LilyPond

Il compositore tipografico per la musica

Learning Manual

Il team di sviluppo di LilyPond

Questo file fornisce un'introduzione alla versione di LilyPond 2.14.2.

Per maggiori informazioni su come questo manuale si integra col resto della documentazione, o per leggere questo manuale in altri formati, si veda Sezione "Manuali" in *Informazioni generali*.

Se ti manca qualche manuale, puoi trovare la completa documentazione all'indirizzo http://www.lilypond.org/.

Copyright © 1999–2011 by the authors.

La traduzione della seguente nota di copyright è gentilmente offerta per le 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.

> E' garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation; senza alcuna sezione non modificabile. Una copia della licenza è acclusa nella sezione intitolata "Licenza per Documentazione Libera GNU".
>
> 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 2.14.2

Sommario

1 Tutorial ... 1

1.1 Compilare un file ... 1
1.1.1 Inserire l'input ... 1
1.1.2 MacOS X ... 2
1.1.3 Windows ... 6
1.1.4 Linea di comando ... 13
1.2 Come scrivere i file di input ... 13
1.2.1 Notazione semplice ... 13
1.2.2 Lavorare sui file di input ... 18
1.3 Gestire gli errori ... 19
1.3.1 Consigli generali per la risoluzione dei problemi ... 19
1.3.2 Alcuni errori comuni ... 19
1.4 Come leggere i manuali ... 19
1.4.1 Materiale omesso ... 19
1.4.2 Esempi cliccabili ... 20
1.4.3 Panoramica dei manuali ... 20

2 Notazione comunemente utilizzata ... 21

2.1 Notazione su un solo pentagramma ... 21
2.1.1 Controlli di battuta ... 21
2.1.2 Alterazioni e armature di chiave ... 21
2.1.3 Legature di valore e di portamento ... 23
2.1.4 Articolazione e dinamica ... 24
2.1.5 Aggiungere il testo ... 25
2.1.6 Code automatiche e manuali ... 26
2.1.7 Comandi di tempo avanzati ... 26
2.2 Note simultanee ... 27
2.2.1 Espressioni musicali ... 27
2.2.2 Righi multipli ... 29
2.2.3 Gruppi di pentagrammi ... 30
2.2.4 Combinare le note negli accordi ... 31
2.2.5 Polifonia su un singolo rigo ... 31
2.3 Canzoni ... 32
2.3.1 Impostare canzoni semplici ... 32
2.3.2 Allineare il testo alla melodia ... 32
2.3.3 Testo su più righi ... 36
2.4 Ritocchi finali ... 37
2.4.1 Organizzare i brani con le variabili ... 37
2.4.2 Aggiungere i titoli ... 38
2.4.3 Nomi assoluti delle note ... 39
2.4.4 Dopo il tutorial ... 40

3 Concetti fondamentali ... 41

3.1 Come funzionano i file di input di LilyPond ... 41
3.1.1 Introduzione alla struttura di un file di LilyPond ... 41
3.1.2 La partitura è una (singola) espressione musicale composta ... 43
3.1.3 Annidare le espressioni musicali ... 45

3.1.4 Sul non annidamento di parentesi e legature di valore ... 47
3.2 Le voci contengono la musica ... 48
3.2.1 Sento le Voci ... 48
3.2.2 Definire esplicitamente le voci ... 52
3.2.3 Voci e musica vocale ... 56
3.3 Contesti e incisori ... 59
3.3.1 I contesti ... 59
3.3.2 Creare i contesti ... 60
3.3.3 Gli incisori ... 62
3.3.4 Modificare le proprietà di contesto ... 63
3.3.5 Aggiungere e togliere gli incisori ... 67
3.4 Estendere i modelli ... 70
3.4.1 Soprano e violoncello ... 70
3.4.2 Partitura vocale a quattro parti SATB ... 73
3.4.3 Scrivere una partitura da zero ... 78
3.4.4 Ridurre l'input grazie a variabili e funzioni ... 83
3.4.5 Partiture e parti ... 85

4 Tweaking output ... 87
4.1 Tweaking basics ... 87
4.1.1 Introduction to tweaks ... 87
4.1.2 Objects and interfaces ... 87
4.1.3 Naming conventions of objects and properties ... 88
4.1.4 Tweaking methods ... 88
4.2 The Internals Reference manual ... 91
4.2.1 Properties of layout objects ... 91
4.2.2 Properties found in interfaces ... 95
4.2.3 Types of properties ... 97
4.3 Appearance of objects ... 98
4.3.1 Visibility and color of objects ... 98
4.3.2 Size of objects ... 102
4.3.3 Length and thickness of objects ... 105
4.4 Placement of objects ... 106
4.4.1 Automatic behavior ... 106
4.4.2 Within-staff objects ... 107
Fingering ... 108
4.4.3 Outside-staff objects ... 110
4.5 Collisions of objects ... 116
4.5.1 Moving objects ... 116
4.5.2 Fixing overlapping notation ... 118
4.5.3 Real music example ... 123
4.6 Further tweaking ... 131
4.6.1 Other uses for tweaks ... 131
4.6.2 Using variables for tweaks ... 133
4.6.3 Style sheets ... 135
4.6.4 Other sources of information ... 139
4.6.5 Advanced tweaks with Scheme ... 140

Appendice A Modelli **142**
A.1 Rigo singolo 142
A.1.1 Solo note 142
A.1.2 Note e testo 142
A.1.3 Note e accordi 143
A.1.4 Note, testo e accordi. 144
A.2 Modelli per pianoforte 144
A.2.1 Solo pianoforte 144
A.2.2 Pianoforte e melodia con testo 145
A.2.3 Pianoforte con testo al centro 146
A.2.4 Pianoforte con dinamiche al centro 147
A.3 Quartetto d'archi 148
A.3.1 Quartetto d'archi 148
A.3.2 Parti di un quartetto d'archi 150
A.4 Gruppi vocali 152
A.4.1 Partitura vocale SATB 152
A.4.2 Partitura vocale SATB e automatica riduzione per pianoforte 154
A.4.3 SATB con contesti allineati 156
A.4.4 SATB su quattro righi 158
A.4.5 Strofa sola e ritornello a due parti 159
A.4.6 Inni 161
A.4.7 Salmi 163
A.5 Modelli per orchestra 166
A.5.1 Orchestra, coro e pianoforte 166
A.6 Modelli per notazione antica 169
A.6.1 Transcrizione di musica mensurale 169
A.6.2 Trascrizione di musica Gregoriana 175
A.7 Altri modelli 176
A.7.1 Combo jazz 176

Appendice B GNU Free Documentation License **183**

Appendice C Indice di LilyPond **190**

1 Tutorial

Questo capitolo fornisce un'introduzione di base all'uso di LilyPond.

1.1 Compilare un file
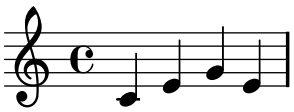
1.1.1 Inserire l'input

"Compilazione" è il termine usato per indicare l'elaborazione di un file di input in formato LilyPond per produrre uno o più file di output. I file di output generalmente sono PDF (per la stampa e la visualizzazione), MIDI (per la riproduzione audio) e PNG (per l'utilizzo online). I file di input di LilyPond sono semplici file di testo.

Questo esempio mostra un semplice file di input:

```
\version "2.14.2"
{
  c' e' g' e'
}
```

L'output grafico è:



> **Nota:** Le note e i testi nel file LilyPond devono essere sempre scritti tra `{` **parentesi graffe** `}`. Per evitare ambiguità, le parentesi dovrebbero essere delimitate da degli spazi, a meno che non si trovino all'inizio o alla fine di una linea. Può darsi che in alcuni esempi di questo manuale le parentesi verranno omesse, ma non dimenticarle nella tua musica! Per maggiori informazioni sull'aspetto degli esempi nel manuale, si veda Sezione 1.4 [Come leggere i manuali], pagina 19.

Inoltre, l'input di LilyPond è **sensibile alle maiuscole**. '`{ c d e }`' è un input valido; '`{ C D E }`' invece produrrà un messaggio di errore.

Generare l'output

Come generare l'output di LilyPond dipende dal tuo sistema operativo e dai programmi che usi.

- Sezione 1.1.2 [MacOS X], pagina 2 Sezione 1.1.2 [MacOS X], pagina 2 (grafico)
- Sezione 1.1.3 [Windows], pagina 6 Sezione 1.1.3 [Windows], pagina 6 (grafico)
- Sezione 1.1.4 [Linea di comando], pagina 13 Sezione 1.1.4 [Linea di comando], pagina 13 (linea di comando)

Si noti che sono disponibili molti altri editor di testo con un miglior supporto per LilyPond. Per maggiori informazioni, si veda Sezione "Editing facilitato" in *Informazioni generali*.

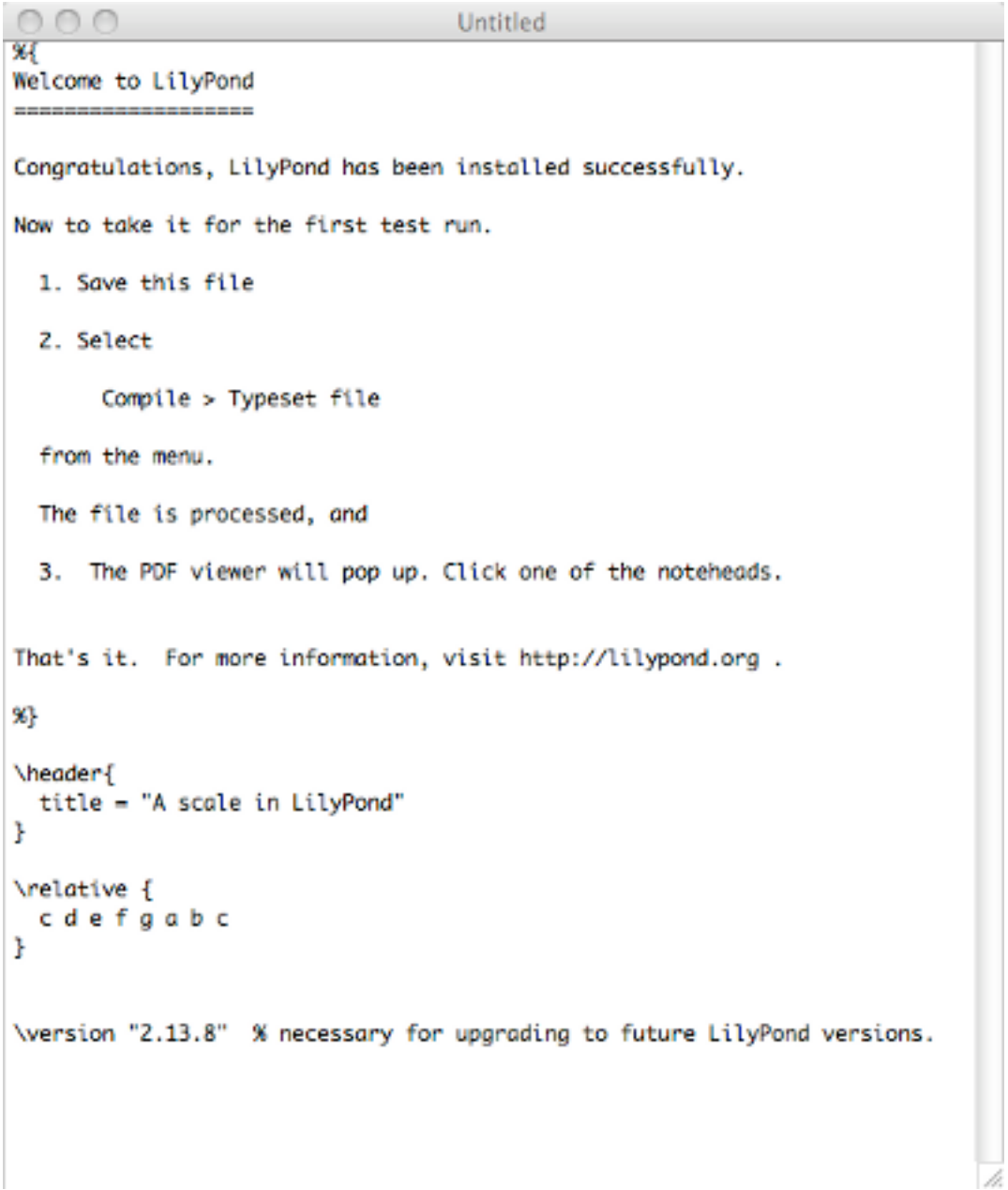
> **Nota:** La prima volta che si esegue LilyPond la compilazione potrà richiedere uno o due minuti perché tutti i font di sistema devono essere prima analizzati. In seguito, LilyPond sarà molto più veloce!

1.1.2 MacOS X

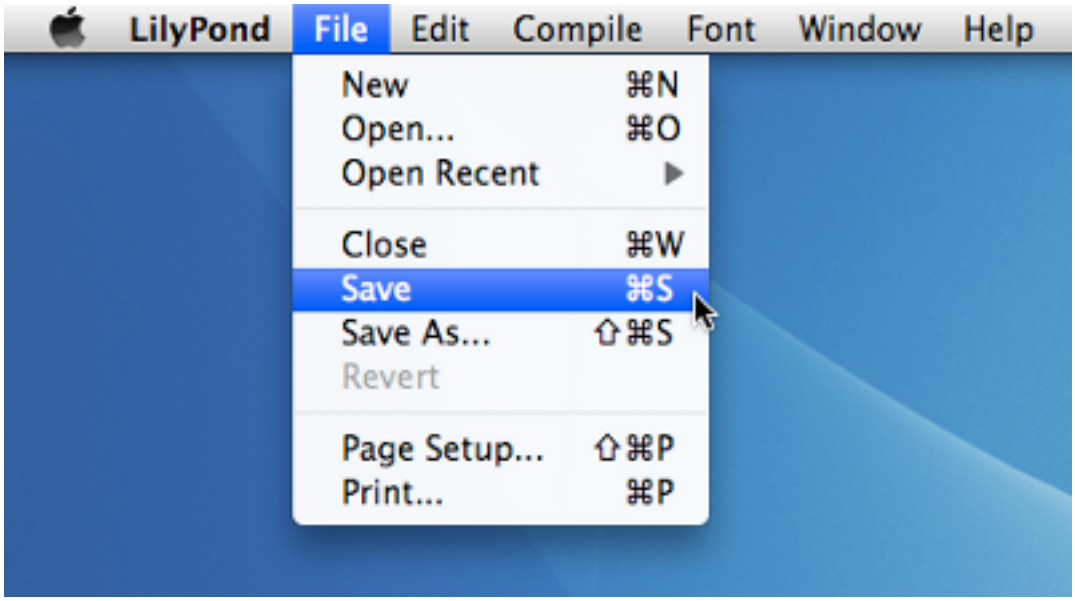
Nota: These instructions assume that you are using the LilyPond application. If you are using any of the programs described in Sezione "Easier editing" in *Informazioni generali*, please consult the documentation for those programs if you have any problems.

Step 1. Create your '`.ly`' file

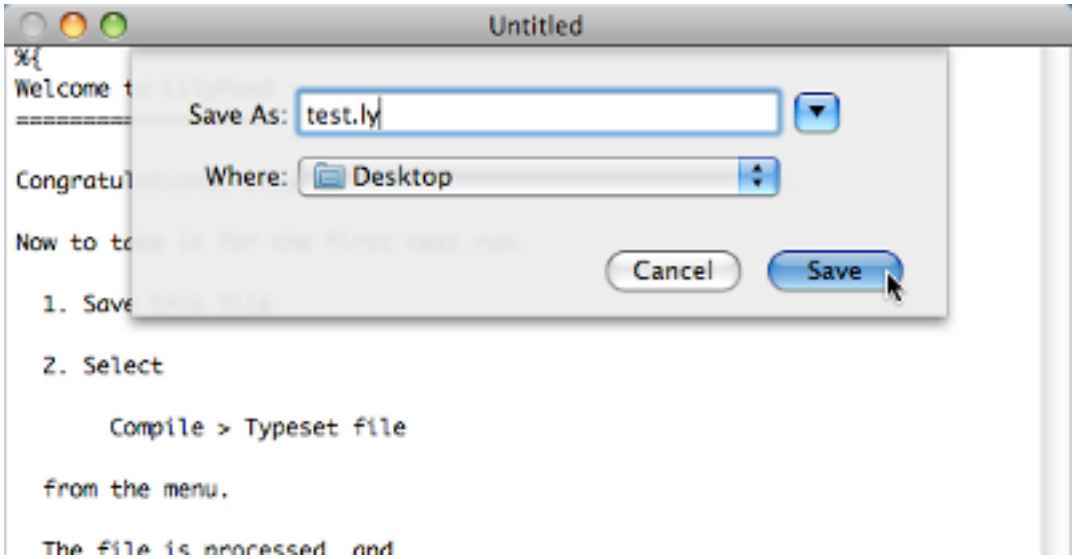
Double click the `LilyPond.app`, an example file will open.



From the menus along the top left of your screen, select `File > Save`.

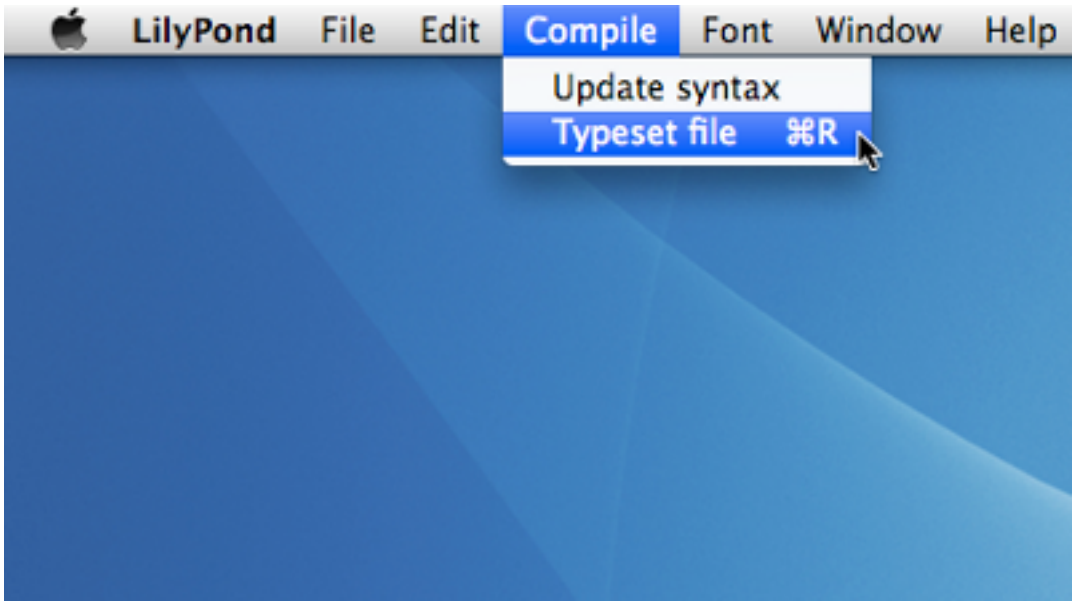


Choose a name for your file, for example '`test.ly`'.

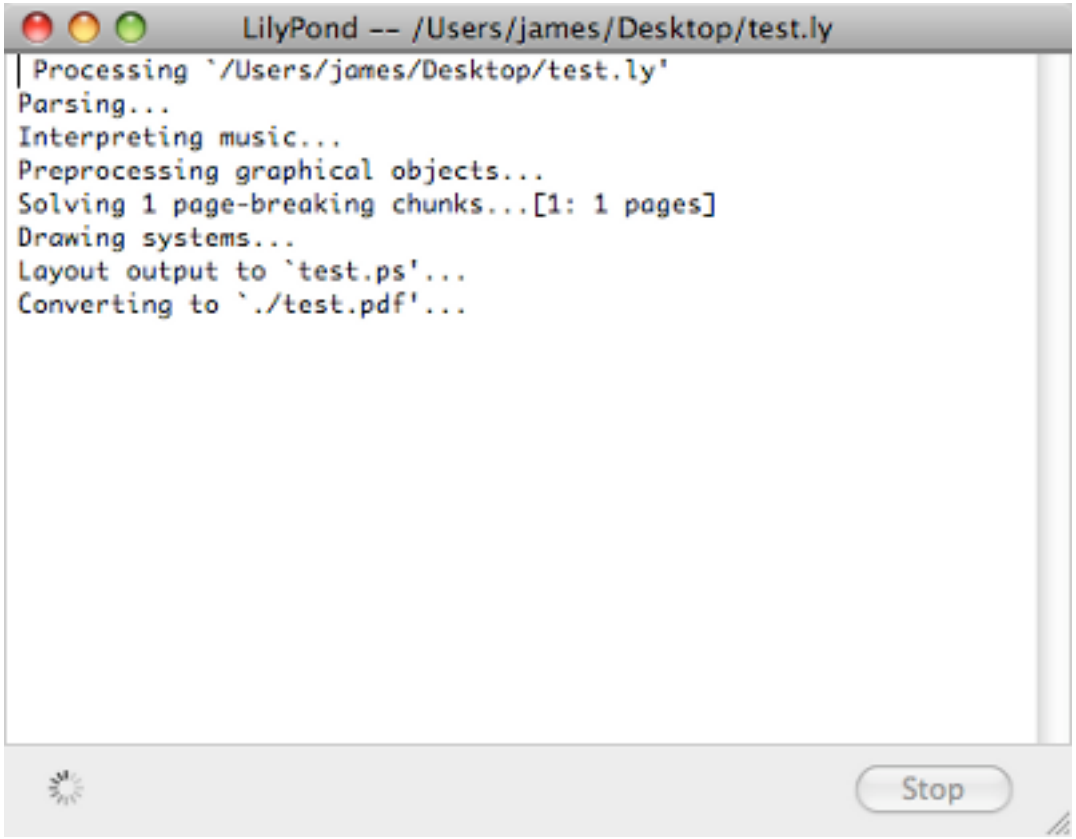


Step 2. Compile (with LilyPad)

From the same menus, select `Compile` > `Typeset`.

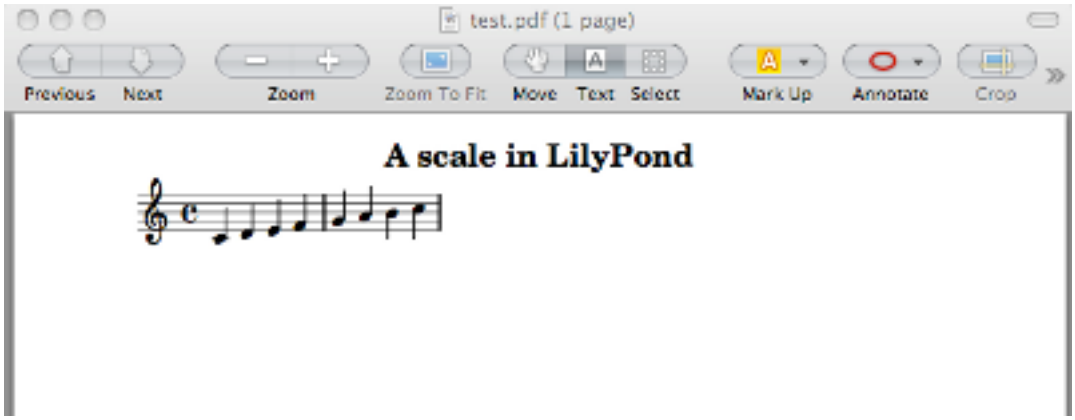


A new window will open showing a progress log of the compilation of the file you have just saved.



