interno ... 201
taglio addizionale, modificare ... 201
tam tam ... 766
tamburello ... 766
tante voci ... 179
`\taor` ... 409
*taqasim* ... 477
taqasim ... 476
tasti, diagramma personalizzato ... 375
tasti, diagramma personalizzato, aggiungere ... 381
tasti, diagramma, trasposizione ... 380
tasti, diagrammi automatici ... 388
tasti, strumento a, diteggiatura della mano destra ... 391
tastiera del diagramma dei tasti, personalizzata ... 383
tastiera, definire predefinita ... 384
tastiera, diagramma ... 368
tastiera, diagramma personalizzato ... 368
tastiera, tabella alternativa ... 387
tasto ... 351
tasto, strumento con, armonici ... 393
tasto, strumento con, nota smorzata ... 393
`teaching` ... 34
*teaching*, stile delle alterazioni ... 34
`\teeny` ... 222, 251, 714
tempo ... 67, 71
`\tempo` ... 71
tempo composto, indicazione ... 80
*tempo indication* ... 75
tempo polimetrico ... 78
tempo polimetrico, con travature ... 78
tempo, in una partitura ... 122
tempo, modifica, senza il segno di metronomo ... 74
tempo, stile ... 68
`\temporary` ... 623, 657, 816
tenere la musica etichettata ... 515
tenore, chiave di ... 17, 702
*tenuto* ... 126
tenuto ... 124
`\tenuto` ... 124, 764
terzina ... 49
terzina, formattazione della ... 50
testa di nota ... 222
testa di nota antica ... 447
testa di nota romboidale ... 345
testa di nota, Aiken ... 42
testa di nota, armonico ... 39
testa di nota, barrata ... 39
testa di nota, chitarra ... 39
testa di nota, Christian Harmony ... 42
testa di nota, esercizio ... 40
testa di nota, facile da suonare ... 40
testa di nota, forma ... 42
testa di nota, Funk ... 42
testa di nota, Harmonia Sacra ... 42
testa di nota, improvvisazione ... 45
testa di nota, notazione semplificata ... 40
testa di nota, parlato ... 39
testa di nota, romboidale ... 39
testa di nota, Sacred Harp ... 42
testa di nota, Southern Harmony ... 42
testa di nota, speciale ... 39
testa di nota, stile ... 701
testa di nota, stili ... 39
testa di nota, tagliata ... 45
testa di nota, Walker ... 42
teste di nota antiche ... 463
testo a capo automaticamente ... 254
testo a due colonne ... 246
testo a margine ... 247
testo a margine, pausa multipla ... 66
testo al livello superiore ... 245
testo cantato, condiviso tra voci ... 297
testo cantato, su abbellimenti ... 301
testo condizionale ... 497
testo dell'ottava ... 25
testo formattato su più pagine ... 260
testo giustificato ... 254
testo indipendente ... 245
testo multilinea ... 253
testo separato ... 245
testo su pausa multipla ... 64
testo tra virgolette ... 239
testo tra virgolette, in modalità markup ... 247
testo vocale alternato ... 295
testo vocale diviso ... 295
testo vocale e melodie ... 271
testo vocale, allineamento a una melodia ... 269
testo vocale, allineamento con una melodia sporadica ... 599
testo vocale, allineamento orizzontale ... 287
testo vocale, aumentare lo spazio tra ... 285
testo vocale, citazione ... 268
testo vocale, con travatura ... 88
testo vocale, e markup ... 269
testo vocale, e note legate ... 293
testo vocale, evitare le stanghette ... 287
testo vocale, formattazione ... 268
testo vocale, mantenere dentro il margine ... 287
testo vocale, posizionamento ... 207
testo vocale, punteggiatura ... 268
testo vocale, ripetizioni ... 287
testo vocale, ripetizioni con finali alternativi ... 291
testo vocale, saltare le note ... 292
testo vocale, spazi ... 268
testo vocale, spaziatura della versione 2.12 ... 283
testo vocale, tenerlo dentro il margine ... 240
testo vocale, uso delle variabili ... 279
testo, allineamento ... 251
testo, allineamento orizzontale ... 251
testo, allineamento verticale ... 252
testo, altre lingue ... 239
testo, assegnato a una voce ... 175
testo, centrare sulla pagina ... 253
testo, comando di allineamento ... 254
testo, decorazione ... 255
testo, dentro le legature di portamento ... 137
testo, dimensione ... 248
testo, elemento non vuoto ... 239
testo, esteso su più pagine ... 260
testo, fuori dal margine ... 240
testo, in colonne ... 246, 253
testo, incorniciatura ... 255

testo, indicazione ... 242
testo, inserimento ... 268
testo, nella parentesi della volta ... 162
testo, padding ... 255
testo, posizionamento ... 281
testo, \skip ... 61
testo, sulla stanghetta ... 242
testo, tenerlo dentro il margine ... 240
*Text* ... 240, 242, 245, 246, 248, 251, 255, 257, 259, 260, 264, 269
`text` ... 340, 714
text column, left-aligned ... 725
text column, right-aligned ... 728
*text-interface* ... 613, 757
*text-script-interface* ... 613
*Text_engraver* ... 610
`\textLengthOff` ... 64, 66, 240
`\textLengthOn` ... 64, 66, 131, 240
*TextScript* ... 126, 240, 246, 251, 255, 257, 259, 260, 419
*TextSpanner* ... 242, 638
`\textSpannerDown` ... 241
`\textSpannerNeutral` ... 241
`\textSpannerUp` ... 241
*The Emmentaler font* ... 741
`\thumb` ... 124, 227
*tie* ... 59, 82
*Tie* ... 59
`\tie` ... 715
tie-ing text ... 715
*TieColumn* ... 59, 656
`TieColumn` ... 58
`\tied-lyric` ... 745
`\tieDashed` ... 56
`\tieDashPattern` ... 56, 816
`\tieDotted` ... 56
`\tieDown` ... 56
`\tieHalfDashed` ... 56
`\tieHalfSolid` ... 56
`\tieNeutral` ... 56
`\tieSolid` ... 56
`\tieUp` ... 56
`tieWaitForNote` ... 58
timbales ... 766
`\time` ... 67, 88, 816
*time signature* ... 71
`\times` ... 816
*TimeScaledMusic* ... 53
*TimeSignature* ... 71, 81
`timeSignatureFraction` ... 78
*Timing_translator* ... 71, 76, 81, 108, 123, 776
`\tiny` ... 222, 251, 715
tipi di carattere disponibili, elenco ... 264
tipi di carattere, famiglia ... 250
tipo di carattere ... 261, 679, 769
tipo di carattere (notazione), dimensione standard ... 226
tipo di carattere, cambiare ... 248
tipo di carattere, dimensione ... 248
tipo di carattere, impostare la dimensione ... 552
tipo di carattere, modificarli per l'intero documento ... 264
tipo di carattere, ridimensionamento ... 222
tipo di carattere, trovare disponibilità ... 264
titolo ... 486
`\tocItem` ... 511, 816
`\tocItemWithDotsMarkup` ... 510
togliere i numeri di battuta ... 112
togliere la musica etichettata ... 515
togliere le citazioni in corpo più piccolo ... 221
tom ... 766
tom alto ... 766
tom basso ... 766
tom medio ... 766
tonale, pedale ... 340
*Top* ... 1, 592
`top-margin` ... 542
`top-markup-spacing` ... 544
`top-system-spacing` ... 544
`toplevel-bookparts` ... 771
`toplevel-scores` ... 771
tracce MIDI ... 533
`\translate` ... 252, 729
`\translate-scaled` ... 252, 729
translating text ... 729
*Translation* ... 612
`\transparent` ... 759
`\transpose` ... 6, 11, 14, 816
`\transposedCueDuring` ... 220, 816
*TransposedMusic* ... 14
*transposing instrument* ... 28, 311
`\transposition` ... 27, 213, 816
trascrizione di musica mensurale ... 198
trascrizione moderna, della musica gregoriana ... 324
trascrizione, da mensurale a moderno ... 470
trasformazione retrograda ... 15
trasparente, nota ... 229
trasparenti, rendere gli oggetti ... 639
trasporre ... 11
traspositore, strumento ... 12
trasposizione ... 11
trasposizione del diagramma dei tasti ... 380
trasposizione dell'ottava ... 18
trasposizione modale ... 15
trasposizione opzionale dell'ottava ... 18
trasposizione, altezze, "intelligente" ... 13
trasposizione, chiave ... 18
trasposizione, delle altezze ... 11
trasposizione, delle note ... 11
trasposizione, e ottava relativa ... 6
trasposizione, MIDI ... 27
trasposizione, strumento ... 27
trasversale ai righi, accordo ... 338
trasversale ai righi, gambo ... 338
trasversale ai righi, linea ... 337
trasversale ai righi, nota ... 338
trasversale ai righi, parentesi dell'arpeggio ... 339
trasversale, nota ... 333
trasversale, travatura ... 333
tratteggio, modelli, legatura di portamento ... 136, 137
tratti di suddivisione della travatura, direzione ... 94
trattino ... 279
travatura a raggiera ... 100
travatura angolare ... 86
travatura con angolazione ... 86
travatura con angolazione, modifica ... 86
travatura convergente o divergente ... 100
travatura del tremolo ... 167
travatura manuale ... 85
travatura manuale, abbellimenti ... 97