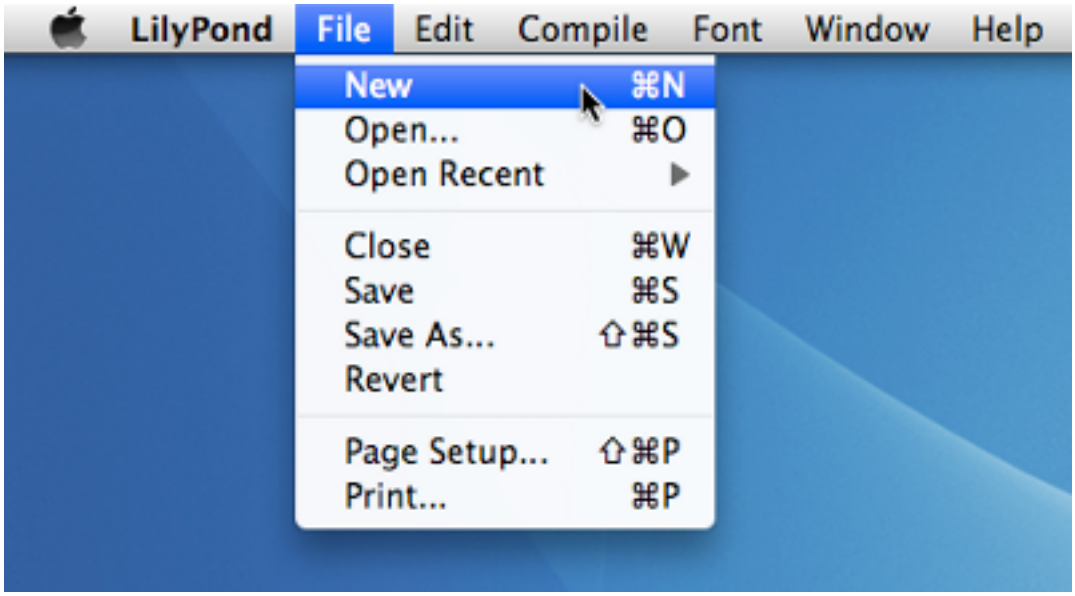
Step 3. View output

Once the compilation has finished, a PDF file will be created with the same name as the original file and will be automatically opened in the default PDF viewer and displayed on your screen.

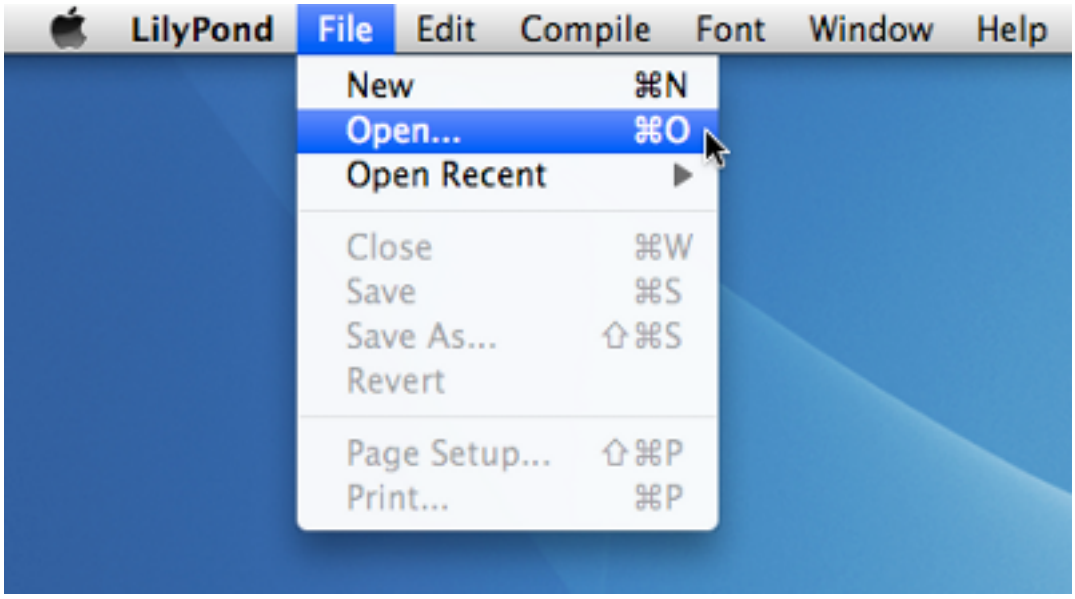


Other commands

To create new files for LilyPond, begin by selecting `File > New`



or `File > Open` to open and edit existing files you have saved previously.



You must save any new edits you make to your file before you `Compile > Typeset` and if the PDF file is not displayed check the window with the progress log for any errors.

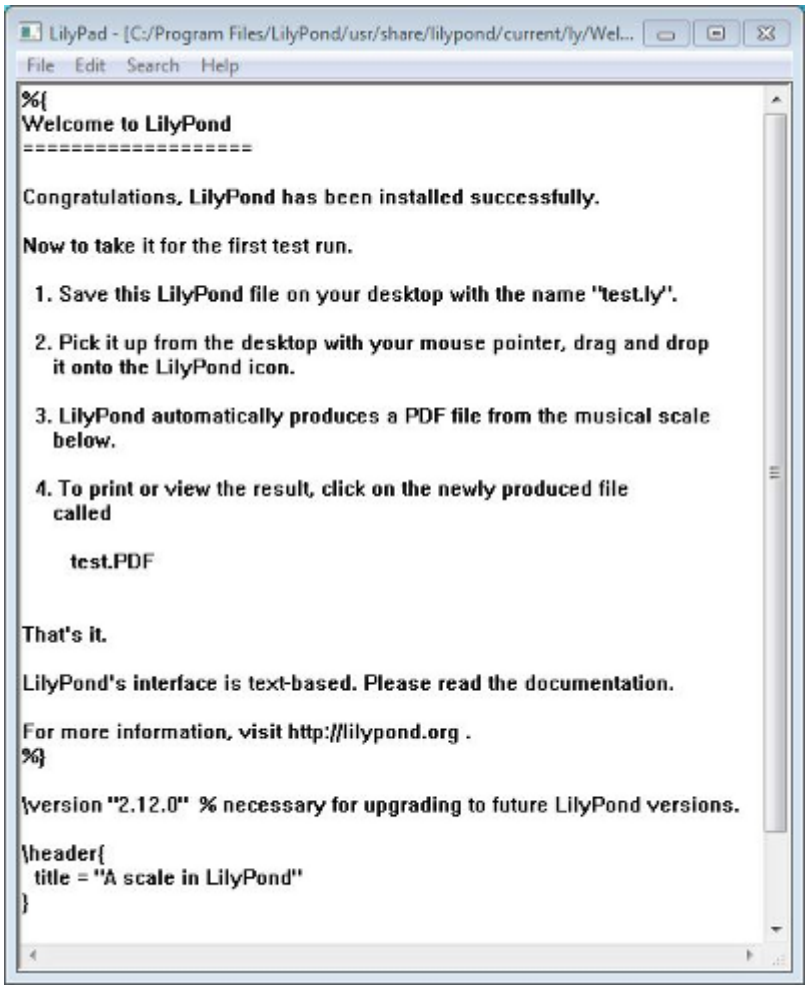
If you are not using the default Preview PDF viewer that comes with the Mac Operating system and you have the PDF file generated from a previous compilation open, then any further compilations may fail to generate an update PDF until you close the original.

1.1.3 Windows

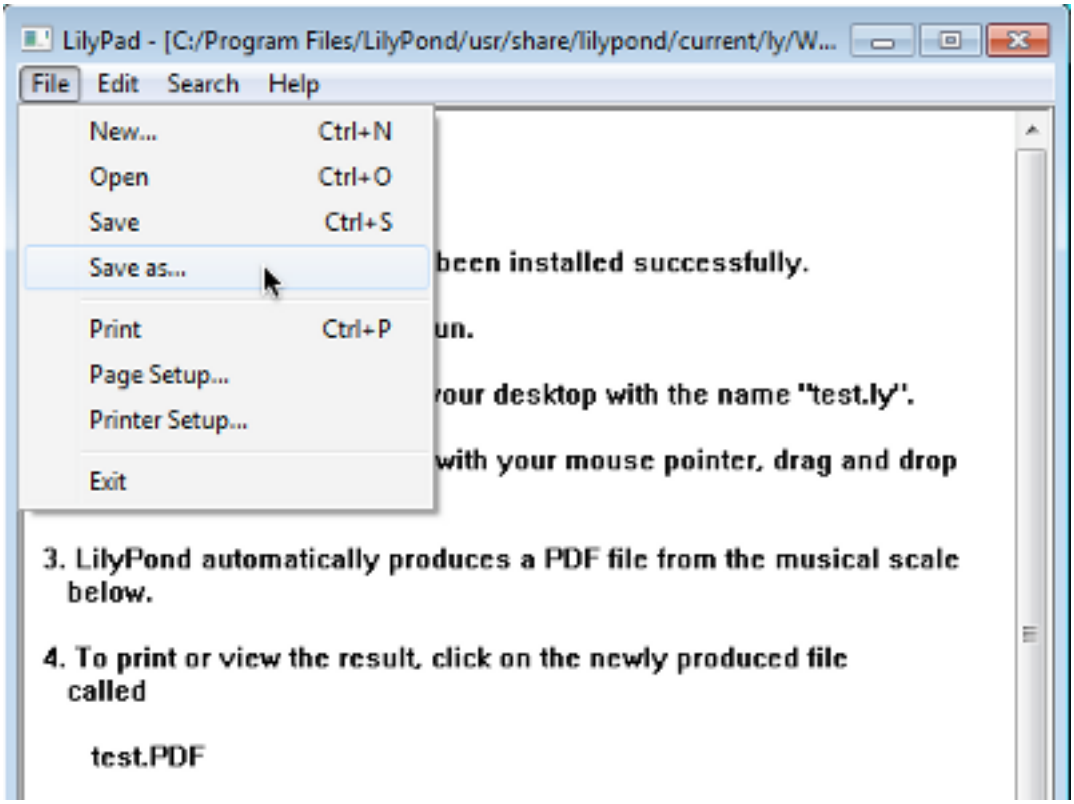
Nota: These instructions assume that you are using the built-in LilyPad editor. If you are using any of the programs described in Sezione "Easier editing" in *Informazioni generali*, please consult the documentation for those programs if you have any problems compiling a file.

Step 1. Create your '.ly' file

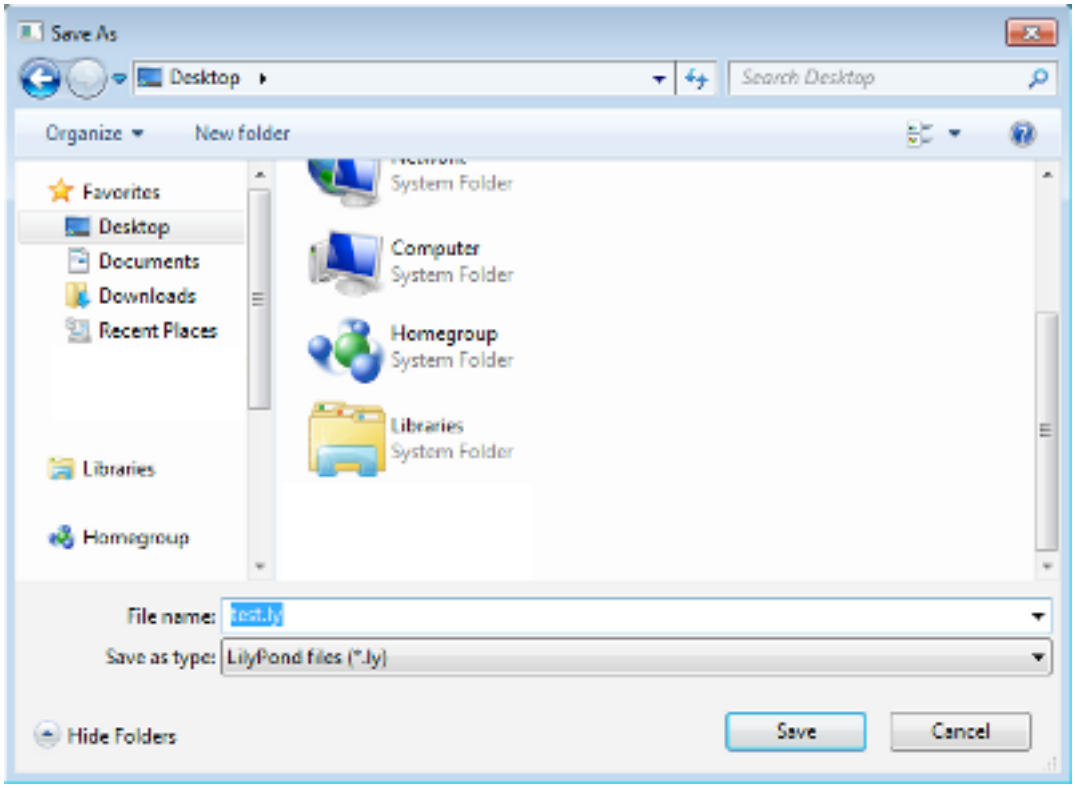
Double-click the LilyPond icon on your desktop, an example file will open.



From the menus that appear along the top of the example file, select `File > Save as`. Do not use the `File > Save` for the example file as this will not work until you have given it a valid LilyPond file name.



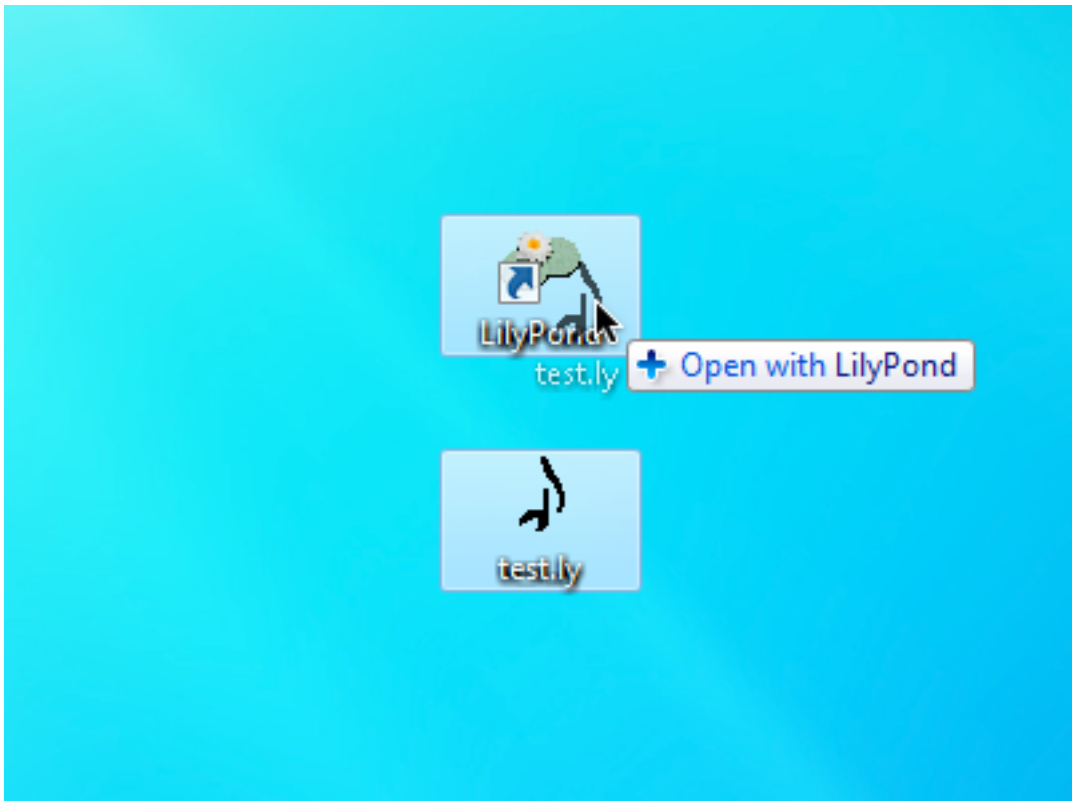
Choose a name for your file, for example '`test.ly`'.



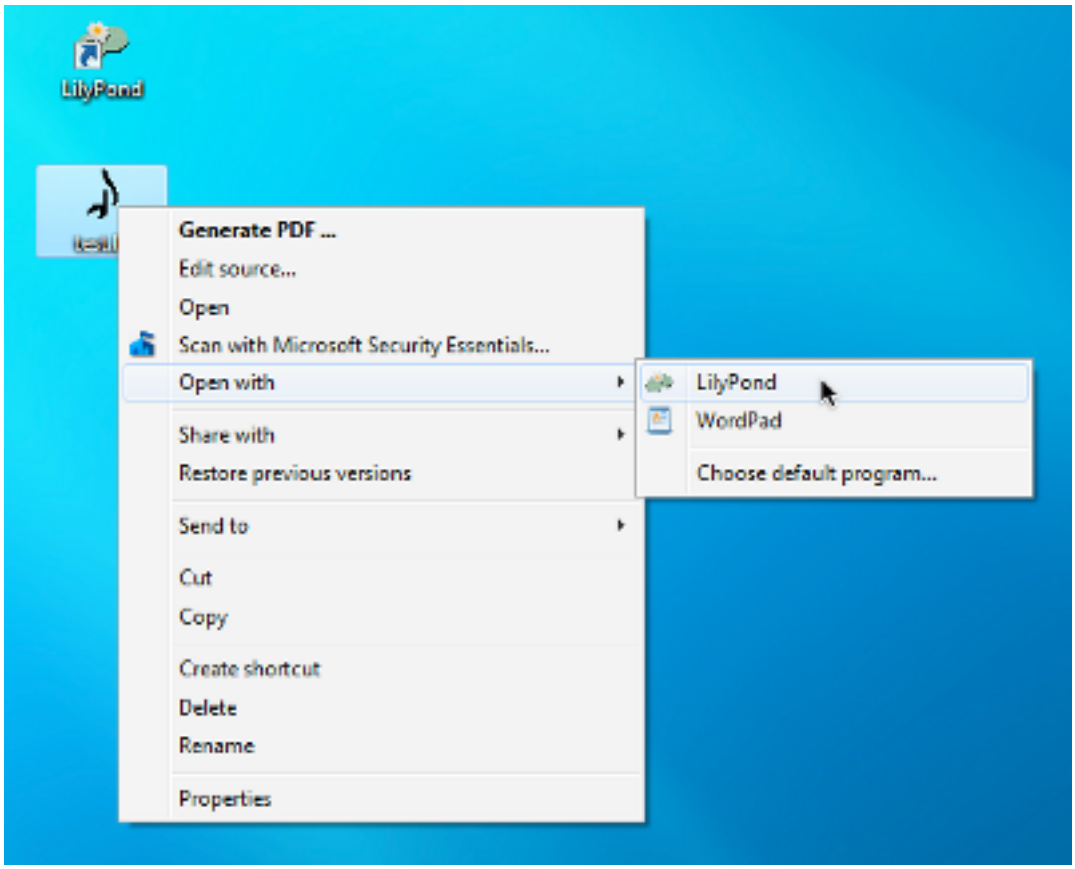
Step 2a. Compile (with drag-and-drop)

Depending on what you prefer, to compile your file either:

Drag-and-drop the file directly onto the LilyPond icon.



Right-click on the file and from the pop-up context menu choose `Open with > LilyPond`.

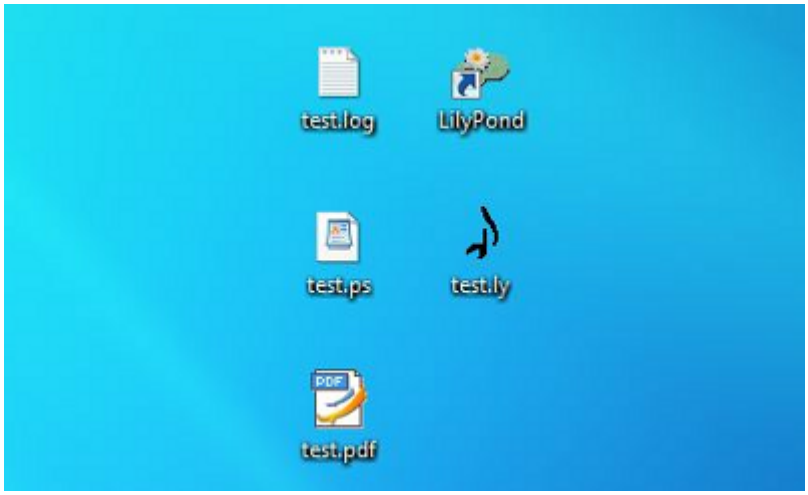


Step 2b. Compile (with double-clicking)

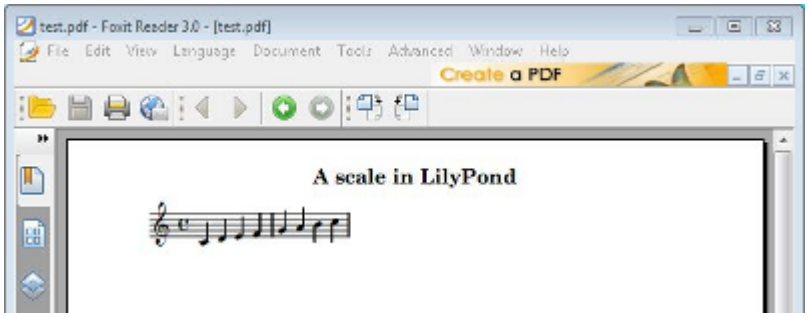
Or simply double-click the '`test.ly`'.

Step 3. View output

During the compilation of the '`test.ly`' file, a command window will, very briefly open and then close. Three additional files will have been created during this process.