travatura manuale, scorciatoia per impostare la direzione ... 97
travatura trasversala al rigo ... 333
travatura, cadenze ... 76
travatura, con melisma ... 86
travatura, con tempi polimetrici ... 78
travatura, con testo vocale ... 88
travatura, estremità, in una partitura ... 95
travatura, estremità, in voci multiple ... 95
travatura, gruppo irregolare, interruzione di linea ... 53
travatura, in intavolatura ... 356
travatura, interruzioni di linea ... 86
travatura, musica in tempo libero ... 76
travatura, `\partCombine` con `\autoBeamOff` ... 87
travatura, personalizzazione delle regole ... 85
travatura, punta ... 98
travatura, suddivisione ... 93
travature manuali ... 97
tre corde ... 340
`\treCorde` ... 340
tremolo ... 167
`tremolo` ... 167
tremolo tra due righi ... 168
tremolo, segno ... 168
triade ... 420
`\triangle` ... 256, 739
triangolo ... 766
*trill* ... 152
`\trill` ... 124, 150, 764
trillo ... 124, 150
trillo con alterazione ... 151
trillo con altezza ... 151
trillo con notina ... 151
trillo con notina e alterazione ... 151
*TrillSpanner* ... 152, 638
*triplet* ... 53
trovare i tipi di carattere disponibili ... 264
*Tunable context properties* ... 278, 279, 617
*tuplet* ... 53
`\tuplet` ... 49, 78, 816
*TupletBracket* ... 53
`\tupletDown` ... 49
`\tupletNeutral` ... 49
*TupletNumber* ... 53
`TupletNumber` ... 50
`\tupletSpan` ... 50, 816
`tupletSpannerDuration` ... 50
`\tupletUp` ... 49
turca, musica ... 478
turco, nome delle note in ... 478
`\turn` ... 124, 764
`\tweak` ... 619, 622, 816
`\tweak`, modificare i punti di controllo ... 621
`tweak`, relazione con `\override` ... 622
*Tweaks and overrides* ... 646
`two-sided` ... 546
`\type` ... 607
`\typewriter` ... 715

# U

U.C. ... 340
ukulele, diagramma dei tasti ... 370
una corda ... 340
`\unaCorda` ... 340
*UnaCordaEvent* ... 341
*UnaCordaPedal* ... 341
*UnaCordaPedalLineSpanner* ... 341
*unbreakable-spanner-interface* ... 88
`\underline` ... 248, 715
underlining text ... 715
`\undertie` ... 716
undertie-ing text ... 716
`\undo` ... 624, 817
`unfold` ... 162
unfold, finali alternativi ... 162
unfold, ripetizione ... 162
*UnfoldedRepeatedMusic* ... 160, 164
`\unfoldRepeats` ... 532, 817
*Unfretted strings* ... 344
`\unHideNotes` ... 229
Unicode ... 519
unione delle parti ... 185
unire le note ... 179
`\unset` ... 616
`\upbow` ... 124, 344, 765
`\upmordent` ... 124, 764
`\upprall` ... 124, 764
`\upright` ... 716
*Uso da linea di comando* ... 522
UTF-8 ... 519