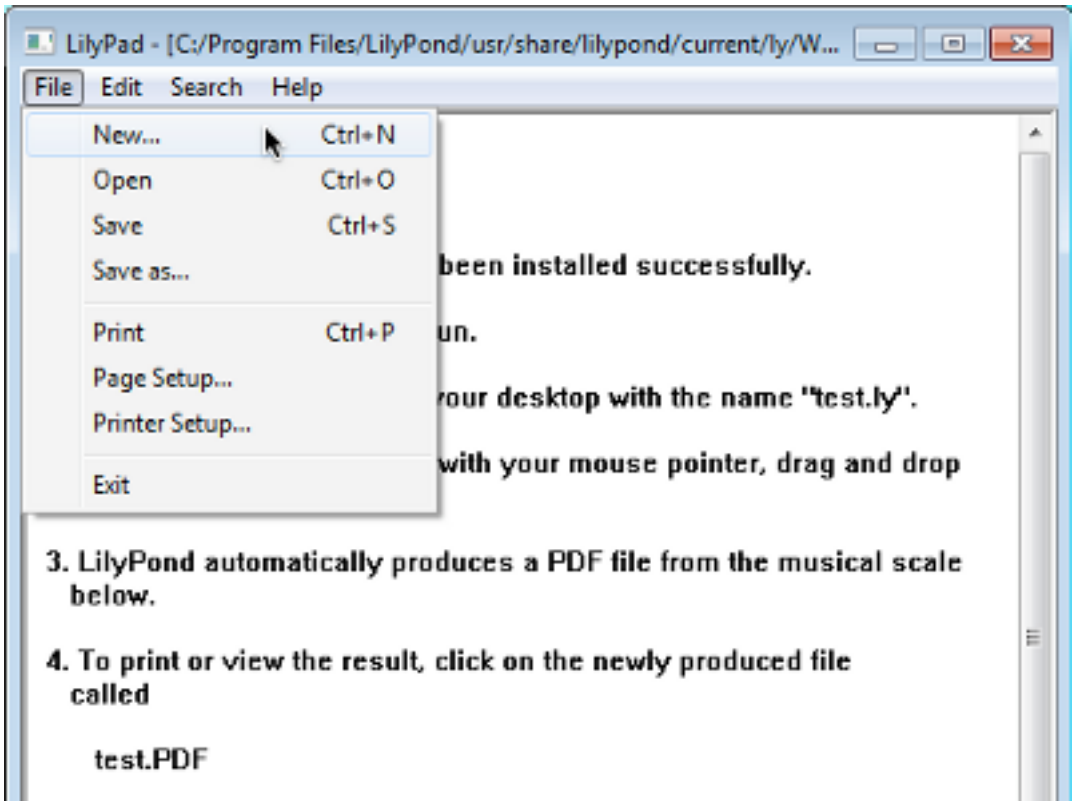


The PDF file contains the engraved '`test.ly`' file.

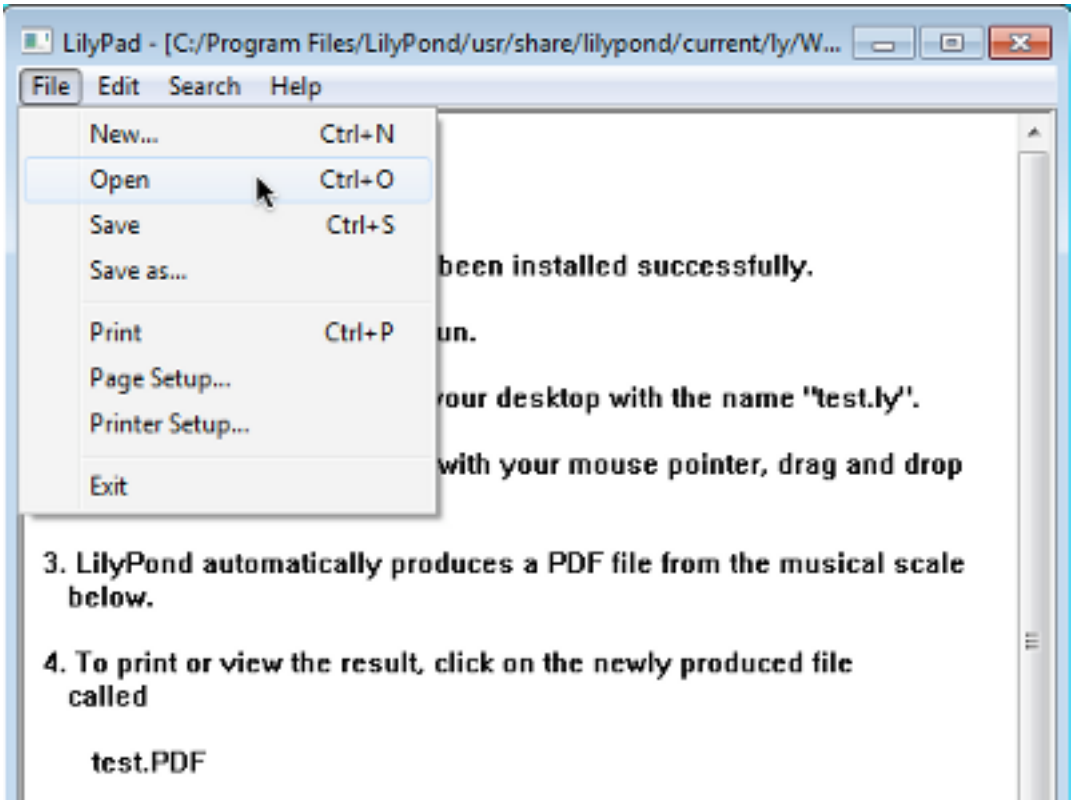


Other commands

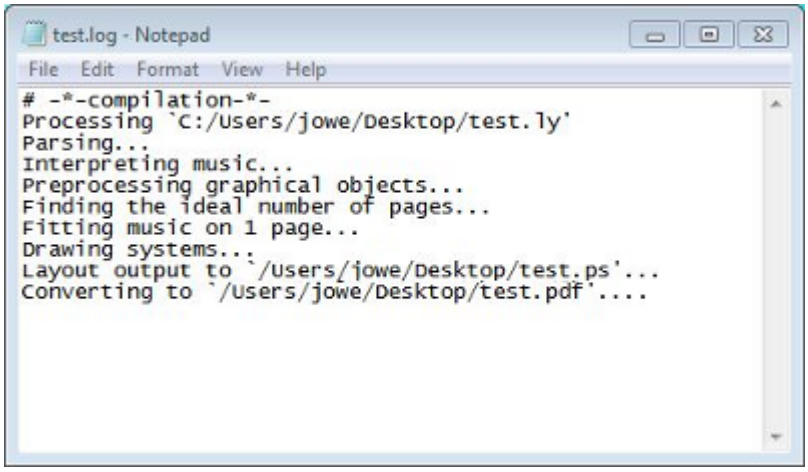
To create a new file, begin by selecting `File > New` from within any previously created file.



or `File > Open` to open and edit any files you have saved before.



You must save any new edits you make before you compile it and if the PDF file is not created, check the log file that will have been created during the compilation attempt, for any errors.



This log file is overwritten each time you compile your LilyPond file.

The PS file is used internally by LilyPond to create the PDF file and can be ignored. It also gets overwritten each time you compile your file.

If you are viewing your file in a PDF viewer, then you must close the PDF if you wish to make a new compilation as it may fail to create the new PDF while it is still being viewed.

1.1.4 Linea di comando

> **Nota:** These instructions assume that you are familiar with command-line programs. If you are using any of the programs described in Sezione "Easier editing" in *Informazioni generali*, please consult the documentation for those programs if you have any problems compiling a file.

Step 1. Create your '`.ly`' file

Create a text file called '`test.ly`' and enter:

```
\version "2.14.1"
{
c' e' g' e'
}
```

Step 2. Compile (with command-line)

To process '`test.ly`', type the following at the command prompt:

```
lilypond test.ly
```

You will see something resembling:

```
GNU LilyPond 2.14.1
Processing `test.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Solving 1 page-breaking chunks...[1: 1 pages]
Drawing systems...
Layout output to `test.ps'...
Converting to `./test.pdf'...
```

Step 3. View output

You may view or print the resulting '`test.pdf`'.
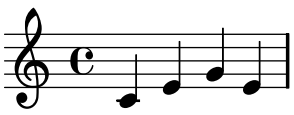
1.2 Come scrivere i file di input

Questa sezione introduce le basi della sintassi di LilyPond e ha l'obiettivo di aiutarti ad iniziare a scrivere i file di input.

1.2.1 Notazione semplice

LilyPond aggiungerà automaticamente alcuni elementi della notazione. Nell'esempio seguente, abbiamo specificato soltanto quattro note, ma LilyPond ha aggiunto una chiave, il tempo e le durate.

```
{
  c' e' g' e'
}
```
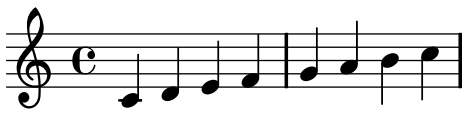


Questo comportamento può essere modificato, ma nella maggior parte dei casi questi valori sono utili.

Altezze

Glossario musicale: Sezione “altezza” in *Glossario Musicale*, Sezione “intervallo” in *Glossario Musicale*, Sezione “scala” in *Glossario Musicale*, Sezione “Do centrale” in *Glossario Musicale*, Sezione “ottava” in *Glossario Musicale*, Sezione “alterazione” in *Glossario Musicale*.
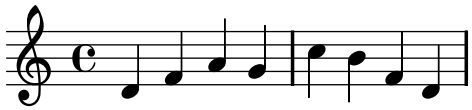
Il modo più semplice per inserire le note è usare il modo `\relative` (relativo). In questo modo, l’ottava viene scelta automaticamente in base al principio per cui la nota che segue deve essere posizionata vicino a quella precedente, ovvero deve essere posizionata nell’ottava che si trova entro tre spazi di pentagramma dalla nota precedente. Per iniziare, scriveremo il pezzo musicale più elementare, una *scala*, in cui ogni nota si trova entro una distanza di appena uno spazio di pentagramma dalla nota precedente.

```
% set the starting point to middle C
\relative c' {
  c d e f
  g a b c
}
```



La nota iniziale è un *Do centrale*. Ogni nota successiva viene posta il più vicino possibile alla nota precedente – in altre parole, la prima `c` è il Do più vicino al Do centrale. Questo è seguito dal Re più vicino alla nota precedente. Possiamo creare melodie che hanno intervalli più ampi, sempre usando soltanto il modo `\relative`:
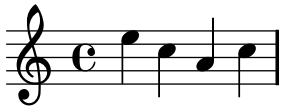
```
\relative c' {
  d f a g
  c b f d
}
```



Non è necessario che la prima nota della melodia inizi con la nota che specifica l’altezza iniziale. Nell’esempio precedente, la prima nota – il `Re` – è il Re più vicino al Do centrale.
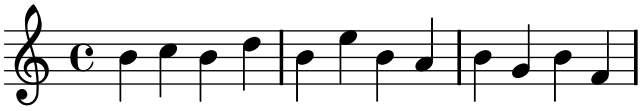
Se si aggiungono (o si rimuovono) apostrofi `'` o virgole `,` dal comando ‘`\relative c'`’, possiamo cambiare l’ottava di partenza:

```
% one octave above middle C
\relative c'' {
  e c a c
}
```



Il modo relativo all’inizio può apparire disorientante, ma è il sistema più semplice per inserire gran parte delle melodie. Vediamo come questo calcolo relativo funziona in pratica. Se si parte da un Si, che si trova sulla linea centrale in chiave di violino, si possono raggiungere un Do, un Re e un Mi entro 3 spazi di pentagramma andando in su, e un La, un Sol e un Fa entro 3 spazi di pentagramma andando in giù. Quindi se la nota che segue il Si è un Do, un Re o un Mi, si troverà sopra il Si, mentre il La, il Sol o il Fa si troveranno sotto.
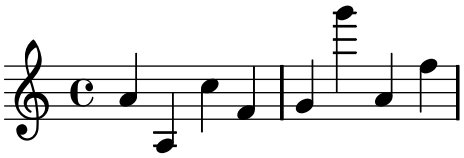
```
\relative c'' {
  b c  % c is 1 staff space up, so is the c above
  b d  % d is 2 up or 5 down, so is the d above
  b e  % e is 3 up or 4 down, so is the e above
  b a  % a is 6 up or 1 down, so is the a below
  b g  % g is 5 up or 2 down, so is the g below
  b f  % f is 4 up or 3 down, so is the f below
}
```



Accade esattamente lo stesso quando qualcuna di queste note ha un diesis o un bemolle. Le *Alterazioni* sono **completamente ignorate** nel calcolo della posizione relativa. Viene fatto lo stesso conteggio dello spazio del pentagramma da una nota in una qualsiasi altra posizione sul pentagramma.

Per inserire intervalli che sono più ampi di tre spazi di pentagramma, si può alzare l'*ottava* aggiungendo al nome della nota una virgoletta ad apice singolo `'` (o apostrofo). Si può abbassare l'ottava aggiungendo al nome della nota una virgola `,`.

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



Per cambiare una nota di due (o più!) ottave, si usano multipli di `''` o `,,` – ma stai attento a usare due virgolette ad apici semplici `''` e non quelle a doppi apici `"` !

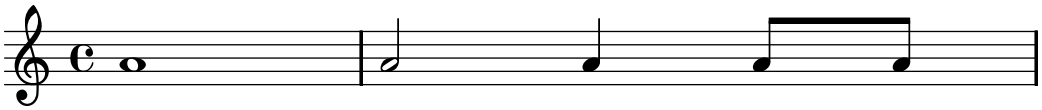
Durate (ritmi)

Glossario musicale: Sezione "coda" in *Glossario Musicale*, Sezione "durata" in *Glossario Musicale*, Sezione "semibreve" in *Glossario Musicale*, Sezione "minima" in *Glossario Musicale*, Sezione "semiminima" in *Glossario Musicale*, Sezione "nota puntata" in *Glossario Musicale*.

La *durata* di una nota viene indicata da un numero che segue il nome della nota: `1` per una *semibreve*, `2` per una *minima*, `4` per una *semiminima* e così via. Le *Code* vengono aggiunte in automatico.

Se non si specifica una durata, si usa la durata precedente anche per la nota successiva. La durata predefinita della prima nota è di un quarto.

```
\relative c'' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a a2
}
```





Per creare *note puntate*, aggiungi un punto `.` al numero di durata. La durata di una nota puntata deve essere dichiarata esplicitamente (cioè con un numero).

```
\relative c'' {
  a a a4. a8
  a8. a16 a a8. a8 a4.
}
```
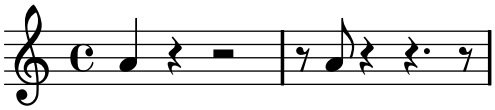


Pause

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

Una *pausa* viene inserita proprio come una nota ma col nome `r` :

```
\relative c'' {
  a4 r r2
  r8 a r4 r4. r8
}
```
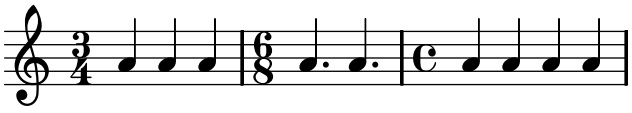


Indicazione di tempo

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

Il *tempo* si imposta con il comando `\time`:

```
\relative c'' {
  \time 3/4
  a4 a a
  \time 6/8
  a4. a
  \time 4/4
  a4 a a a
}
```



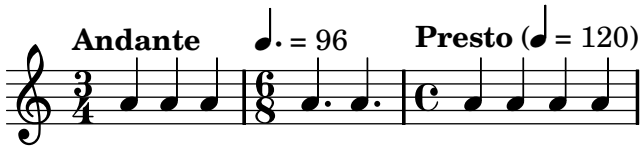
Indicazioni di tempo

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

L’ *indicazione di tempo* e l’ *indicazione metronomica* si impostano col comando `\tempo`:

```
\relative c'' {
  \time 3/4
  \tempo "Andante"
  a4 a a
  \time 6/8
```

```
  \tempo 4. = 96
  a4. a
  \time 4/4
  \tempo  "Presto" 4 = 120
  a4 a a a
}
```
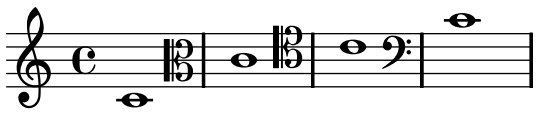


Chiave

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

La *chiave* si imposta con il comando `\clef`:
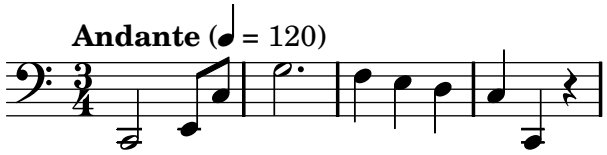
```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



Tutto insieme

Ecco un piccolo esempio che mostra tutti questi elementi insieme:

```
\relative c, {
  \clef "bass"
  \time 3/4
  \tempo "Andante" 4 = 120
  c2 e8 c'
  g'2.
  f4 e d
  c4 c, r
}
```



Vedi anche

Guida alla notazione: Sezione “Writing pitches” in *Guida alla Notazione*, Sezione “Writing rhythms” in *Guida alla Notazione*, Sezione “Writing rests” in *Guida alla Notazione*, Sezione “Time signature” in *Guida alla Notazione*, Sezione “Clef” in *Guida alla Notazione*.

1.2.2 Lavorare sui file di input

I file di input di LilyPond sono simili ai file sorgenti di molti comuni linguaggi di programmazione. Contengono una dichiarazione di versione, sono sensibili alle maiuscole, e in generale gli spazi bianchi vengono ignorati. Le espressioni musicali si formano con parentesi graffe `{ }`, e i commenti sono indicati con `%` o `%{ ... %}` .

Se le frasi precedenti ti sembrano assurde, non preoccuparti! Spiegheremo cosa significano tutti questi termini:

- **Dichiarazione di versione**: Ogni file LilyPond deve contenere una dichiarazione di versione. Una dichiarazione di versione è una linea che indica la versione di LilyPond per la quale il file è stato scritto, come nel seguente esempio:

  ```
  \version "2.14.2"
  ```

  Per convenzione, la dichiarazione di versione viene posta all'inizio del file LilyPond.

  La dichiarazione di versione è importante per almeno due ragioni. Primo, permette l'aggiornamento automatico del file di input file via via che la sintassi di LilyPond cambia. Secondo, indica la versione di LilyPond richiesta per compilare il file.

  Se la dichiarazione di versione è omessa dal file di input, LilyPond mostra un avvertimento durante la compilazione del file.

- **Sensibile alle maiuscole**: distingue tra lettere in minuscolo (es: `a, b, s, t`) o in maiuscolo (es: `A, B, S, T`). Le note sono in minuscolo: `{ c d e }` è un input valido; `{ C D E }` causerà un messaggio di errore.

- **Insensibile agli spazi**: non importa quanti spazi (o tabulazioni o nuove linee) aggiungi. '`{ c4 d e }`' ha lo stesso significato di '`{ c4 d e }`' e:

  ```
  { c4                        d
                        e   }
  ```

  Certo, l'esempio precedente è scomodo da leggere. Una buona regola pratica è indentare i blocchi di codice con una tabulazione o due spazi:

  ```
  {
    c4 d e
  }
  ```

  Tuttavia, uno spazio bianco *è* necessario per separare molti elementi sintattici gli uni dagli altri. In altre parole, lo spazio bianco può essere *aggiunto*, ma non può essere *eliminato*. Dal momento che uno spazio bianco mancante può dare adito a strani errori, si consiglia di inserire sempre uno spazio bianco prima e dopo ogni elemento sintattico, ad esempio, prima e dopo ogni parentesi graffa.

- **Espressioni**: ogni parte dell'input di LilyPond deve avere `{` **parentesi graffe** `}` intorno. Queste parentesi dicono a LilyPond che l'input costituisce un'espressione musicale singola, proprio come le parentesi `()` in matematica. Per evitare ambiguità, le parentesi dovrebbero essere racchiuse tra spazi, a meno che non si trovino all'inizio o alla fine di una linea.

  Anche un comando di LilyPond seguito da un'espressione semplice in parentesi (come '`\relative c' { ... }`') conta come un'espressione musicale singola.

- **Commenti**: un commento è un appunto per il lettore umano dell'input musicale; viene ignorato quando l'input viene analizzato, dunque non ha alcun effetto sull'output finale. Ci sono due tipi di commenti. Il simbolo di percentuale `%` introduce un commento di linea; tutto quello che sta dopo `%` su quella linea verrà ignorato. Per convenzione, un commento di linea viene posto *sopra* il codice a cui si riferisce.

  ```
  a4 a a a
  % questo commento si riferisce ai Si
  b2 b
  ```

Un commento di blocco segna come commento un'intera sezione di input musicale. Tutto ciò che è compreso tra `%{` e `%}` viene ignorato. Tuttavia, i commenti di blocco non si 'annidano'. Ovvero non si può inserire un commento di blocco dentro a un altro commento di blocco. Se ci provi, il primo `%}` interromperà *entrambi* i commenti di blocco. Il seguente frammento mostra gli usi possibili per i commenti:

```
% ecco le note di "Ah! Vous dirai-je, Maman"
  c4 c g' g a a g2

%{
  Questa linea e le note sotto vengono ignorate,
  perché si trovano in un commento di blocco.

  f4 f e e d d c2
%}
```

1.3 Gestire gli errori

Talvolta LilyPond non genera l'output che desideri. Questa sezione fornisce alcuni link che possono aiutarti a risolvere i problemi che potrai incontrare.

1.3.1 Consigli generali per la risoluzione dei problemi

La risoluzione dei problemi in LilyPond può essere un compito impegnativo per le persone che sono abituate a un'interfaccia grafica, perché è possibile creare dei file di input non validi. Quando questo accade, un approccio logico è il modo migliore per identificare e risolvere il problema. Alcune linee guida che ti aiutano a imparare questo sono indicate in Sezione "Risoluzione dei problemi" in *Uso del Programma*.

1.3.2 Alcuni errori comuni

Ci sono alcuni errori comuni che sono difficili da risolvere basandosi soltanto sui messaggi di errore che compaiono. Questi sono descritti in Sezione "Errori comuni" in *Uso del Programma*.

1.4 Come leggere i manuali

Questa sezione spiega come leggere la documentazione in modo efficiente e introduce alcune utili funzionalità interattive che sono disponibili nella versione online della documentazione.

1.4.1 Materiale omesso

L'input di LilyPond deve essere compreso dai segni `{ }` o da '`\relative c'' { ... }`', come abbiamo visto in Sezione 1.2.2 [Lavorare sui file di input], pagina 18. Nel resto di questo manuale gran parte degli esempi ometterà questi segni. Per replicare gli esempi, puoi copiare e incollare l'input mostrato, ma **devi** aggiungere '`\relative c'' { ... }`' in questo modo:

```
\relative c'' {
  ...inserire qui l'esempio...
}
```
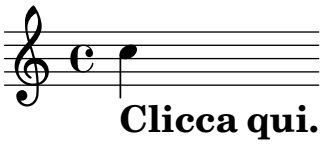
Perché omettere le parentesi? Gran parte degli esempi in questo manuale possono essere inseriti nel mezzo di un pezzo più ampio. Per questi esempi, non ha senso includere '`\relative c'' { ... }`' – non si deve mettere un `\relative` dentro un altro `\relative`! Se includessimo '`\relative c'' { ... }`' in ogni esempio, non potresti copiare un piccolo esempio della documentazione e incollarlo dentro a un tuo brano più esteso. La maggior parte delle persone vuole aggiungere del materiale a un brano esistente, per questo abbiamo strutturato il manuale in questo modo.

Ricorda che i file LilyPond devono avere una dichiarazione di versione (`\version`). Nei manuali la dichiarazione è omessa perché gli esempi sono frammenti di codice e non file completi. Ma dovresti prendere l'abitudine di includerla nei tuoi file.

1.4.2 Esempi cliccabili

> **Nota:** Queste funzionalità sono disponibili soltanto nei manuali in HTML.

Molte persone imparano ad usare un programma provando e smanettando. Questo è possibile anche con LilyPond. Se clicchi su un'immagine nella versione HTML di questo manuale, vedrai l'esatto input LilyPond usato per generare quell'immagine. Provalo su questa immagine:



Clicca qui.

Tagliando e copiando quel che si trova nella sezione "ly snippet", puoi ricavare un modello di partenza per sperimentare. Per vedere esattamente lo stesso output (larghezza della linea e tutto il resto), copia tutto da "Start cut-&-pastable section" alla fine del file.

1.4.3 Panoramica dei manuali

La documentazione su LilyPond è vasta. I nuovi utenti talvolta sono confusi su quali parti dovrebbero leggere, e ogni tanto saltano la lettura di parti fondamentali.

> **Nota:** Non saltare le parti importanti della documentazione. Altrimenti ti risulterà molto più difficile comprendere le sezioni successive.

- **Prima di fare** *qualsiasi* **tentativo**: leggi il Capitolo 1 [Tutorial], pagina 1 e la Capitolo 2 [Notazione comunemente utilizzata], pagina 21 del Manuale di apprendimento. Se ti imbatti in termini musicali che non conosci, cercali nel Sezione "Glossario" in *Glossario Musicale*.
- **Prima di tentare di scrivere un pezzo musicale completo**: leggi i Capitolo 3 [Concetti fondamentali], pagina 41 del Manuale di apprendimento. Dopo puoi dare un'occhiata alle sezioni rilevanti della Sezione "Notation reference" in *Guida alla Notazione*.
- **Prima di cercare di cambiare l'output predefinito**: leggi il Capitolo 4 [Tweaking output], pagina 87 del Manuale di apprendimento.
- **Prima di iniziare un grande progetto**: leggi il documento Sezione "Consigli su come scrivere i file" in *Uso del Programma* del manuale di Uso del programma.

2 Notazione comunemente utilizzata

Questo capitolo spiega come creare dei belli spartiti che facciano uso della notazione musicale comunemente utilizzata, seguendo il materiale esposto nel Capitolo 1 [Tutorial], pagina 1.
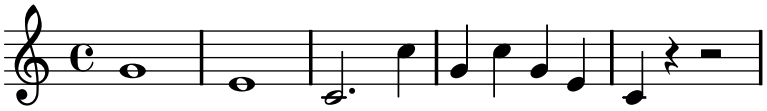
2.1 Notazione su un solo pentagramma

Questa sezione presenta la notazione comunemente usata per una singola voce su un solo pentagramma.

2.1.1 Controlli di battuta

I *controlli di battuta*, pur se non strettamente necessari, dovrebbero essere usati nel codice di input per mostrare dove si vuole che cadano le stanghette. Vengono inseriti col simbolo della barra verticale, `|`. Grazie ai controlli di battuta, il programma può verificare che tu abbia inserito delle durate che facciano sì che ogni misura raggiunga la giusta durata. I controlli di battuta rendono anche il codice di input più facile da leggere, perché aiutano a tenere tutto in ordine.

```
g1 | e1 | c2. c'4 | g4 c g e | c4 r r2 |
```



Vedi anche

Guida alla notazione: Sezione “Bar and bar number checks” in *Guida alla Notazione*.

2.1.2 Alterazioni e armature di chiave
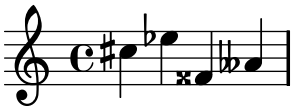
> **Nota:** I nuovi utenti sono spesso confusi riguardo a questi concetti – leggi il messaggio di avviso in fondo a questa pagina, soprattutto se non hai una buona conoscenza della teoria musicale!

Alterazioni

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 “alterazione o accidente” in *Glossario Musicale*.

Un *diesis* si ottiene aggiungendo il suffisso `is` al nome della nota, e un *bemolle* aggiungendo `es`. Come puoi immaginare, un *doppio diesis* o un *doppio bemolle* si ottengono aggiungendo `isis` o `eses`. Questa sintassi deriva dalle convenzioni per i nomi delle note presenti nelle lingue nordiche e germaniche, come il tedesco e l’olandese. Per usare altri nomi per le *alterazioni*, si veda Sezione “Note names in other languages” in *Guida alla Notazione*.

```
cis4 ees fisis, aeses
```
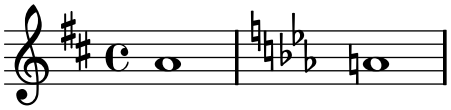


Armature di chiave

Glossario musicale: Sezione “armatura di chiave” in *Glossario Musicale*, Sezione “maggiore” in *Glossario Musicale*, Sezione “minore” in *Glossario Musicale*.

L’ *armatura di chiave* viene impostata col comando `\key` seguito da un’altezza e da `\major` o `\minor`.

```
\key d \major
a1 |
\key c \minor
a1 |
```



Attenzione: armature di chiave e altezze
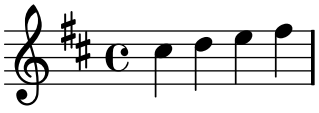
Glossario musicale: Sezione “alterazione” in *Glossario Musicale*, Sezione “armatura di chiave” in *Glossario Musicale*, Sezione “altezza” in *Glossario Musicale*, Sezione “bemolle” in *Glossario Musicale*, Sezione “bequadro” in *Glossario Musicale*, Sezione “diesis” in *Glossario Musicale*, Sezione “trasposizione” in *Glossario Musicale*.

Per determinare se mostrare o meno un’*alterazione*, LilyPond esamina le altezze e l’*armatura di chiave*. L’armatura di chiave influisce soltanto sulle alterazioni che vengono *mostrate*, non sull’*altezza* della nota! Questa è una caratteristica che spesso causa confusione ai nuovi utenti, quindi la spiegheremo più dettagliatamente.

LilyPond fa una netta distinzione tra contenuto musicale e aspetto grafico. L’alterazione (*bemolle*, *bequadro* o *diesis*) di una nota fa parte dell’altezza ed è quindi contenuto musicale. Se un’alterazione (un segno *stampato* di bemolle, bequadro o diesis) venga posta oppure no di fronte alla nota corrispondente è una questione di aspetto grafico. La formattazione segue delle regole, dunque le alterazioni sono inserite automaticamente in base a queste regole. Le altezze nella tua musica sono opere d’arte, quindi non verranno aggiunte in automatico: sei tu a dover inserire la nota che vuoi sentire.

In questo esempio:
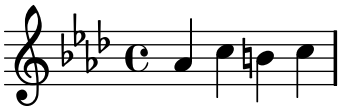
```
\key d \major
cis4 d e fis
```



Nessuna nota ha un’alterazione rispetto all’armatura di chiave, ma devi comunque aggiungere `is` e scrivere `cis` e `fis` nel file di input.

Il codice `b` non significa “stampa un punto nero esattamente nella linea centrale del pentagramma.” Piuttosto, significa “c’è una nota con altezza Si-bequadro.” Nella tonalità di La bemolle maggiore, *deve* avere un’alterazione:

```
\key aes \major
aes4 c b c
```



Se l'esempio precedente sembra poco chiaro, considera questo: se tu stessi suonando un pianoforte, quale tasto premeresti? Se premi un tasto nero, allora *devi* aggiungere `-is` o `-es` al nome della nota!

Aggiungere esplicitamente tutte le alterazioni richiederà un po' più di lavoro in fase di scrittura, ma il vantaggio è che la *trasposizione* è più semplice, e le alterazioni possono essere prodotte usando diverse convenzioni. Per alcuni esempi che mostrano come sia possibile produrre delle alterazioni in base a regole diverse, si veda Sezione "Automatic accidentals" in *Guida alla Notazione*.

Vedi anche

Guida alla notazione: Sezione "Note names in other languages" in *Guida alla Notazione*, Sezione "Accidentals" in *Guida alla Notazione*, Sezione "Automatic accidentals" in *Guida alla Notazione*, Sezione "Key signature" in *Guida alla Notazione*.
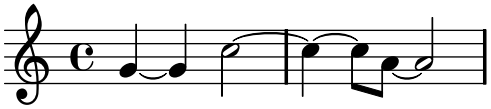
2.1.3 Legature di valore e di portamento

Legature di valore

Glossario musicale: Sezione "legatura di valore" in *Glossario Musicale*.

Una *legatura di valore* si ottiene apponendo una tilde `~` alla prima nota della legatura.

```
g4~ g c2~ | c4~ c8 a~ a2 |
```
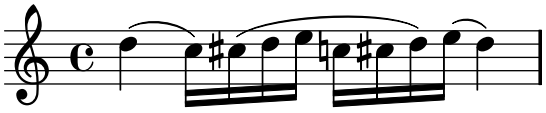


Legature di portamento

Glossario musicale: Sezione "legatura di portamento" in *Glossario Musicale*.

Una *legatura di portamento* è una linea curva che collega più note. La nota iniziale e quella finale sono indicate rispettivamente con `(` e `)`.

```
d4( c16) cis( d e c cis d) e( d4)
```
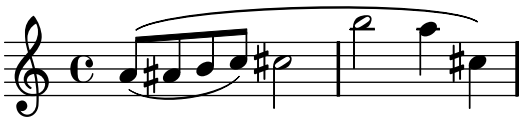


Legature di frase

Glossario musicale: Sezione "legatura di portamento" in *Glossario Musicale*, Sezione "legatura di frase" in *Glossario Musicale*.

Le legature di portamento che indicano una *frase* più lunga possono essere inserite con `\(` e `\)`. E' possibile avere allo stesso tempo sia le *legature di portamento* sia le legature di frase, ma non si possono avere simultaneamente diverse legature di portamento, o diverse legature di frase.

```
a8(\( ais b c) cis2 b'2 a4 cis,\)
```
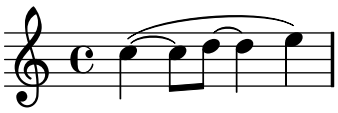


Attenzione: legature di portamento vs. legature di valore

Glossario musicale: Sezione “articolazione” in *Glossario Musicale*, Sezione “legatura di portamento” in *Glossario Musicale*, Sezione “legatura di valore” in *Glossario Musicale*.

Una *legatura di portamento* ha lo stesso aspetto di una *legatura di valore*, ma un significato diverso. Una legatura di valore rende semplicemente la nota più lunga, e può essere usata solo con coppie di note della stessa altezza. Le legature di portamento indicano l’*articolazione* delle note, e possono essere usate con ampi gruppi di note. Legature di valore e legature di portamento possono essere annidate le une dentro le altre.

```
c4~( c8 d~ d4 e)
```



Vedi anche

Guida alla notazione: Sezione “Ties” in *Guida alla Notazione*, Sezione “Slurs” in *Guida alla Notazione*, Sezione “Phrasing slurs” in *Guida alla Notazione*.
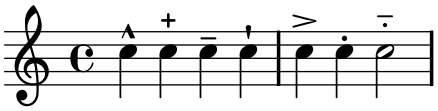
2.1.4 Articolazione e dinamica

Articolazioni

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

Le *articolazioni* di uso comune possono essere aggiunte a una nota con una lineetta `-` e un singolo carattere:

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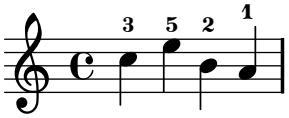


Diteggiature

Glossario musicale: Sezione “diteggiatura” in *Glossario Musicale*.
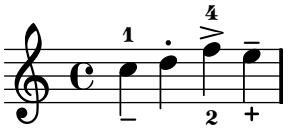
Analogalmente, le indicazioni di *diteggiatura* possono essere aggiunte a una nota con una lineetta (`-`) e il numero che si vuole visualizzare:

```
c4-3 e-5 b-2 a-1
```



Articolazioni e diteggiature solitamente sono posizionate verticalmente in automatico, ma si può specificarne la direzione sostituendo la lineetta (`-`) con `^` (su) o `_` (giù). Si possono usare anche articolazioni multiple sulla stessa nota. Tuttavia, nella maggior parte dei casi è meglio lasciare che sia LilyPond a determinare le direzioni delle articolazioni.

```
c4_-^1 d^. f^4_2-> e^-_+
```
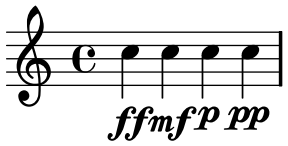


Dinamica

Glossario musicale: Sezione “dinamica” in *Glossario Musicale*, Sezione “crescendo” in *Glossario Musicale*, Sezione “decrescendo” in *Glossario Musicale*.
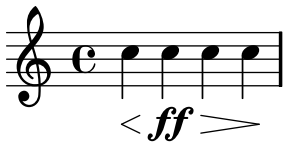
I segni di *dinamica* si ottengono aggiungendo alla nota i simboli (preceduti da un segno di barra invertita, o backslash):

```
c4\ff c\mf c\p c\pp
```



Crescendi e *decrescendi* iniziano coi comandi `\<` e `\>`. Il seguente segno di dinamica, ad esempio `\ff`, terminerà il (de)crescendo, oppure può essere usato il comando `\!`:

```
c4\< c\ff\> c c\!
```



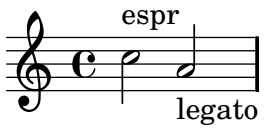
Vedi anche

Guida alla notazione: Sezione “Articulations and ornamentations” in *Guida alla Notazione*, Sezione “Fingering instructions” in *Guida alla Notazione*, Sezione “Dynamics” in *Guida alla Notazione*.

2.1.5 Aggiungere il testo

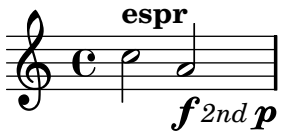
Puoi aggiungere del testo nei tuoi spartiti:

```
c2^"espr" a_"legato"
```



Per aggiungere delle formattazioni puoi usare il comando `\markup`:

```
c2^\markup{ \bold espr}
a2_\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
}
```



Vedi anche
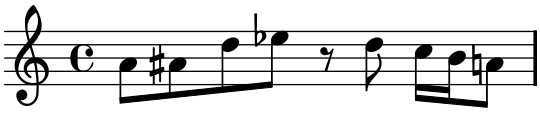
Guida alla notazione: Sezione “Writing text” in *Guida alla Notazione*.

2.1.6 Code automatiche e manuali

Glossario musicale: Sezione “travatura” in *Glossario Musicale*.
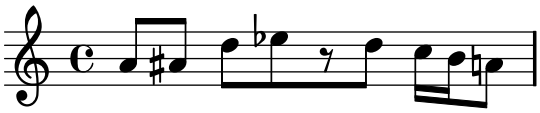
Tutte le *travature* vengono disegnate automaticamente:

```
a8 ais d ees r d c16 b a8
```
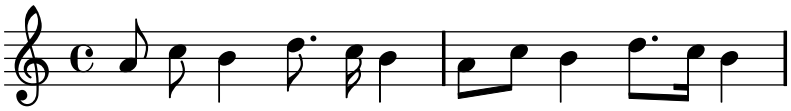


Se non ti piace il modo in cui vengono prodotte automaticamente le travature, è possibile sovrascriverle manualmente. Per correggere soltanto una singola travatura, indica la prima nota da raggruppare con `[` e l’ultima con `]`.

```
a8[ ais] d[ ees r d] c16 b a8
```



Se desideri disattivare il raggruppamento automatico, interamente o per un’ampia sezione del brano, usa il comando `\autoBeamOff` per disattivare il raggruppamento automatico e `\autoBeamOn` per riattivarlo.

```
\autoBeamOff
a8 c b4 d8. c16 b4 |
\autoBeamOn
a8 c b4 d8. c16 b4 |
```



Vedi anche

Guida alla notazione: Sezione “Automatic beams” in *Guida alla Notazione*, Sezione “Manual beams” in *Guida alla Notazione*.
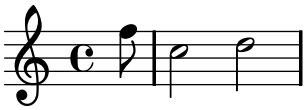
2.1.7 Comandi di tempo avanzati

Battuta parziale

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

Un’ *anacrusi* può essere inserita con la parola chiave `\partial`. Questa è seguita da una durata: `\partial 4` è un’anacrusi di semiminima e `\partial 8` di croma.

```
\partial 8 f8 |
c2 d |
```

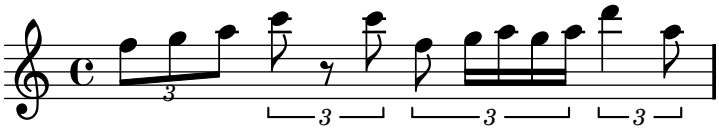


Gruppi irregolari

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

I *gruppi irregolari* vengono preceduti dalla parola chiave `\times`. Questa richiede due argomenti: una frazione e un frammento di musica. La durata del frammento viene moltiplicata per la frazione. Le terzine fanno sì che le note occupino 2/3 della loro durata, quindi una *terzina* ha 2/3 come frazione:

```
\times 2/3 { f8 g a }
\times 2/3 { c8 r c }
\times 2/3 { f,8 g16[ a g a] }
\times 2/3 { d4 a8 }
```
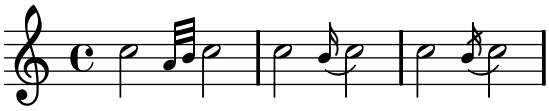


Abbellimenti

Glossario musicale: Sezione "abbellimenti" in *Glossario Musicale*, Sezione "acciaccatura" in *Glossario Musicale*, Sezione "appoggiatura" in *Glossario Musicale*.

Gli *abbellimenti* sono creati col comando `\grace`, ma possono essere creati anche ponendo davanti a un'espressione musicale le parole chiave `\appoggiatura` o `\acciaccatura`:

```
c2 \grace { a32[ b] } c2 |
c2 \appoggiatura b16 c2 |
c2 \acciaccatura b16 c2 |
```



Vedi anche

Guida alla notazione: Sezione "Grace notes" in *Guida alla Notazione*, Sezione "Tuplets" in *Guida alla Notazione*, Sezione "Upbeats" in *Guida alla Notazione*.

2.2 Note simultanee

Questa sezione spiega come inserire più note simultanee: molteplici strumenti, molteplici righi di pentagramma per un singolo strumento (es: piano), e accordi.

In musica per polifonia si intende la presenza di più di una voce in un brano. In LilyPond per polifonia si intende la presenza di più di una voce sullo stesso pentagramma.

2.2.1 Espressioni musicali
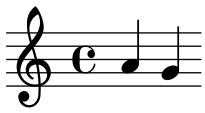
Nei file di input di LilyPond, la musica è rappresentata dalle *espressioni musicali*. Anche una singola nota è un'espressione musicale:

```
a4
```
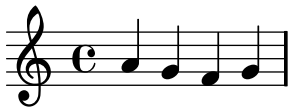


Se si racchiude una nota tra parentesi si crea un'*espressione musicale composta*. In questo esempio abbiamo creato un'espressione musicale composta da due note:

```
{ a4 g4 }
```



Se si mette un gruppo di espressioni musicali (es: note) tra parentesi, significa che sono in sequenza (ovvero, ciascuna espressione segue la precedente). Il risultato è un'altra espressione musicale:

```
{ { a4 g } f4 g }
```



Analogia: le espressioni matematiche

Questo meccanismo è analogo a quello delle formule matematiche: una grande formula può essere creata creando piccole formule. Tali formule sono chiamate espressioni, e possono contenere altre espressioni, così che sia possibile costruire a piacere espressioni grandi e complesse. Ad esempio,

```
1

1 + 2

(1 + 2) * 3

((1 + 2) * 3) / (4 * 5)
```

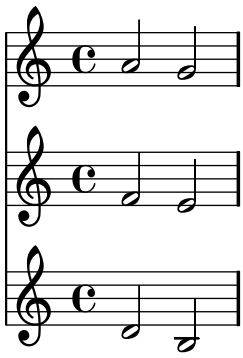
Questa è una sequenza di espressioni, dove ogni espressione è racchiusa in quella successiva (più grande). Le espressioni più semplici sono i numeri, e quelle più grandi si ottengono combinando le espressioni con gli operatori (come `+`, `*` e `/`) e le parentesi. Come le espressioni matematiche, le espressioni musicali possono essere annidate a qualsivoglia grado di profondità, e questo è indispensabile per musica complessa come le partiture polifoniche.

Espressioni musicali simultanee: righi multipli

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

Questa tecnica è utile per la musica *polifonica*. Per inserire della musica che abbia più voci o più linee di pentagramma, basta combinare le espressioni in parallelo. Per indicare che le due voci devono suonare contemporaneamente, basta inserire una combinazione simultanea di espressioni musicali. Un' espressione musicale 'simultanea' si forma racchiudendo le espressioni all'interno di `<<` e `>>`. Nel seguente esempio, tre sequenze (tutte contenenti due note separate) vengono combinate in simultanea:

```
\relative c'' {
  <<
    { a2 g }
    { f2 e }
    { d2 b }
  >>
}
```
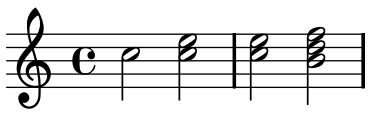


Si noti che abbiamo indentato ogni livello dell'input con una diversa quantità di spazi. LilyPond non si preoccupa di quanto spazio c'è all'inizio di una linea, tuttavia indentare il codice di LilyPond in questo modo lo rende molto più semplice da leggere per l'essere umano.

> **Nota:** ogni nota è relativa alla nota precedente nell'input, e non è relativa al `c''` nel comando `\relative` iniziale.

Espressioni musicali simultanee: rigo singolo

Per determinare il numero di pentagrammi in un brano, LilyPond guarda l'inizio della prima espressione. Se c'è una nota singola, ci sarà un pentagramma; se c'è un'espressione simultanea, ci saranno più pentagrammi. L'esempio seguente mostra un'espressione complessa, ma poiché inizia con una nota singola sarà impostata su un singolo rigo.