# V

varbaritono, chiave di ... 17
`\varcoda` ... 124, 766
variabile dell'analizzatore sintattico ... 771
`variabile globale` ... 771
variabile Scheme ... 771
variabili ... 485
variabili, oggetti e proprietà ... 770
variabili, uso delle ... 513
variante del segno di coda ... 124
Vaticana, Editio ... 441
*VaticanaStaff* ... 195
`VaticanaStaff` ... 194, 452
`VaticanaVoice` ... 452
`\vcenter` ... 729
`\verbatim-file` ... 759
`\version` ... 485
`\versus` ... 457
*VerticalAxisGroup* ... 209, 309, 563, 564, 565, 567, 568, 818
`VerticalAxisGroup` ... 561
verticale, spaziatura ... 561, 582
vertically centering text ... 729
`\verylongfermata` ... 124, 765
vibraslap ... 766
violino, chiave di ... 17, 702
`\virga` ... 456, 462
virgolette, nel testo vocale ... 275
`\virgula` ... 454
visibilità dell'indicazione di tempo ... 67
visibilità dell'oggetto ... 638

visibilità della chiave trasposta ..... 644
visibilità della parentesi del gruppo irregolare ..... 52
*Visibilità e colore degli oggetti* ..... 62, 209, 229, 325, 601, 638, 640, 644
*Vocal music* ..... 268, 306, 308, 311, 317, 318
vocale, partitura, aggiungere suggerimenti ..... 315
voce ..... 175
voce apposita per gestire le interruzioni ..... 556
voce, ambitus ..... 36
voce, citare ..... 213, 217
voce, con ottavazione ..... 25
voce, divisa ..... 309
voce, `\partCombine` con `\autoBeamOff` ..... 87
voce, seguire ..... 337
voce, spostamento ..... 179
voce, stile di ..... 179
voce, ulteriore, nella musica polifonica ..... 182
voci, multiple ..... 179
*Voice* ..... 38, 45, 190, 216, 222, 275, 578, 614
`voice` ..... 28, 30
`Voice` ..... 175
*voice*, stile delle alterazioni ..... 30
*VoiceFollower* ..... 338, 638
`\voiceFour` ..... 175
`\voiceFourStyle` ..... 179
`\voiceNeutralStyle` ..... 179
`\voiceOne` ..... 175
`\voiceOneStyle` ..... 179
`\voices` ..... 178, 817
`\voiceThree` ..... 175
`\voiceThreeStyle` ..... 179
`\voiceTwo` ..... 175
`\voiceTwoStyle` ..... 179
`\void` ..... 537, 817
*volta* ..... 160
volta ..... 153
volta della ripetizione, modificare ..... 160
volta, parentesi ..... 160
volta, parentesi accorciate ..... 157
volta, prima ..... 153
volta, seconda ..... 153
volta, stile della doppia ripetizione ..... 158
*Volta_engraver* ..... 427
`Volta_engraver` ..... 158
*VoltaBracket* ..... 160, 162
*VoltaRepeatedMusic* ..... 160, 162
volume MIDI ..... 527
`\vspace` ..... 730
vuoto, accordo ..... 119, 333

# W

Walker, testa di nota ..... 42
`\walkerHeads` ..... 42
`\walkerHeadsMinor` ..... 43
`whichBar` ..... 107
whistle ..... 766
`\whiteout` ..... 759
`\whiteTriangleMarkup` ..... 428
*Winds* ..... 407, 409, 411, 419
`\with` ..... 600, 605
`\with-color` ..... 229, 760
`\with-dimensions` ..... 760
`\with-dimensions-from` ..... 760
`\with-link` ..... 761
`\with-outline` ..... 761
`\with-url` ..... 740
`\withMusicProperty` ..... 817
woodblock ..... 766
`\woodwind-diagram` ..... 748
`\wordwrap` ..... 254, 731
`\wordwrap-field` ..... 730
`\wordwrap-internal` ..... 762
`\wordwrap-lines` ..... 260, 762
`\wordwrap-string` ..... 731
`\wordwrap-string-internal` ..... 763
*World music* ..... 473, 474, 476, 477

# X

x11, colore ..... 229
`x11-color` ..... 229, 231
`X-offset` ..... 561
`\xNote` ..... 39, 817
`\xNotesOff` ..... 39
`\xNotesOn` ..... 39