```
\relative c'' {
  c2 <<c e>> |
  << { e2 f } { c2 <<b d>> } >>
}
```
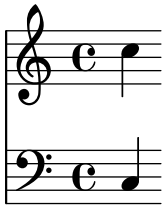


2.2.2 Righi multipli

Come abbiamo visto in Sezione 2.2.1 [Espressioni musicali], pagina 27, i file di input di LilyPond si costruiscono in base alle espressioni musicali. Se la partitura inizia con espressioni musicali simultanee, LilyPond crea più di un rigo musicale. Tuttavia, è più facile vedere quel che accade se creiamo ogni rigo musicale esplicitamente.

Per ottenere più di un rigo musicale, ogni brano musicale che costituisce un rigo è contrassegnato da `\new Staff`. Questi elementi `Staff` vengono poi combinati in parallelo con `<<` e `>>`:

```
\relative c'' {
  <<
    \new Staff { \clef "treble" c4 }
    \new Staff { \clef "bass" c,,4 }
  >>
}
```
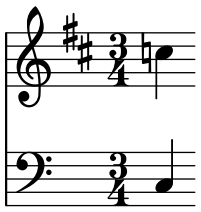


Il comando `\new` introduce un 'contesto di notazione.' Un contesto è un ambiente in cui vengono interpretati gli eventi musicali (come le note o i comandi `\clef`). Nel caso di brani semplici, tali contesti vengono creati automaticamente. Per brani più complessi, è meglio contrassegnare esplicitamente i contesti.

Ci sono vari tipi di contesto. `Score`, `Staff`, e `Voice` gestiscono la notazione melodica, mentre `Lyrics` imposta i testi e `ChordNames` visualizza i nomi degli accordi.

In termini di sintassi, se si inserisce `\new` prima di un'espressione musicale, si crea un'espressione musicale più grande. In questo modo assomiglia al segno di minore in matematica. La formula $(4 + 5)$ è un'espressione, quindi $-(4 + 5)$ è un'espressione più grande.

Le indicazioni di tempo inserite in un rigo musicale si estendono di default a tutti gli altri righi. L'armatura di chiave di un rigo, invece, *non* si estende agli altri. Questo diverso comportamento di default è dovuto al fatto che le partiture con strumenti traspositori sono molto più frequenti delle partiture poliritmiche.

```
\relative c'' {
  <<
    \new Staff { \clef "treble" \key d \major \time 3/4 c4 }
    \new Staff { \clef "bass" c,,4 }
  >>
}
```



2.2.3 Gruppi di pentagrammi
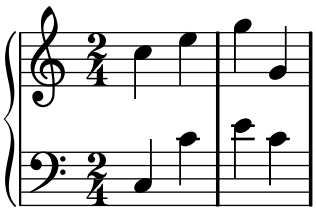
Glossario musicale: Sezione "graffa" in *Glossario Musicale*, Sezione "pentagramma o rigo" in *Glossario Musicale*, Sezione "sistema" in *Glossario Musicale*.

La musica per pianoforte viene stampata su due righi musicali collegati con una *graffa*. Produrre un pentagramma di questo tipo è simile all'esempio polifonico in ⟨undefined⟩ [Multiple staves], pagina ⟨undefined⟩. In questo caso, però, l'intera espressione è inserita all'interno di `PianoStaff`:

```
\new PianoStaff <<
  \new Staff ...
  \new Staff ...
>>
```

Ecco un piccolo esempio:

```
\relative c'' {
  \new PianoStaff <<
    \new Staff { \time 2/4 c4 e | g g, | }
    \new Staff { \clef "bass" c,,4 c' | e c | }
  >>
}
```



Altri raggruppamenti di pentagrammi vengono preceduti da `\new GrandStaff`, per le partiture orchestrali, e da `\new ChoirStaff`, per le partiture corali. Ognuno di questi gruppi di pentagramma costituiscono un altro tipo di contesto, un contesto che genera la graffa all'estremità sinistra di ogni sistema e controlla inoltre l'estensione delle linee della battuta.

Vedi anche
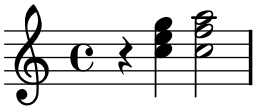
Guida alla notazione: Sezione "Keyboard and other multi-staff instruments" in *Guida alla Notazione*, Sezione "Displaying staves" in *Guida alla Notazione*.

2.2.4 Combinare le note negli accordi

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

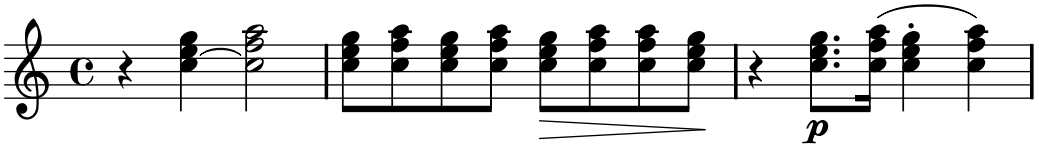
Abbiamo visto in precedenza come le note possano essere combinate in *accordi* racchiudendole tra parentesi a doppi angoli per indicare che sono simultanee. Tuttavia, il modo normale di indicare un accordo è quello di circondare le note con delle parentesi ad angolo *singolo*. Si noti che tutte le note in un accordo devono avere la stessa durata, e che la durata è posta dopo la parentesi chiusa.

```
r4 <c e g> <c f a>2
```



Pensa agli accordi come a qualcosa di equivalente alle note singole: quasi ogni cosa che puoi attaccare a una nota singola può essere attaccata a un accordo, e tutto questo deve stare *fuori* dalle parentesi angolari. Ad esempio, con gli accordi si possono combinare simboli come le travature e le legature di valore. Questi devono essere posti fuori dalle parentesi angolari.

```
r4 <c e g>~ <c f a>2 |
<c e g>8[ <c f a> <c e g> <c f a>] <c e g>\>[ <c f a> <c f a> <c e g>]\! |
r4 <c e g>8.\p <c f a>16( <c e g>4-. <c f a>) |
```



Vedi anche

Guida alla notazione: Sezione "Chorded notes" in *Guida alla Notazione*.

2.2.5 Polifonia su un singolo rigo

La musica polifonica in lilypond, per quanto non difficile, fa riferimento a concetti non ancora affrontati, quindi non la presenteremo subito. Saranno le sezioni successive a introdurre questi concetti e a spiegarli via via.

Vedi anche

Manuale di Apprendimento: Sezione 3.2 [Le voci contengono la musica], pagina 48.

Guida alla notazione: Sezione "Simultaneous notes" in *Guida alla Notazione*.

2.3 Canzoni
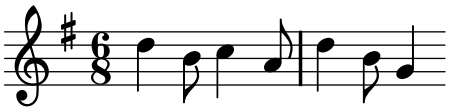
Questa sezione presenta la musica vocale e gli spartiti di semplici canzoni.

2.3.1 Impostare canzoni semplici

Glossario musicale: Sezione “testo” in *Glossario Musicale*.
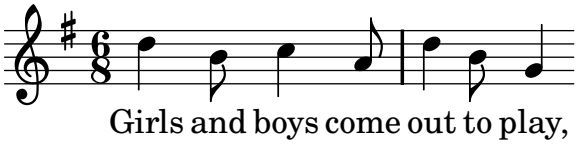
Questo è l’inizio della melodia di una filastrocca, *Girls and boys come out to play*:

```
\relative c'' {
  \key g \major
  \time 6/8
  d4 b8 c4 a8 | d4 b8 g4
}
```



I *testi* possono essere collegati a queste note, combinandoli a esse per mezzo della parola chiave `\addlyrics`. I testi si inseriscono separando ogni sillaba con uno spazio.

```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4
  }
  \addlyrics {
    Girls and boys come | out to play,
  }
>>
```



> **Nota:** È fondamentale che l’ultima sillaba sia separata dalla parentesi graffa di chiusura con uno spazio o una nuova linea, altrimenti si presumerà che sia parte della sillaba, e questo causerà un errore poco chiaro, vedi Sezione “Apparent error in ../ly/init.ly” in *Uso del Programma*.

Si notino le doppie parentesi ad angolo `<< ... >>` che circondano l’intero brano per indicare che la musica e il testo devono trovarsi in simultanea.

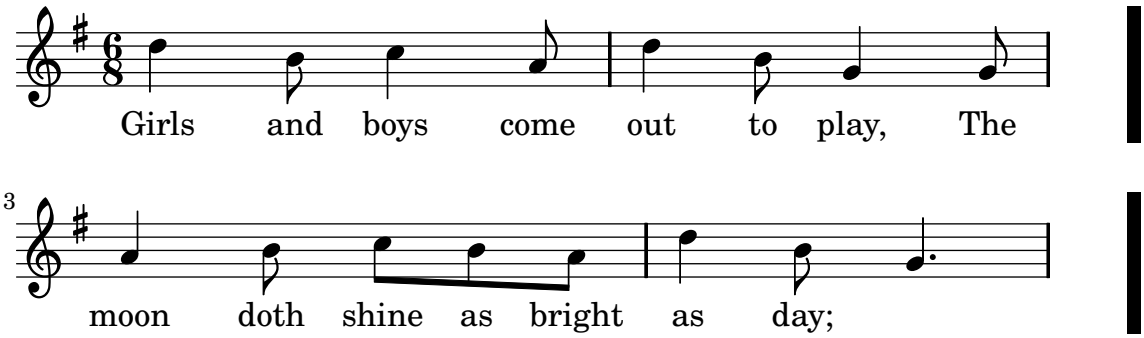
2.3.2 Allineare il testo alla melodia

Glossario musicale: Sezione “melisma” in *Glossario Musicale*, Sezione “linea di estensione” in *Glossario Musicale*.

La prossima riga della filastrocca è *The moon doth shine as bright as day*. Aggiungiamola:

```
<<
  \relative c'' {
    \key g \major
```

```
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c b a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```
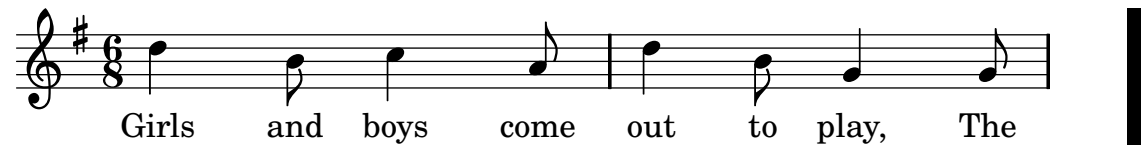


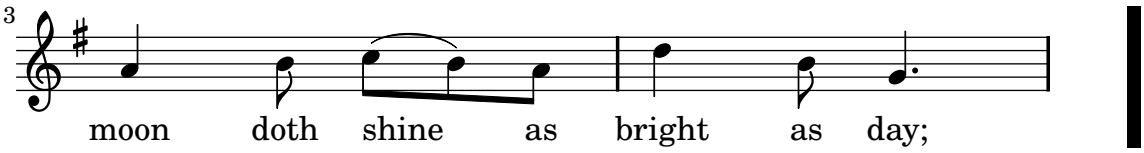
Se compili il codice dell'esempio precedente, dovresti vedere alcuni avvisi nell'output del terminale:

```
song.ly:12:29: warning: barcheck failed at: 5/8
    The | moon doth shine as
                              | bright as day; |
song.ly:12:46: warning: barcheck failed at: 3/8
    The | moon doth shine as | bright as day;
                                                |
```

Questo è un ottimo esempio dell'utilità dei controlli di battuta. Tornando alla musica, si può vedere che il testo aggiunto non risulta ben allineato alle note. La parola *shine* dovrebbe essere cantata su due note, non una. Questo si chiama *melisma*, una singola sillaba che viene cantata per più di una nota. Ci sono molti modi per estendere una sillaba su molteplici note, e il più semplice è aggiungere una legatura di portamento che le colleghi, per i dettagli si veda Sezione 2.1.3 [Legature di valore e di portamento], pagina 23:

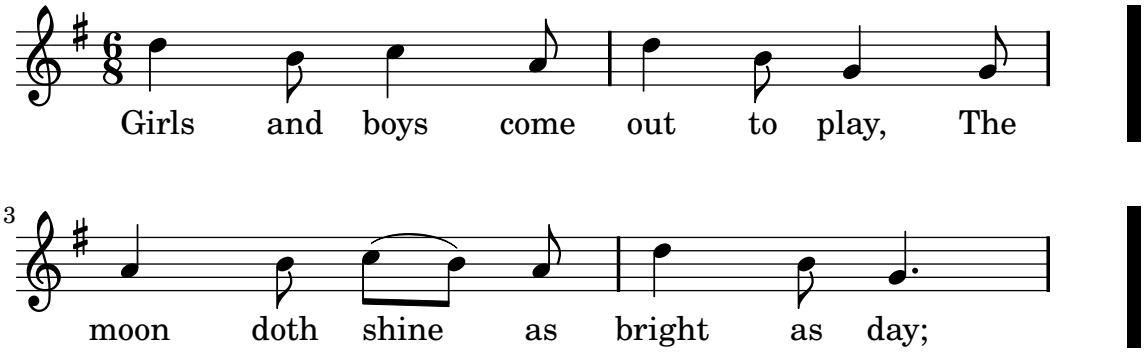
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c( b) a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```
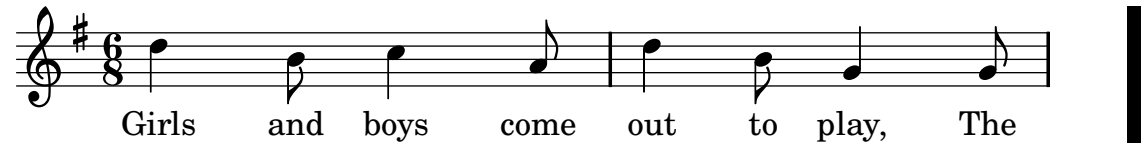




Adesso le parole sono allineate correttamente con le note, ma il raggruppamento automatico per le note che si trovano sopra *shine as* non ha un aspetto corretto. Possiamo correggerlo inserendo i comandi per il raggruppamento manuale così da scavalcare, in questo caso, il raggruppamento automatico; per i dettagli si veda Sezione 2.1.6 [Code automatiche e manuali], pagina 26.

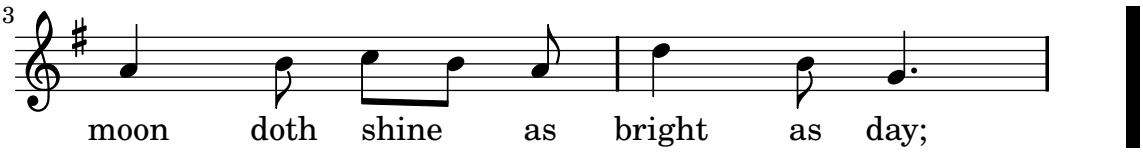
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c([ b]) a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine as | bright as day; |
  }
>>
```



Come alternativa all'uso delle legature di portamento, si possono indicare i melismi nel testo stesso, usando il trattino basso _ per ogni nota che si voglia includere nel melisma:
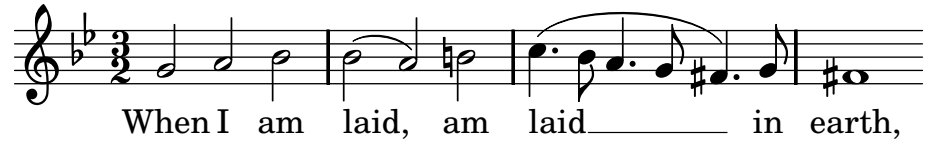
```
<<
  \relative c'' {
    \key g \major
    \time 6/8
    d4 b8 c4 a8 | d4 b8 g4 g8 |
    a4 b8 c[ b] a | d4 b8 g4. |
  }
  \addlyrics {
    Girls and boys come | out to play,
    The | moon doth shine _ as | bright as day; |
  }
>>
```
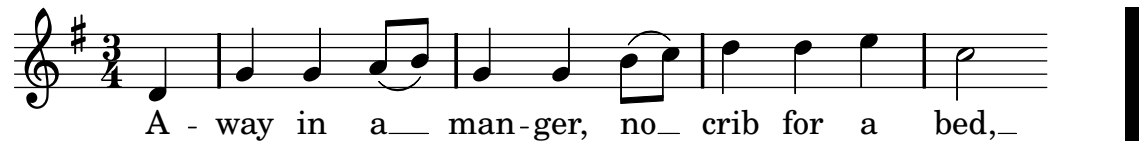




Se una sillaba si estende per molte note o per una singola nota molto lunga, solitamente viene disegnata una *linea di estensione* che va dalla sillaba e si estende per tutte le note comprese in quella sillaba. Si scrive con due trattini bassi `__`. Il seguente esempio è tratto dalle prime tre battute del *Lamento di Didone*, dal *Dido and Aeneas* di Purcell:

```
<<
  \relative c'' {
    \key g \minor
    \time 3/2
    g2 a bes | bes2( a) b2 |
    c4.( bes8 a4. g8 fis4.) g8 | fis1
  }
  \addlyrics {
    When I am | laid,
    am | laid __ in | earth,
  }
>>
```



Nessuno degli esempi visti finora ha comportato l'uso di parole aventi più di una sillaba. Tali parole vengono solitamente suddivise una sillaba per nota, con trattini posti tra le sillabe. Tali trattini vengono inseriti con due lineette, e producono un trattino centrato tra le sillabe. Ecco un esempio che, oltre a questo, mostra tutto quello che abbiamo imparato finora sull'allineamento del testo con le note.

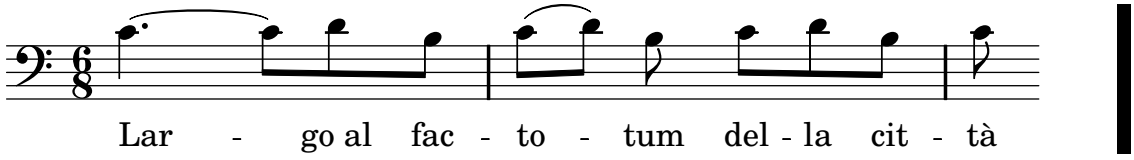
```
<<
  \relative c' {
    \key g \major
    \time 3/4
    \partial 4
    d4 | g4 g a8( b) | g4 g b8( c) |
    d4 d e | c2
  }
  \addlyrics {
    A -- | way in a __ | man -- ger,
    no __ | crib for a | bed, __
  }
>>
```



Alcuni testi, specialmente quelli in italiano, richiedono l'opposto: associare più di una sillaba a una singola nota. Questo può essere ottenuto collegando le sillabe insieme con un singolo trattino basso `_` (senza spazi), o includendole tra virgolette. Ecco un esempio tratto dal *Figaro*

di Rossini, dove *al* deve essere cantato sulla stessa nota del *go* di *Largo*, nell'aria di Figaro *Largo al factotum*:

```
<<
  \relative c' {
    \clef "bass"
    \key c \major
    \time 6/8
    c4.~ c8 d b | c8([ d]) b c d b | c8
  }
  \addlyrics {
    Lar -- go_al fac -- | to -- tum del -- la cit -- | tà
  }
>>
```
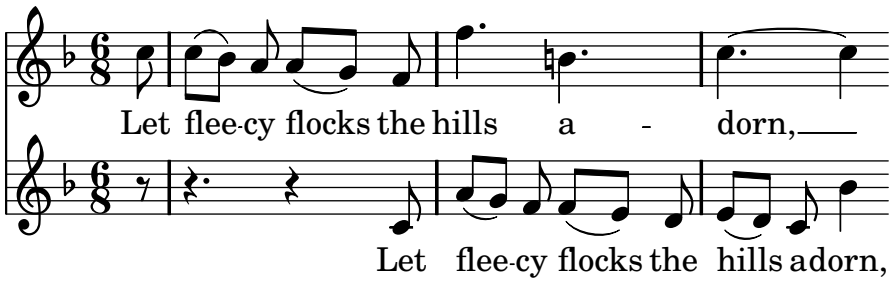


Vedi anche

Guida alla notazione: Sezione "Vocal music" in *Guida alla Notazione*.

2.3.3 Testo su più righi

Per porre lo stesso testo sotto più righi musicali, si può usare un approccio più semplice, ovvero `\addlyrics`. Ecco un esempio tratto dal *Judas Maccabæus* di Handel:

```
<<
  \relative c'' {
    \key f \major
    \time 6/8
    \partial 8
    c8 | c8([ bes]) a a([ g]) f | f'4. b, | c4.~ c4
  }
  \addlyrics {
    Let | flee -- cy flocks the | hills a -- | dorn, __
  }
  \relative c' {
    \key f \major
    \time 6/8
    \partial 8
    r8 | r4. r4 c8 | a'8([ g]) f f([ e]) d | e8([ d]) c bes'4
  }
  \addlyrics {
    Let | flee -- cy flocks the | hills a -- dorn,
  }
>>
```



Per scrivere partiture un po' più complesse di questi semplici esempi, è meglio separare la struttura dello spartito dalle note e dai testi mediante l'uso di variabili. Di queste si parla più approfonditamente in Sezione 2.4.1 [Organizzare i brani con le variabili], pagina 37.

Vedi anche

Guida alla notazione: Sezione "Vocal music" in *Guida alla Notazione*.

2.4 Ritocchi finali

Questa è la sezione finale della guida; spiega come dare gli ultimi ritocchi a semplici pezzi, e fornisce un'introduzione al resto del manuale.

2.4.1 Organizzare i brani con le variabili
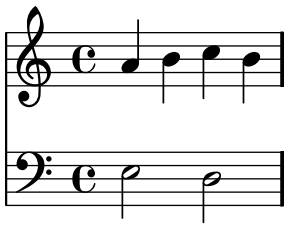
Quando tutti gli elementi discussi precedentemente vengono combinati insieme per produrre file di maggiori dimensioni, anche le espressioni musicali diventano molto più grandi. Nella musica polifonica con molti righi musicali, i file di input possono diventare molto confusi. Possiamo ridurre tale confusione attraverso l'uso delle *variabili*.

Con le variabili (conosciute anche come identificatori o macro), possiamo scomporre le espressioni musicali complesse. Una variabile viene assegnata nel seguente modo:

```
namedMusic = { ... }
```

I contenuti dell'espressione musicale `namedMusic` possono essere usati in seguito ponendo una barra inversa (backslash) di fronte al nome (`\namedMusic`, proprio come in un normale comando LilyPond).

```
violin = \new Staff {
  \relative c'' {
    a4 b c b
  }
}
cello = \new Staff {
  \relative c {
    \clef "bass"
    e2 d
  }
}
{
  <<
    \violin
    \cello
  >>
}
```
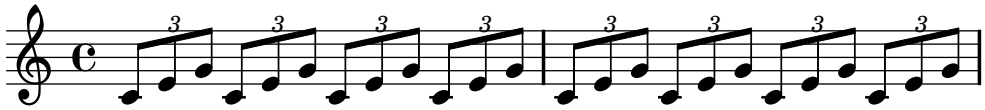


Il nome di una variabile deve contenere soltanto caratteri alfabetici, non può avere numeri, trattini bassi (underscore) o trattini di altro tipo.

Le variabili devono essere definite *prima* della principale espressione musicale, ma possono poi essere usate quante volte si vuole e ovunque, una volta definite. Possono essere usate anche all'interno della definizione di un'altra variabile successiva, dando la possibilità di accorciare l'input se una sezione della musica viene ripetuta molte volte.

```
tripletA = \times 2/3 { c,8 e g }
barA = { \tripletA \tripletA \tripletA \tripletA }

\relative c'' {
  \barA \barA
}
```



Le variabili possono essere utilizzate per molti tipi di oggetto nell'input. Ad esempio,

```
width = 4.5\cm
name = "Wendy"
aFivePaper = \paper { paperheight = 21.0 \cm }
```

A seconda dei suoi contesti, la variabile può essere usata in punti differenti. L'esempio seguente usa le variabili mostrate sopra:

```
\paper {
  \aFivePaper
  line-width = \width
}
{
  c4^\name
}
```

2.4.2 Aggiungere i titoli

Titolo, compositore, numero di opus e informazioni simili vengono inserite nel blocco `\header`. Questo si trova fuori dalla principale espressione musicale; il blocco `\header` viene solitamente posto sotto il numero di versione.

```
\version "2.14.2"
\header {
  title = "Symphony"
  composer = "Me"
  opus = "Op. 9"
}

{
  ... music ...
}
```
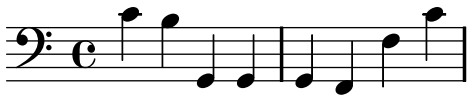
Quando il file viene elaborato, sopra la musica vengono visualizzati il titolo e il compositore. Si possono trovare maggiori informazioni sui titoli in Sezione “Creating titles” in *Guida alla Notazione*.

2.4.3 Nomi assoluti delle note

Finora abbiamo sempre usato `\relative` per definire le altezze. Questo è il modo più semplice per inserire gran parte della musica, ma esiste anche un altro modo per definire le altezze: il modo assoluto.

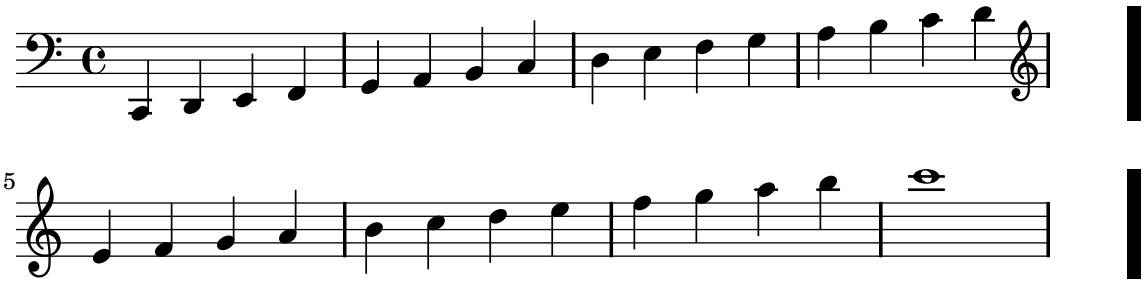
Se si omette `\relative`, LilyPond tratta tutte le altezze come valori assoluti. Un `c'` si riferirà sempre al Do centrale, un `b` si riferirà sempre alla nota che si trova un semitono sotto il Do centrale, e un `g,` indicherà sempre la nota sull’ultima riga della chiave di basso.

```
{
  \clef "bass"
  c'4 b g, g, |
  g,4 f, f c' |
}
```
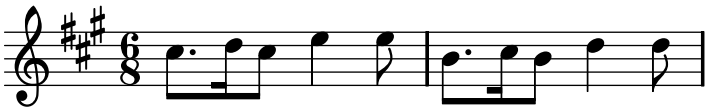


Ecco una scala con quattro ottave:

```
{
  \clef "bass"
  c,4 d, e, f, |
  g,4 a, b, c |
  d4 e f g |
  a4 b c' d' |
  \clef "treble"
  e'4 f' g' a' |
  b'4 c'' d'' e'' |
  f''4 g'' a'' b'' |
  c'''1 |
}
```
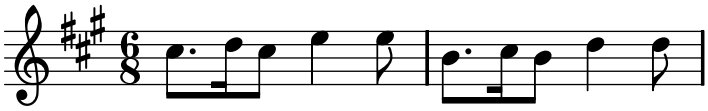


Come si può vedere, scrivere una melodia in chiave di violino richiede un ampio uso di virgolette `'`. Si consideri questo frammento tratto da Mozart:

```
{
  \key a \major
  \time 6/8
  cis''8. d''16 cis''8 e''4 e''8 |
  b'8. cis''16 b'8 d''4 d''8 |
}
```



Tutte queste virgolette rendono l'input meno leggibile e sono una fonte di errori. Usando `\relative`, l'esempio precedente è molto più semplice da leggere e scrivere:

```
\relative c'' {
  \key a \major
  \time 6/8
  cis8. d16 cis8 e4 e8
  b8. cis16 b8 d4 d8
}
```



Se si fa un errore con un segno di ottava (`'` o `,`) quando si lavora col modo `\relative`, è molto evidente – tante note saranno nell'ottava sbagliata. Quando si lavora in modo assoluto, un singolo errore non sarà tanto visibile, e non sarà così facile da trovare.

Tuttavia, il modo assoluto è utile per la musica che fa uso di ampi intervalli, e lo è ancora di più per i file LilyPond generati dal computer.

2.4.4 Dopo il tutorial

Dopo aver finito di leggere la guida, probabilmente dovresti cercare di scrivere uno o due brani. Puoi iniziare aggiungendo delle note a uno dei ⟨undefined⟩ [Templates], pagina ⟨undefined⟩. Se necessiti di una qualche notazione che non è stata coperta nel tutorial, dai un'occhiata alla Guida alla notazione, a cominciare da Sezione "Musical notation" in *Guida alla Notazione*. Se vuoi scrivere per un gruppo di strumenti che non è incluso nei template, dai un'occhiata a ⟨undefined⟩ [Extending the templates], pagina ⟨undefined⟩.

Una volta che hai scritto alcuni brevi pezzi, puoi proseguire la lettura del Manuale di Apprendimento (capitoli 3-5). Ovviamente non c'è niente di male nel leggerli subito! Però tieni conto che la parte restante del Manuale di Apprendimento parte dal presupposto che tu abbia già confidenza con l'input di LilyPond. Puoi saltare questi capitoli adesso, e tornare a leggerli quando ti sarai fatto un po' di esperienza.

In questo tutorial e nel resto del Manuale di Apprendimento, alla fine di ogni sezione c'è un paragrafo **Vedi anche**, che contiene riferimenti incrociati ad altre sezioni: non dovresti seguire questi riferimenti quando leggi il manuale per la prima volta; quando avrai completato la lettura di tutto il Manuale di Apprendimento, potrai rileggere alcune sezioni e seguire i riferimenti incrociati per letture di approfondimento.

Se non lo hai già fatto, *ti consigliamo* di leggere Sezione 1.4.3 [Panoramica dei manuali], pagina 20. Contiene molte informazioni su LilyPond, ed è quindi utile per i nuovi utenti, che spesso non sanno dove cercare aiuto. Se dedichi cinque minuti all'attenta lettura di quella sezione, potresti risparmiarti ore di frustrazione sprecate a guardare in posti sbagliati!

3 Concetti fondamentali

Nel Tutorial abbiamo visto come produrre dei belli spartiti da un semplice file di testo. Questa sezione presenta i concetti e le tecniche richiesti per produrre partiture ugualmente belle, ma più complesse.

3.1 Come funzionano i file di input di LilyPond

Il formato di input di LilyPond ha una struttura piuttosto libera, che dà agli utenti esperti una grande flessibilità nell'organizzare i file come preferiscono. Ma questa flessibilità può creare confusione nei nuovi utenti. Questa sezione spiegherà in parte questa struttura, ma sorvolerà su alcuni dettagli in favore della semplicità. Per una descrizione completa del formato di input, si veda Sezione "File structure" in *Guida alla Notazione*.

3.1.1 Introduzione alla struttura di un file di LilyPond

Un esempio basilare di un file di input di LilyPond è

```
\version "2.14.2"

\header { }

\score {
  ...espressione musicale composta...  % tutta la musica va qui!
  \layout { }
  \midi { }
}
```

Ci sono molte varianti a questo modello di base, ma questo esempio serve da utile punto di partenza.

Finora nessuno degli esempi che abbiamo visto ha usato il comando `\score{}`. Questo si spiega col fatto che LilyPond, quando elabora un input semplice, aggiunge automaticamente gli altri comandi necessari. LilyPond tratta un input come questo:

```
\relative c'' {
  c4 a d c
}
```

come forma abbreviata per questo:

```
\book {
  \score {
    \new Staff {
      \new Voice {
        \relative c'' {
          c4 a b c
        }
      }
    }
    \layout { }
  }
}
```

In altre parole, se l'input contiene un'espressione musicale singola, LilyPond interpreterà il file come se l'espressione musicale fosse racchiusa dentro i comandi mostrati sopra.

Attenzione! Molti esempi nella documentazione di LilyPond ometteranno i comandi `\new Staff` e `\new Voice`, lasciando che questi siano creati implicitamente. Per gli esempi semplici

questo metodo funziona bene, ma per quelli più complessi, soprattutto quando vengono usati ulteriori comandi, la creazione implicita dei contesti può dare risultati inattesi, ad esempio creando dei righi non voluti. Il modo per creare i contesti esplicitamente è spiegato in ⟨undefined⟩ [Contexts and engravers], pagina ⟨undefined⟩.

> **Nota:** Quando si inseriscono più di poche linee di musica, si consiglia di creare sempre esplicitamente i righi e le voci.

Ora però torniamo al primo esempio ed esaminiamo il comando `\score`, lasciando gli altri comandi secondo l'impostazione predefinita.

Un blocco `\score` deve sempre contenere una sola espressione musicale, e questa deve trovarsi subito dopo il comando `\score`. Ricorda che un'espressione musicale può essere qualsiasi cosa, da una singola nota a una grande espressione composta come

```
{
  \new StaffGroup <<
    ...inserisci qui l'intera opera di Wagner...
  >>
}
```

Tutto quanto è compreso in `{ ... }` costituisce un'unica espressione musicale.

Come abbiamo detto prima, il blocco `\score` può contenere altri elementi, come ad esempio

```
\score {
  { c'4 a b c' }
  \header { }
  \layout { }
  \midi { }
}
```

Si noti che questi tre comandi – `\header`, `\layout` e `\midi` – sono speciali: diversamente da molti altri comandi che iniziano con un backslash (\), *non* sono espressioni musicali né fanno parte di alcuna espressione musicale. Dunque, possono essere collocati dentro o fuori da un blocco `\score`. Di solito questi comandi vengono posti fuori dal blocco `\score` – ad esempio, `\header` spesso viene messo sopra il comando `\score`, come mostra l'esempio all'inizio di questa sezione.

Altri due comandi che non hai incontrato finora sono `\layout { }` e `\midi {}`. Se questi appaiono come in figura, LilyPond creerà rispettivamente un output per la stampa e uno per il MIDI. Sono descritti dettagliatamente nella Guida alla notazione, in Sezione "Score layout" in *Guida alla Notazione*, e Sezione "Creating MIDI files" in *Guida alla Notazione*.

Puoi scrivere molteplici blocchi `\score`. Ciascuno verrà trattato come una partitura separata, ma saranno tutti combinati in un unico file di output. Non è necessario il comando `\book` – ne verrà creato uno automaticamente. Tuttavia, se si desiderano file di output separati da un file '`.ly`', allora si deve usare il comando `\book` per separare le diverse sezioni: ogni blocco `\book` produrrà un file di output separato.

In breve:

Ogni blocco `\book` crea un file di output separato (ovvero, un file PDF). Se non ne hai aggiunto uno esplicitamente, LilyPond racchiude implicitamente tutto il tuo codice di input in un blocco `\book`.

Ogni blocco `\score` è un pezzo di musica separato all'interno di un blocco `\book`.

Ogni blocco `\layout` influenza il blocco `\score` o `\book` in cui compare – ovvero, un blocco `\layout` che si trova dentro un blocco `\score` riguarda solo quel blocco `\score`, mentre un blocco `\layout` che si trova fuori da un blocco `\score` (e quindi in un blocco `\book`, esplicitamente o implicitamente) riguraderà ogni `\score` in quel `\book`.

Per maggiori dettagli si veda Sezione “Multiple scores in a book” in *Guida alla Notazione*.

Un’altra grande scorciatoia è la possibilità di definire variabili, come è spiegato in Sezione 2.4.1 [Organizzare i brani con le variabili], pagina 37). Tutti i modelli usano questa forma

```
melodia = \relative c' {
  c4 a b c
}

\score {
  \melodia
}
```

Quando LilyPond esamina questo file, prende il valore di `melodia` (tutto ciò che si trova dopo il segno di uguale) e lo inserisce ovunque si trovi `\melodia`. Non c’è una regola specifica per i nomi – il nome può essere `melodia`, `globale`, `tempo`, `manodestrapiano`, o qualsiasi altro nome. Ricordati che puoi usare quasi ogni nome che vuoi, purché esso contenga solo caratteri alfabetici e sia diverso dai nomi dei comandi di LilyPond. Le esatte limitazioni relative ai nomi delle variabili sono spiegate dettagliatamente in Sezione “File structure” in *Guida alla Notazione*.

Vedi anche

Per una definizione completa del formato di input, si veda Sezione “File structure” in *Guida alla Notazione*.

3.1.2 La partitura è una (singola) espressione musicale composta

Abbiamo visto l’organizzazione generale dei file di input di LilyPond nella sezione precedente, Sezione 3.1.1 [Introduzione alla struttura di un file di LilyPond], pagina 41. Ma sembra che abbiamo saltato la parte più importante: cosa dobbiamo scrivere dopo `\score`?

In realtà non l’abbiamo affatto dimenticato. Il grande mistero è, semplicemente, che *non c’è* alcun mistero. La seguente frase spiega tutto:

> *Un blocco* `\score` *deve iniziare con un’espressione musicale composta.*

Per capire cosa si intende per espressione musicale e per espressione musicale composta, potrebbe esserti utile ripassare il tutorial, Sezione 2.2.1 [Espressioni musicali], pagina 27. In quella sezione, abbiamo visto come costruire grandi espressioni musicali a partire da piccoli brani – abbiamo iniziato con le note, poi gli accordi, etc. Adesso inzieremo da una grande espressione musicale e proseguiremo poi a spiegarne i dettagli. Per semplicità, nel nostro esempio useremo soltanto un canto e un pianoforte. Per questa formazione non abbiamo bisogno di `StaffGroup`, che non fa altro che raggruppare un insieme di righi con una parentesi graffa a sinistra, ma abbiamo comunque bisogno dei righi per il canto e per il pianoforte.

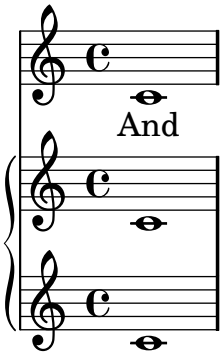
```
\score {
  <<
    \new Staff = "canto" <<
    >>
    \new PianoStaff = "pianoforte" <<
    >>
  >>
  \layout { }
}
```

In questo esempio abbiamo dato dei nomi ai righi – “canto” e “pianoforte”. Non è necessario in questo caso, ma è una buona abitudine da coltivare, perché ti permette di vedere a colpo d’occhio a cosa serve ciascun rigo.

Ricorda che si usano `<< ... >>` invece di `{ ... }` per indicare la musica simultanea. In questo modo la parte vocale e quella di pianoforte appaiono una sopra l’altra nello spartito. Il costrutto

`<< ... >>` non sarebbe necessario per il rigo del cantante nell'esempio precedente se contenesse soltanto un'espressione musicale sequenziale, ma `<< ... >>` al posto delle parentesi è necessario se la musica sul rigo deve contenere due o più espressioni simultanee, ad esempio due voci simultanee, o una voce con del testo. Vogliamo avere una voce con del testo, dunque ci servono le parentesi ad angolo. Aggiungeremo la musica vera e propria in seguito; per adesso mettiamo soltanto delle semplici note e del testo. Se hai dimenticato come aggiungere del testo, potresti voler ripassare `\addlyrics` in Sezione 2.3.1 [Impostare canzoni semplici], pagina 32.

```
\score {
  <<
    \new Staff = "canto" <<
      \new Voice = "vocal" { c'1 }
      \addlyrics { And }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "upper" { c'1 }
      \new Staff = "lower" { c'1 }
    >>
  >>
  \layout { }
}
```



Ora abbiamo molti più dettagli. Abbiamo il rigo del cantante: esso contiene una `Voice` o voce (in LilyPond, questo termine si riferisce a un insieme di note, non necessariamente alle note della voce – ad esempio, un violino di solito costituisce una voce) e del testo. Abbiamo anche il rigo del pianoforte, che a sua volta comprende un rigo superiore (per la mano destra) e uno inferiore (per la mano sinistra), sebbene a quest'ultimo debba ancora essere assegnata una chiave di basso.

A questo punto possiamo iniziare ad inserire le note. All'interno delle parentesi graffe vicine a `\new Voice = "vocal"`, possiamo iniziare a scrivere
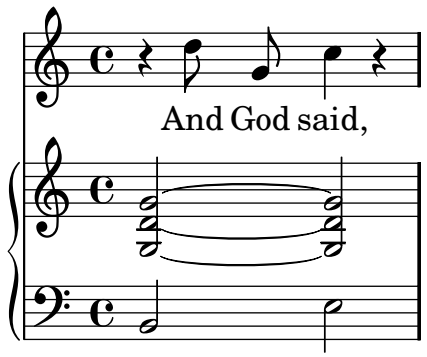
```
\relative c'' {
  r4 d8\noBeam g, c4 r
}
```

Ma se facessimo così, la sezione `\score` diventerebbe molto lunga, e sarebbe più difficile comprendere quel che accade. Usiamo quindi le variabili piuttosto. Queste sono state introdotte alla fine della sezione precedente, ricordi? Per far sì che i contenuti della variabile `text` siano interpretati come testo, li facciamo precedere da `\lyricmode`. Come in `\addlyrics`, questo comando trasforma la modalità di input in modalità testo. Senza di esso, LilyPond cercherebbe di interpretare i contenuti come se fossero note, e questo produrrebbe degli errori. (Sono disponibili molte altre modalità di input, si veda Sezione "Input modes" in *Guida alla Notazione*.)

Dunque se aggiungiamo un po' di note e una chiave di basso per la mano sinistra, otteniamo un brano musicale vero e proprio:

```
melody = \relative c'' { r4 d8\noBeam g, c4 r }
text   = \lyricmode { And God said, }
upper  = \relative c'' { <g d g,>2~ <g d g,> }
lower  = \relative c { b2 e }

\score {
  <<
    \new Staff = "canto" <<
      \new Voice = "vocal" { \melody }
      \addlyrics { \text }
    >>
    \new PianoStaff = "piano" <<
      \new Staff = "upper" { \upper }
      \new Staff = "lower" {
        \clef "bass"
        \lower
      }
    >>
  >>
  \layout { }
}
```



Quando scrivi (o leggi) una sezione `\score`, prenditela comoda e stai attento. Comincia dal livello più esterno, poi lavora su ogni livello più piccolo. È anche molto utile essere rigorosi nell'indentare l'input – ovvero fare attenzione che ogni elemento di uno stesso livello presente nell'editor di testo si trovi nella stessa posizione orizzontale.

Vedi anche
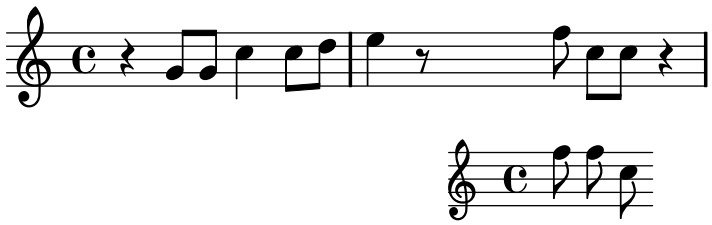
Guida alla notazione: Sezione "Structure of a score" in *Guida alla Notazione*.

3.1.3 Annidare le espressioni musicali

Non è obbligatorio dichiarare tutti i righi fin dall'inizio; possono essere invece introdotti temporaneamente in ogni momento. Questo è utile in particolare per creare le sezioni ossia – si veda Sezione "ossia" in *Glossario Musicale*. Ecco un semplice esempio che mostra come inserire un nuovo rigo temporaneamente, per la durata di tre note:

```
\new Staff {
  \relative g' {
    r4 g8 g c4 c8 d |
```
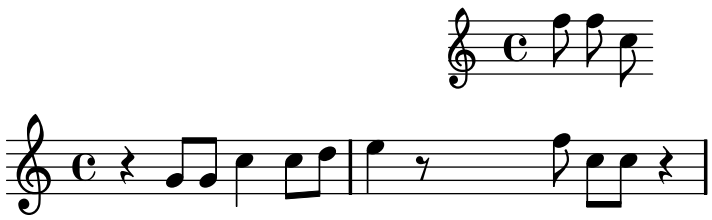
```
    e4 r8
    <<
      { f8 c c }
      \new Staff {
        f8 f c
      }
    >>
    r4 |
  }
}
```



Si noti che la dimensione della chiave è la stessa di una chiave che segue un cambio di chiave –ovvero leggermente più piccola della chiave all'inizio del rigo. Questo è utile per le chiavi che devono essere posizionate a metà di un rigo.

La sezione ossia può anche essere posta sopra il rigo nel seguente modo:

```
\new Staff = "main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
      } { f8 f c }
    >>
    r4 |
  }
}
```



Questo esempio usa `\with`, che verrà spiegato in modo più completo in seguito. È un modo per cambiare il comportamento predefinito di un singolo rigo. In questo caso, indica che il nuovo rigo deve essere posizionato sopra il rigo chiamato "main" invece che nella posizione predefinita, che è in basso.

Vedi anche

Gli ossia vengono spesso scritti senza armatura di chiave e senza tempo, e solitamente hanno un font più piccolo. Richiedono ulteriori comandi che non sono stati ancora presentati. Si veda Sezione 4.3.2 [Size of objects], pagina 102, e Sezione "Ossia staves" in *Guida alla Notazione*.

3.1.4 Sul non annidamento di parentesi e legature di valore

Abbiamo già incontrato vari tipi di parentesi e di costrutti che fanno uso di parentesi mentre scrivevamo il file di input di LilyPond. Ognuna obbedisce a diverse regole, e questo può generare confusione all'inizio. Rivediamo prima i diversi tipi di parentesi e di costrutti in parentesi.
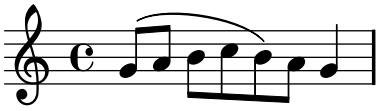
Tipo di parentesi	**Funzione**
`{ .. }`	Racchiude un frammento di musica sequenziale
`< .. >`	Racchiude le note di un accordo
`<< .. >>`	Racchiude le espressioni musicali simultanee
`( .. )`	Contrassegna l'inizio e la fine di una legatura di portamento
`\( .. \)`	Contrassegna l'inizio e la fine di una legatura di frase
`[ .. ]`	Contrassegna l'inizio e la fine di una travatura impostata manualmente

A questi dovremmo aggiungere altri costrutti che generano linee tra e lungo le note: legature di valore (contrassegnate dal tilde, `~`), i gruppi irregolari scritti in questa forma `\times x/y {..}`, e gli abbellimenti, indicati con `\grace{..}`.

Fuori da LilyPond, l'uso convenzionale delle parentesi esige che i diversi tipi siano annidati adeguatamente, in questo modo, `<< [ { ( .. ) } ] >>`, dove le parentesi che chiudono si trovano esattamente nell'ordine opposto alle parentesi che aprono. Questo **è** un requisito per i tre tipi di parentesi descritti dal termine 'Racchiude' nella tabella precedente – devono annidarsi correttamente. Tuttavia, i restanti costrutti in parentesi, descritti dal termine 'Contrassegna' nella tabella, così come le legature e i gruppi irregolari, **non** devono annidarsi correttamente con alcuna delle parentesi o dei costrutti in parentesi. Infatti queste non sono parentesi nel senso che racchiudono qualcosa – sono semplicemente dei contrassegni che indicano dove qualcosa inizia e finisce.

Quindi, ad esempio, una legatura di frase può iniziare prima di una travatura inserita manualmente e finire prima della fine della travatura – non molto musicale, forse, ma possibile:

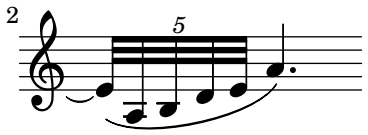
```
g8\( a b[ c b\) a] g4
```



In generale, tipi diversi di parentesi, costrutti in parentesi e segni che riguardano gruppi irregolari, legature e abbellimenti possono essere combinati liberamente. L'esempio seguente mostra una travatura che si estende su un gruppo irregolare (linea 1), una legatura di portamento che si estende su una terzina (linea 2), una travatura e una legatura di portamento che si estendono su una terzina, una legatura di valore che attraversa due gruppi irregolari, e una legatura di frase che si estende fuori da un gruppo irregolare (linee 3 e 4).

```
r16[ g \times 2/3 { r16 e'8] }
g,16( a \times 2/3 { b16 d) e }
g,8[( a \times 2/3 { b8 d) e~] } |
\times 4/5 { e32\( a, b d e } a4.\)
```





3.2 Le voci contengono la musica

Un cantante ha bisogno della voce per cantare, e lo stesso vale per LilyPond. La musica vera e propria per tutti gli strumenti di una partitura è contenuta nelle Voci – il più importante concetto di LilyPond.
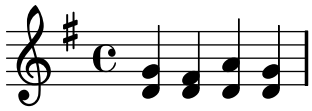
3.2.1 Sento le Voci

I livelli più profondi, più interni e più importanti di uno spartito di LilyPond sono chiamati 'Voice contexts' («Contesti della voce») o semplicemente 'Voices' («Voci»). In altri programmi di notazione le voci sono chiamate talvolta 'layers' («livelli»).

Il livello o contesto della voce è l'unico che può contenere la musica. Se un contesto della voce non è dichiarato esplicitamente, ne viene creato uno automaticamente, come abbiamo visto all'inizio di questo capitolo. Alcuni strumenti, come ad esempio un oboe, possono produrre una sola nota per volta. La musica scritta per tali strumenti è monofonica e necessita di una sola voce. Invece gli strumenti che possono produrre più di una nota contemporaneamente, come ad esempio il pianoforte, richiederanno spesso voci multiple per codificare le diverse note e ritmi simultanei che sono capaci di riprodurre.

Ovviamente, una singola voce può contenere molte note in un accordo, dunque quando l'uso delle voci multiple è davvero necessario? Si osservi questo esempio di quattro accordi:

```
\key g \major
<d g>4 <d fis> <d a'> <d g>
```
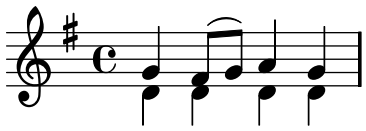


Questa musica può essere espressa usando soltanto i simboli dell'accordo, ovvero le parentesi angolari, `< ... >`, e una singola voce è sufficiente. Ma cosa accadrebbe se il Fa# fosse in realtà una nota di un ottavo seguita da un Sol di un ottavo, una nota di passaggio che porta al La? In questo caso abbiamo due note che iniziano nello stesso momento ma hanno durate diverse: il Re da un quarto e il Fa# da un ottavo. Come si possono scrivere queste note? Non possono essere scritte come un accordo perché tutte le note di un accordo devono avere la stessa durata. E non possono nemmeno essere scritte come due note in sequenza perché devono iniziare in contemporanea. Si tratta quindi di un caso in cui sono necessarie due voci.

Vediamo come ottenerle nella sintassi di input di LilyPond.

Il modo più semplice per inserire frammenti che utilizzino più di una voce su un rigo è scrivere ogni voce come una sequenza (con `{...}`), e poi combinarle in simultanea tramite le doppie parentesi angolari, `<<...>>`. Per collocarli in voci distinte, i frammenti devono essere separati da un doppio backslash, `\\`. Senza di esso, le note sarebbero inserite in un'unica voce, e questo normalmente causerebbe degli errori. Questa tecnica è particolarmente adatta ai brani che sono in gran parte monofonici ma con brevi e occasionali sezioni polifoniche.

Ecco come suddividere gli accordi precedenti in due voci e aggiungere sia la nota di passaggio che la legatura di portamento:
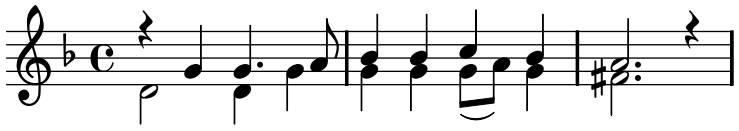
```
\key g \major
%    Voice "1"                Voice "2"
<< { g4 fis8( g) a4 g } \\ { d4 d d d }  >>
```



Si noti come i gambi della seconda voce adesso siano rivolti in basso.
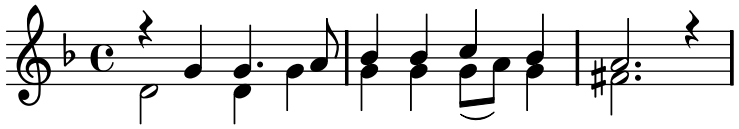
Ecco un altro semplice esempio:

```
\key d \minor
%    Voice "1"              Voice "2"
<< { r4 g g4. a8 }    \\ { d,2 d4 g }       >> |
<< { bes4 bes c bes } \\ { g4 g g8( a) g4 } >> |
<< { a2. r4 }         \\ { fis2. s4 }       >> |
```



Non è necessario usare un costrutto con << \\ >> in ogni battuta. Per musiche che hanno poche note in ogni battuta questo layout può aiutare la leggibilità del codice, ma se ci sono molte note in ogni battuta è preferibile dividere ogni voce, così:

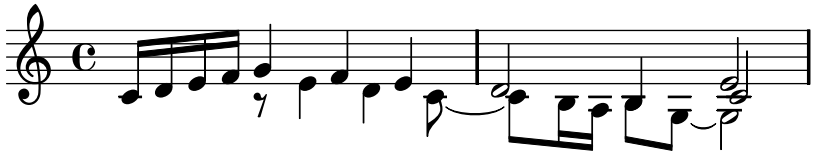
```
\key d \minor
<< {
  % Voice "1"
  r4 g g4. a8 |
  bes4 bes c bes |
  a2. r4 |
} \\ {
  % Voice "2"
  d,2 d4 g |
  g4 g g8( a) g4 |
  fis2. s4 |
} >>
```



Questo esempio ha solo due voci, ma si potrebbe usare lo stesso costrutto per scrivere tre o più voci aggiungendo più backslash.

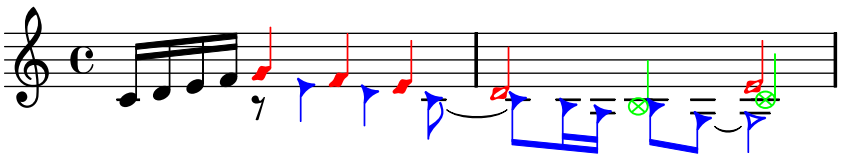
I contesti della voce hanno i nomi "1", "2", etc. In ogni contesto, la direzione verticale di legature di portamento, gambi, legature di valore, dinamica, etc., è impostata correttamente.

```
\new Staff \relative c' {
  % Main voice
  c16 d e f
  %    Voice "1"     Voice "2"                Voice "3"
  << { g4 f e } \\ { r8 e4 d c8~ } >> |
  << { d2 e }   \\ { c8 b16 a b8 g~ g2 } \\ { s4 b c2 } >> |
}
```



Tutte queste voci sono separate dalla voce principale che contiene le note e che si trova fuori dal costrutto `<< .. >>`, che chiameremo il *costrutto simultaneo*. Legature di portamento e di valore possono connettere solo note che fanno parte della stessa voce, quindi le legature non possono entrare in un costrutto simultaneo o uscirne. Viceversa, voci parallele appartenenti a costrutti simultanei distinti sullo stesso rigo sono la stessa voce. Anche altre caratteristiche della voce di riferimento sono trasferite ai costrutti simultanei. Ecco lo stesso esempio, ma con colori e teste delle note diversi per ogni voce. Si noti che i cambiamenti in una voce non interessano le altre voci, ma persistono sulla stessa voce in seguito. Si noti anche che le note legate possono essere divise sulle stesse voci in due costrutti, come mostra qui la voce con i triangoli blu.

```
\new Staff \relative c' {
  % Main voice
  c16 d e f
  <<  % Bar 1
    {
      \voiceOneStyle
      g4 f e
    }
  \\
    {
      \voiceTwoStyle
      r8 e4 d c8~
    }
  >> |
  <<  % Bar 2
     % Voice 1 continues
    { d2 e }
  \\
     % Voice 2 continues
    { c8 b16 a b8 g~ g2 }
  \\
    {
      \voiceThreeStyle
      s4 b c2
    }
  >> |
}
```



I comandi `\voiceXXXStyle` sono pensati soprattutto per documenti didattici come questo. Modificano il colore della testa, del gambo e delle travature, e lo stile della testa, così da rendere le voci facilmente distinguibili. La voce uno è impostata su rombi rossi, la voce due su triangoli blu, la voce tre su cerchi barrati verdi, e la voce quattro (non impiegata nell'esempio) su croci magenta; `\voiceNeutralStyle` (anch'esso non usato qui) riporta lo stile all'impostazione predefinita. Vedremo in seguito come l'utente possa creare comandi simili. Si veda Sezione 4.3.1 [Visibility and color of objects], pagina 98 e Sezione 4.6.2 [Using variables for tweaks], pagina 133.

La polifonia non cambia le relazioni tra le note all'interno di un blocco `\relative`. L'altezza di ogni nota continua a essere calcolata in rapporto a quella della nota che la precede, o della prima nota del precedente accordo. Dunque, in
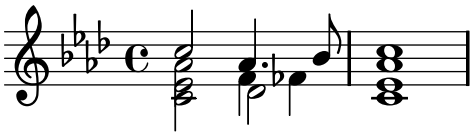
```
\relative c' { notaA << < notaB notaC > \\ notaD >> notaE }
```

`notaB` è relativa a `notaA`
`notaC` è relativa a `notaB`, non a `notaA`;
`notaD` è relativa a `notaB`, non a `notaA` o a `notaC`;
`notaE` è relativa a `notaD`, non a `notaA`.

Un metodo alternativo, che potrebbe essere più chiaro se le note nelle voci sono ampiamente separate, consiste nel porre un comando `\relative` all'inizio di ogni voce:

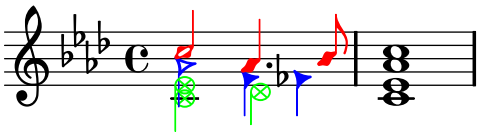
```
\relative c' { noteA ... }
<<
  \relative c'' { < noteB noteC > ... }
\\
  \relative g' { noteD ... }
>>
\relative c' { noteE ... }
```

Analizziamo infine le voci in un brano più complesso. Queste note sono tratte dalle prime due battute del secondo dei due Notturni op. 32 di Chopin. Questo esempio verrà usato successivamente, in questo e nel prossimo capitolo, per illustrare varie tecniche di notazione, quindi per il momento ignora qualsiasi cosa del codice sottostante che ti sembra incomprensibile e concentrati solo sulla musica e sulle voci – le parti più complesse saranno spiegate tutte in sezioni successive.



La direzione dei gambi viene spesso usata per indicare la continuità di due linee melodiche simultanee. In questo esempio i gambi delle note più acute sono tutti rivolti in su mentre i gambi delle note più gravi sono tutti rivolti in giù. Questo è il primo indizio del fatto che è coinvolta più di una voce.
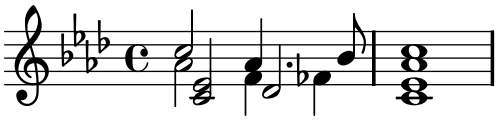
Ma è quando note che iniziano nello stesso momento hanno durate diverse che il ricorso a voci multiple diventa realmente indispensabile. Osserva le note che iniziano alla terza pulsazione della prima battuta. Il La bemolle è una nota di tre ottavi, il Fa è una semiminima e il Re bemolle è una minima. Non possono essere scritte come un accordo perché tutte le note di un accordo devono avere la stessa durata. Né possono essere scritte come note in sequenza, dato che devono iniziare contemporaneamente. Questa sezione della battuta necessita di tre voci, e la normale pratica consiste nello scrivere l'intera battuta su tre voci, come mostrato sotto, dove abbiamo usato diverse teste e colori per le tre voci. Ancora una volta, il codice che sta dietro questo esempio verrà spiegato dopo, quindi ignora quel che non capisci.



Proviamo a scrivere il codice di questa musica da zero. Come vedremo, questo pone alcune difficoltà. Come abbiamo imparato, iniziamo usando il costrutto `<< \\ >>` per inserire la musica della prima battuta in tre voci:
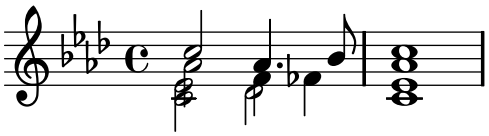
```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 } \\ { aes2 f4 fes } \\ { <ees c>2 des2 }
```

```
  >>
  <c ees aes c>1
}
```



La direzione dei gambi è assegnata automaticamente: le voci dispari avranno i gambi in su e le voci pari i gambi in giù. I gambi per le voci 1 e 2 sono giusti, ma in questo brano i gambi della voce 3 dovrebbero essere in giù. Possiamo correggere semplicemente omettendo la voce tre e ponendo la musica nella voce quattro. Si può fare aggiungendo semplicemente un altro paio di \\:

```
\new Staff \relative c'' {
  \key aes \major
  << % Voice one
    { c2 aes4. bes8 }
  \\ % Voice two
    { aes2 f4 fes   }
  \\ % Omit Voice three
  \\ % Voice four
    { <ees c>2 des2 }
  >> |
  <c ees aes c>1 |
}
```



Possiamo vedere che questo corregge la direzione del gambo, ma espone a un problema talvolta presente con le voci multiple: i gambi delle note in una voce possono collidere con le teste delle note di un'altra voce. Nel disporre le note, LilyPond permette alle note o agli accordi di due diverse voci di occupare la stessa colonna verticale della nota purché i gambi siano in direzioni opposte, ma le note della terza e quarta voce vengono spostate, se necessario, per evitare la collisione tra le teste. Questo di solito funziona bene, ma in questo esempio le note della voce più bassa, con le impostazioni predefinite, appaiono chiaramente disposte in una posizione non ottimale. LilyPond fornisce molti modi per aggiustare la collocazione orizzontale delle note. Per ora, non siamo ancora pronti a cercare di correggere questo problema, dunque dovremo aspettare fino a una prossima sezione – si veda la proprietà `force-hshift` in Sezione 4.5.2 [Fixing overlapping notation], pagina 118.

Vedi anche

Guida alla notazione: Sezione "Multiple voices" in *Guida alla Notazione*.

3.2.2 Definire esplicitamente le voci

I contesti della voce possono anche essere creati manualmente, all'interno di un blocco `<< >>` che crea musica polifonica, usando `\voiceOne` ... `\voiceFour` per indicare le direzioni desiderate per gambi, legature, etc. Nelle partiture più grandi questo metodo è più chiaro, perché fa sì che le voci possano essere separate e nominate in modo più descrittivo.

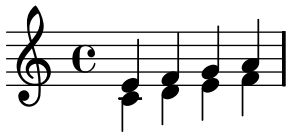
Nello specifico, il costrutto `<< \\ >>` usato nella sezione precedente:

```
\new Staff {
  \relative c' {
    << { e4 f g a } \\ { c,4 d e f } >>
  }
}
```

è equivalente a

```
\new Staff <<
  \new Voice = "1" { \voiceOne \relative c' { e4 f g a } }
  \new Voice = "2" { \voiceTwo \relative c' { c4 d e f } }
>>
```
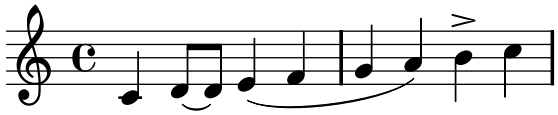
Entrambi hanno come risultato
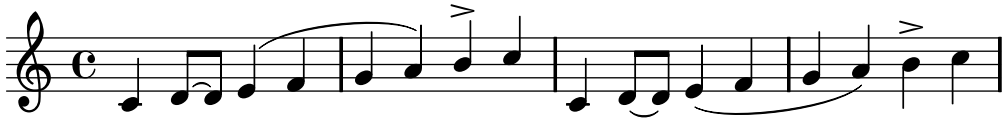


I comandi `\voiceXXX` impostano le direzioni di gambi, legature di portamento, legature di valore, articolazioni, annotazioni, punti di aumentazione di note puntate e diteggiature. `\voiceOne` e `\voiceThree` fanno sì che questi oggetti siano rivolti verso l'alto, mentre `\voiceTwo` e `\voiceFour` fanno sì che puntino verso il basso. Questi comandi producono anche uno spostamento orizzontale per ogni voce quando si crei la necessità di evitare collisioni tra le teste. Il comando `\oneVoice` riporta i valori alle normali impostazioni di una singola voce.

Vediamo tramite alcuni semplici esempi quali effetti esattamente `\oneVoice`, `\voiceOne` e `voiceTwo` hanno su markup, legature di valore, legature di portamento, e dinamica:
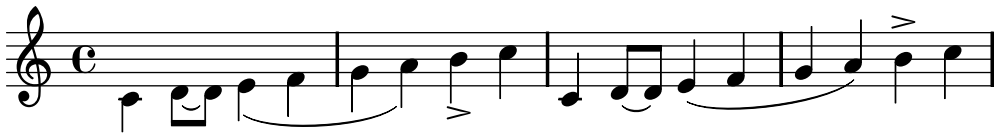
```
\relative c'{
  % Default behavior or behavior after \oneVoice
  c4 d8~ d e4( f | g4 a) b-> c |
}
```



```
\relative c' {
  \voiceOne
  c4 d8~ d e4( f | g4 a) b-> c |
  \oneVoice
  c,4 d8~ d e4( f | g4 a) b-> c |
}
```



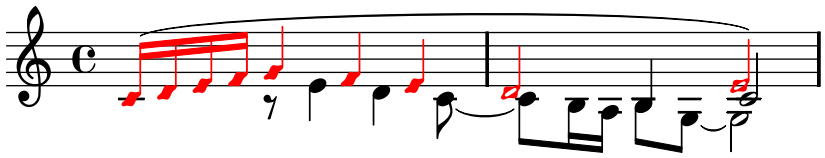
```
\relative c' {
  \voiceTwo
  c4 d8~ d e4( f | g4 a) b-> c |
  \oneVoice
  c,4 d8~ d e4( f | g4 a) b-> c |
}
```



Vediamo adesso, usando l'esempio della sezione precedente, tre modi diversi di scrivere uno stesso passo di musica polifonica, e i rispettivi vantaggi, a seconda delle circostanze.

Un'espressione che appare direttamente in un << >> appartiene alla voce principale (ma, attenzione: **non** in un costrutto << \\ >>). Questo metodo è utile quando le altre voci entrano mentre la voce principale sta già suonando. Ecco una versione più corretta del nostro esempio. Le note a rombi rossi mostrano che la melodia principale si trova ora nel contesto di una voce singola, e questo fa sì che sia possibile disegnare una legatura di frase sopra di esse.

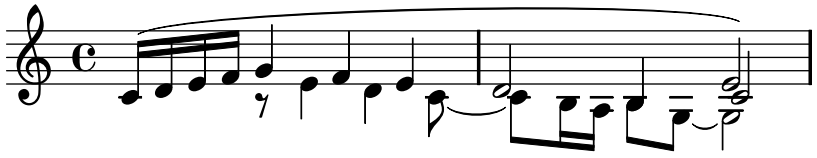
```
\new Staff \relative c' {
  \voiceOneStyle
  % The following notes are monophonic
  c16^( d e f
  % Start simultaneous section of three voices
  <<
    % Continue the main voice in parallel
    { g4 f e | d2 e) | }
    % Initiate second voice
    \new Voice {
      % Set stems, etc., down
      \voiceTwo
      r8 e4 d c8~ | c8 b16 a b8 g~ g2 |
    }
    % Initiate third voice
    \new Voice {
      % Set stems, etc, up
      \voiceThree
      s2. | s4 b c2 |
    }
  >>
}
```



Sono possibili costrutti polifonici annidati più fittamente, e se una voce appare solo brevemente questo potrebbe essere un modo più semplice di scrivere lo spartito:

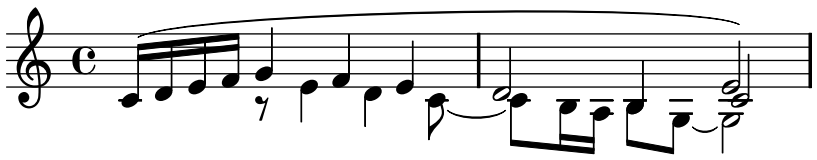
```
\new Staff \relative c' {
  c16^( d e f
  <<
    { g4 f e | d2 e) | }
    \new Voice {
      \voiceTwo
      r8 e4 d c8~ |
      <<
        { c8 b16 a b8 g~ g2 | }
        \new Voice {
          \voiceThree
          s4 b c2 |
```

```
        }
      >>
    }
  >>
}
```



Questo metodo di annidare nuove voci in breve è utile quando solo piccole parti della musica sono polifoniche, ma quando la polifonia è impiegata largamente in tutta la parte può essere più chiaro ricorrere a voci multiple, usando le note spaziatrici per saltare le parti in cui una delle voci è muta, come nel seguente esempio:

```
\new Staff \relative c' <<
  % Initiate first voice
  \new Voice {
    \voiceOne
    c16^( d e f g4 f e | d2 e) |
  }
  % Initiate second voice
  \new Voice {
    % Set stems, etc, down
    \voiceTwo
    s4 r8 e4 d c8~ | c8 b16 a b8 g~ g2 |
  }
  % Initiate third voice
  \new Voice {
    % Set stems, etc, up
    \voiceThree
    s1 | s4 b c2 |
  }
>>
```



Note columns

Le note maggiormente ravvicinate all'interno di un accordo o quelle che compaiono nello stesso momento in voci diverse sono disposte in due, e talvolta più, colonne, per impedire la sovrapposizione delle teste. Esse vengono chiamate colonne delle note. Le singole colonne di ognuna delle voci sono indipendenti, e lo scarto adottato nella voce in uso è determinato dal distanziamento delle colonne nei casi in cui altrimenti si determinerebbe una collisione. Si può vedere nell'esempio in basso. Nella seconda battuta il Do della seconda voce è spostato a destra del Re nella prima voce, e nell'ultimo accordo il Do nella terza voce è spostato anch'esso a destra delle altre note.

I comandi `\shiftOn`, `\shiftOnn`, `\shiftOnnn`, e `\shiftOff` specificano il grado con cui le note e gli accordi della voce debbano essere spostati in caso di collisione. L'impostazione predefinita

prevede che le voci esterne (di norma le voci uno e due) abbiano `\shiftOff` attivato, mentre le voci interne (terza e quarta) abbiano `\shiftOn` attivato. Quando uno spostamento viene applicato, le voci una e tre vengono spostate a destra e le voci due e quattro a sinistra.

`\shiftOnn` e `\shiftOnnn` definiscono ulteriori livelli di scarto che possono essere temporaneamente specificati per risolvere le collisioni in situazioni complesse – si veda Sezione 4.5.3 [Real music example], pagina 123.

Una colonna di note può contenere soltanto una nota (o accordo) di una voce con gambi in su e una nota (o accordo) di una voce con gambi in giù. Se note di due voci che hanno i gambi nella stessa direzione sono poste nella stessa posizione ed entrambe le voci non hanno uno spostamento specificato oppure ne hanno uno dello stesso tipo, si produrrà il messaggio di errore "Too many clashing note columns".

Vedi anche
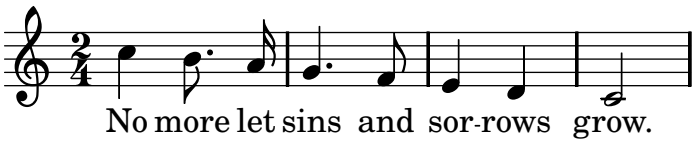
Guida alla notazione: Sezione "Multiple voices" in *Guida alla Notazione*.

3.2.3 Voci e musica vocale

La musica vocale presenta una difficoltà in più: occorre combinare due espressioni – note e testo.

Abbiamo già visto il comando `\addlyrics{}`, che ben si comporta con le partiture semplici. Tuttavia, questa tecnica è piuttosto limitata. Per musica più complessa, occorre introdurre il testo in un contesto `Lyrics` usando `\new Lyrics` e collegando esplicitamente il testo alle note con `\lyricsto{}`, tramite il nome assegnato alla voce.

```
<<
  \new Voice = "one" {
    \relative c'' {
      \autoBeamOff
      \time 2/4
      c4 b8. a16 | g4. f8 | e4 d | c2 |
    }
  }
  \new Lyrics \lyricsto "one" {
    No more let | sins and | sor -- rows | grow. |
  }
>>
```



Si noti che il testo deve essere collegato a un contesto `Voice`, *non* a un contesto `Staff`. Questo è un caso in cui è necessario creare esplicitamente contesti `Staff` e `Voice`.
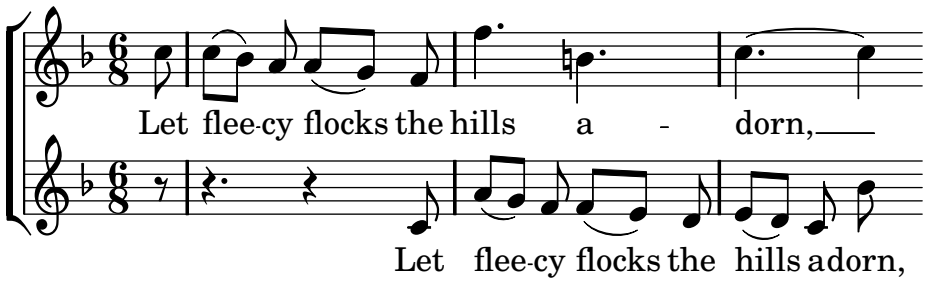
La disposizione automatica delle travature predefinita di LilyPond funziona bene per la musica strumentale, ma non altrettanto per la musica con testi, dove le travature o non sono usate affatto o servono a indicare la presenza di melismi nel testo. Nell'esempio precedente usiamo il comando `\autoBeamOff` per disattivare la travatura automatica.

Riprendiamo l'esempio precedente di Judas Maccabæus per presentare questa tecnica più flessibile. Innanzitutto lo rimaneggiamo per usare delle variabili per mezzo delle quali la musica e il testo possano essere separate dalla struttura del rigo. Inseriamo anche una parentesi ChoirStaff. Il testo deve essere introdotto da `\lyricmode` per assicurare che siano interpretati come testo invece che come musica.

```
global = { \key f \major \time 6/8 \partial 8 }

SopOneMusic = \relative c'' {
  c8 | c8([ bes)] a a([ g)] f | f'4. b, | c4.~ c4
}
SopOneLyrics = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn, __
}
SopTwoMusic = \relative c' {
  r8 | r4. r4 c8 | a'8([ g)] f f([ e)] d | e8([ d)] c bes'
}
SopTwoLyrics = \lyricmode {
  Let | flee -- cy flocks the | hills a -- dorn,
}

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "SopOne" {
        \global
        \SopOneMusic
      }
      \new Lyrics \lyricsto "SopOne" {
        \SopOneLyrics
      }
    >>
    \new Staff <<
      \new Voice = "SopTwo" {
        \global
        \SopTwoMusic
      }
      \new Lyrics \lyricsto "SopTwo" {
        \SopTwoLyrics
      }
    >>
  >>
}
```



Questa è la struttura di base di tutte le partiture vocali. Si possono aggiungere più righi, più voci in ogni rigo, più versi nei testi, e le variabili contenenti la musica possono essere poste in file separati se dovessero diventare troppo lunghe.

Ecco un esempio della prima linea di un inno con quattro strofe, impostate su SATB. In questo caso le parole per tutte e quattro le parti sono le stesse. Si noti l'uso delle variabili per separare

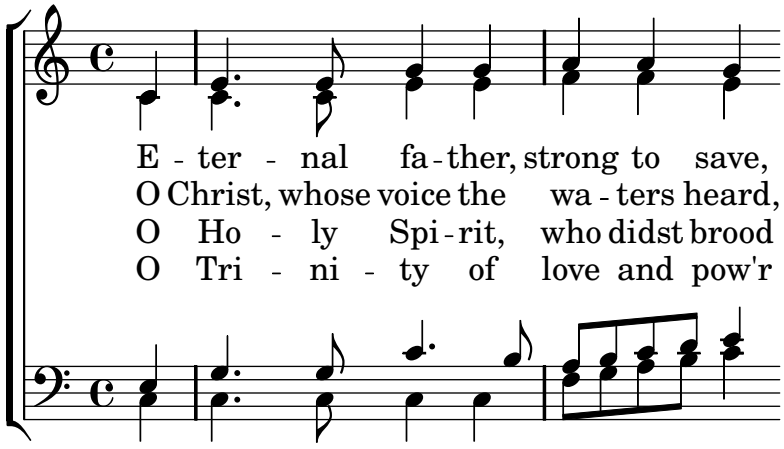
la notazione musicale e le parole dalla struttura del rigo. Si veda anche come una variabile, che abbiamo deciso di chiamare 'keyTime', venga usata per avere vari comandi a disposizione all'interno dei due righi. In altri esempi questo viene spesso chiamato 'global'.

```
keyTime = { \key c \major \time 4/4 \partial 4 }

SopMusic   = \relative c' { c4 | e4. e8 g4  g    | a4   a   g  }
AltoMusic  = \relative c' { c4 | c4. c8 e4  e    | f4   f   e  }
TenorMusic = \relative c  { e4 | g4. g8 c4.   b8 | a8 b c d e4 }
BassMusic  = \relative c  { c4 | c4. c8 c4  c    | f8 g a b c4 }

VerseOne =
  \lyricmode { E -- | ter -- nal fa -- ther, | strong to save, }
VerseTwo   =
  \lyricmode { O | Christ, whose voice the | wa -- ters heard, }
VerseThree =
  \lyricmode { O | Ho -- ly Spi -- rit, | who didst brood }
VerseFour  =
  \lyricmode { O | Tri -- ni -- ty of | love and pow'r }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Sop"  { \voiceOne \keyTime \SopMusic }
      \new Voice = "Alto" { \voiceTwo \AltoMusic }
      \new Lyrics \lyricsto "Sop" { \VerseOne   }
      \new Lyrics \lyricsto "Sop" { \VerseTwo   }
      \new Lyrics \lyricsto "Sop" { \VerseThree }
      \new Lyrics \lyricsto "Sop" { \VerseFour  }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \keyTime \TenorMusic }
      \new Voice = "Bass"  { \voiceTwo \BassMusic }
    >>
  >>
}
```



Vedi anche

Guida alla notazione: Sezione “Vocal music” in *Guida alla Notazione*.
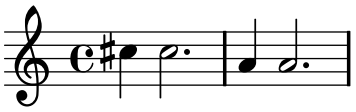
3.3 Contesti e incisori

I contesti e gli incisori sono stati menzionati in modo informale nelle sezioni precedenti; ora dobbiamo approfondire questi concetti, perché sono importanti nell’ottimizzazione dell’output di LilyPond.

3.3.1 I contesti

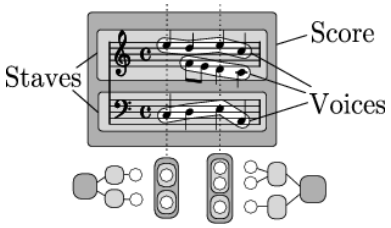
Quando la musica viene elaborata, molti elementi notazionali che non compaiono esplicitamente nel file di input devono essere aggiunti nell’output. Ad esempio, si confrontino l’input e l’output del seguente esempio:

```
cis4 cis2. | a4 a2. |
```



L’input è alquanto essenziale, ma nell’output sono stati aggiunti stanghette, alterazioni, l’armatura di chiave e il tempo. Quando LilyPond *interpreta* l’input l’informazione musicale viene analizzata da sinistra a destra, proprio come un musicista legge uno spartito. Mentre legge l’input, il programma ricorda dove si trovano i confini della misura, e quali altezze richiedono espliciti accidenti. Questa informazione deve essere conservata a diversi livelli. Ad esempio, un accidente influisce solo su un singolo rigo, mentre una stanghetta deve essere sincronizzata lungo l’intera partitura.

All’interno di LilyPond, queste regole e pezzi di informazione sono raggruppati nei *Contesti*. Abbiamo già presentato il contesto `Voice`. Altri contesti sono `Staff` e `Score`. I contesti sono strutturati gerarchicicamente per riflettere la natura gerarchica di una partitura musicale. Ad esempio: un contesto `Staff` può contenere molti contesti `Voice`, e un contesto `Score` può contenere molti contesti `Staff`.



Ogni contesto è responsabile di far rispettare alcune regole di notazione, creare alcuni oggetti della notazione e conservare le proprietà associate. Ad esempio, il contesto `Voice` può introdurre un’alterazione e poi il contesto `Staff` mantiene la regola per mostrare o sopprimere l’alterazione per il resto della misura.

Un altro esempio: la sincronizzazione delle stanghette è gestita, per impostazione predefinita, nel contesto `Score`. Tuttavia, in alcune forme musicali potremmo non volere che le stanghette siano sincronizzate – si consideri una partitura polimetrica in 4/4 e 3/4. In questi casi, dobbiamo modificare le impostazioni predefinite dei contesti `Score` e `Staff`.

Per spartiti molto semplici, i contesti vengono creati implicitamente, e non è necessario occuparsene. Ma per brani più ampi, come qualsiasi cosa abbia più di un rigo, devono essere creati esplicitamente per essere sicuri di avere tutti i righi necessari, e che questi siano nel giusto ordine. Per scrivere brani che utilizzano una notazione speciale, di solito si modificano i contesti esistenti, o addirittura se ne creano di completamente nuovi.

Oltre ai contesti `Score`, `Staff` e `Voice` ci sono contesti che stanno tra i livelli della partitura (score) e del rigo (staff) per controllare i gruppi di pentagrammi, come i contesti `PianoStaff` e `ChoirStaff`. Ci sono anche contesti alternativi per il rigo e la voce, e contesti per il testo, le percussioni, la tastiera, il basso continuo, etc.

I nomi di tutti i tipi di contesto sono formati da una o più parole, e ogni parola viene unita immediatamente alla parola precedente senza trattini o underscore bensì con la prima lettera maiuscola: ad esempio, `GregorianTranscriptionStaff`.

Vedi anche

Guida alla notazione: Sezione "Contexts explained" in *Guida alla Notazione*.

3.3.2 Creare i contesti

In un file di input un blocco della partitura, introdotto dal comando `\score`, contiene un'espressione musicale singola e una definizione di output associata (o un blocco `\layout` o un blocco `\midi`). Di solito si lascia che il contesto `Score` sia creato automaticamente quando inizia l'interpretazione di quell'espressione musicale.

Per le partiture che hanno una sola voce e un solo rigo, si può lasciare che i contesti `Voice` e `Staff` siano creati automaticamente, ma per le partiture più complesse è necessario crearli manualmente. Il comando più semplice per farlo è `\new`. Viene posto prima di un'espressione musicale, ad esempio

```
\new tipo espressione-musicale
```

dove *tipo* è il nome di un contesto (come `Staff` o `Voice`). Questo comando crea un nuovo contesto, e inizia a interpretare *espressione-musicale* all'interno di quel contesto.

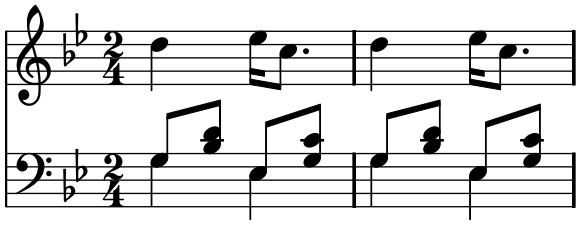
(Si noti che normalmente non è necessario il comando `\new Score`, perché il fondamentale contesto di livello superiore `Score` viene creato automaticamente quando l'espressione musicale all'interno del blocco `\score` viene interpretata. L'unica ragione per creare un contesto `Score` esplicitamente con `\new Score` è di inserire un blocco `\with` dove si possono specificare uno o più valori delle proprietà di contesto, predefiniti per tutto lo spartito.. Le informazioni su come usare i blocchi `\with` si trovano sotto il titolo "Setting context properties with `\\with`" in ⟨undefined⟩ [Modifying context properties], pagina ⟨undefined⟩.)

> **Nota:** `\new Score` non dovrebbe essere usato perché il fondamentale contesto di livello superiore `Score` viene creato automaticamente quando l'espressione musicale all'interno del blocco `\score` viene interpretata. I valori predefiniti delle proprietà di contesto validi per tutta la partitura possono essere modificati nel blocco `\layout`. Vedi ⟨undefined⟩ [Modifying context properties], pagina ⟨undefined⟩

Nelle sezioni precedenti hai già visto molti esempi pratici della creazione di nuovi contesti `Staff` e `Voice`, ma per ricordarti come questi comandi vengano usati in pratica, ecco un esempio di musica vera e propria:

```
\score {  % start of single compound music expression
  <<  % start of simultaneous staves section
    \time 2/4
    \new Staff {  % create RH staff
      \clef "treble"
      \key g \minor
      \new Voice {  % create voice for RH notes
        \relative c'' {  % start of RH notes
          d4 ees16 c8. |
```

```
        d4 ees16 c8. |
      }  % end of RH notes
    }  % end of RH voice
  }  % end of RH staff
  \new Staff <<  % create LH staff; needs two simultaneous voices
    \clef "bass"
    \key g \minor
    \new Voice {  % create LH voice one
      \voiceOne
      \relative g {  % start of LH voice one notes
        g8 <bes d> ees, <g c> |
        g8 <bes d> ees, <g c> |
      }  % end of LH voice one notes
    }  % end of LH voice one
    \new Voice {  % create LH voice two
      \voiceTwo
      \relative g {  % start of LH voice two notes
        g4 ees |
        g4 ees |
      }  % end of LH voice two notes
    }  % end of LH voice two
  >>  % end of LH staff
>>  % end of simultaneous staves section
}  % end of single compound music expression
```



(Si noti che tutte le asserzioni che aprono un blocco o con delle parentesi graffe, `{`, o con delle parentesi angolari doppie, `<<`, sono indentate di due spazi, e la parentesi di chiusura corrispondente è indentata esattamente dello stesso spazio. Pur non essendo un requisito indispensabile, seguire questa pratica ridurrà enormemente il numero di errori da 'parentesi spaiate', ed è quindi fortemente consigliato. Permette di vedere a colpo d'occhio la struttura della musica, e qualsiasi parentesi spaiata sarà facilmente riconoscibile. Si noti anche come il rigo LH faccia uso di parentesi angolari doppie perché richiede due voci, mentre il rigo RH è costituito da una singola espressione musicale compresa tra parentesi graffe perché richiede una sola voce.)

Il comando `\new` può anche dare un nome identificativo al contesto per distinguerlo da altri contesti dello stesso tipo,

```
\new tipo = id espressione-musicale
```

Si noti la distinzione tra il nome del tipo di contesto, `Staff`, `Voice`, etc, e il nome identificativo di un particolare esempio di quel tipo, che può essere qualsiasi sequenza di lettere inventata dall'utente. Nel nome identificativo si possono usare anche numeri e spazi, ma in questo caso deve essere compreso tra virgolette, ovvero `\new Staff = "MioPentagramma 1"` *`espressione-musicale`*. Il nome identificativo viene utilizzato per riportare a quel particolare esempio di un contesto. Abbiamo visto questo utilizzo nella sezione sul testo, si veda Sezione 3.2.3 [Voci e musica vocale], pagina 56.

Vedi anche

Guida alla notazione: Sezione “Creating contexts” in *Guida alla Notazione*.

3.3.3 Gli incisori

Ogni segno presente nell’output di una partitura realizzata con LilyPond è prodotto da un `Engraver` (incisore). Dunque c’è un incisore per creare i righi, uno per le teste delle note, uno per i gambi, uno per le travature, etc, etc. In totale ci sono più di 120 incisori! Fortunatamente, per gran parte delle partiture è necessario conoscerne pochi soltanto, e per partiture semplici non occorre conoscerne alcuno.

Gli incisori vivono ed operano all’interno dei Contesti. Incisori come il `Metronome_mark_engraver`, la cui azione e il cui output si applicano alla partitura nel suo complesso, operano nel contesto di livello più superiore – il contesto `Score`.

Gli incisori `Clef_engraver` e `Key_engraver` devono invece trovarsi in ogni contesto `Staff`, poiché righi diversi potrebbero richiedere diverse chiavi e tonalità.

Gli incisori `Note_heads_engraver` e `Stem_engraver` abitano ogni contesto `Voice`, il contesto che si trova al livello più basso di tutti.

Ogni incisore elabora gli oggetti specifici associati alla sua funzione, e gestisce le proprietà che a quella funzione si riferiscono. Queste proprietà, come le proprietà associate ai contesti, possono essere modificate per cambiare il funzionamento dell’incisore o l’aspetto di quegli elementi nella partitura.

Gli incisori hanno tutti dei nomi composti formati da parole che descrivono la loro funzione. Solo la prima parola inizia con una maiuscola, e il resto è collegato insieme con dei trattini bassi. Quindi l’incisore `Staff_symbol_engraver` ha il compito di creare le linee del pentagramma, il `Clef_engraver` determina e definisce il punto di riferimento dell’altezza sul rigo disegnando il simbolo di una chiave.

Ecco alcuni degli incisori più utilizzati insieme alla loro funzione. Vedrai che di solito è facile indovinare la funzione a partire dal nome, o viceversa.

Incisore	**Funzione**
Accidental_engraver	Crea le alterazioni, le alterazioni di precauzione e di cortesia
Beam_engraver	Incide le travature
Clef_engraver	Incide le chiavi
Completion_heads_engraver	Separa le note che attraversano le stanghette
New_dynamic_engraver	Crea le forcelle e i testi relativi alla dinamica
Forbid_line_break_engraver	Impedisce l’a capo se un elemento musicale è ancora attivo
Key_engraver	Crea l’armatura di chiave
Metronome_mark_engraver	Incide il tempo metronomico
Note_heads_engraver	Incide le teste delle note
Rest_engraver	Incide le pause
Staff_symbol_engraver	Incide le cinque linee (predefinite) del pentagramma
Stem_engraver	Crea i gambi e i tremoli su singoli gambi
Time_signature_engraver	Crea l’indicazione di tempo

Vedremo in seguito come cambiare l’output di Lilypond modificando il comportamento degli incisori.

Vedi anche

Guida al funzionamento interno: Sezione “Engravers and Performers” in *Guida al Funzionamento Interno*.

3.3.4 Modificare le proprietà di contesto

I contesti si occupano di tenere i valori di un certo numero di *proprietà* del contesto. Molte di queste possono essere cambiate per influenzare l'interpretazione dell'input e quindi modificare l'aspetto dell'output. Per cambiarle si usa il comando `\set`. Questo assume la seguente forma

```
\set NomeDelContesto.nomeDellaProprietà = #valore
```

Dove il *NomeDelContesto* è di solito `Score`, `Staff` o `Voice`. Può essere omesso, e in questo caso viene considerato il contesto attuale (solitamente `Voice`).
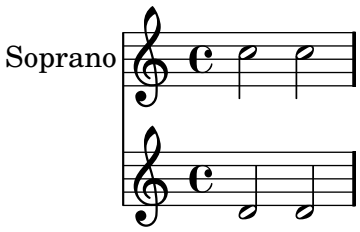
I nomi delle proprietà del contesto consistono in parole unite insieme e senza lineette o trattini bassi, e solo la prima parola inizia con una lettera maiuscola. Ecco alcuni esempi di quelle più usate comunemente. Ma ne esistono molte altre.

nomeDellaProprietà	**Tipo**	**Funzione**	**Valore di esempio**
extraNatural	Booleano	Se è vero, mette i segni di bequadro prima degli accidenti	`#t`, `#f`
currentBarNumber	Intero	Imposta il numero della battuta corrente	`50`
doubleSlurs	Booleano	Se è vero, stampa le legature di portamento sia sopra che sotto le note	`#t`, `#f`
instrumentName	Testo	Imposta il nome da inserire all'inizio del rigo	`"Cello I"`
fontSize	Reale	Aumenta o riduce la dimensione del carattere	`2.4`
stanza	Testo	Imposta il testo da stampare prima dell'inizio di una strofa	`"2"`

mentre un Booleano è o Vero (`#t`) o Falso (`#f`), un Intero è un numero intero positivo, un Reale è un numero decimale positivo o negativo, e il testo è racchiuso tra virgolette. Si noti la presenza del segno cancelletto, (`#`), in due punti diversi – come parte del valore Booleano prima di `t` o `f`, e prima del *valore* nell'affermazione `\set`. Dunque quando si inserisce un valore Booleano bisogna scrivere due cancelletti, ad esempio `##t`.

Prima di poter impostare una qualsiasi di queste proprietà dobbiamo sapere in quale contesto esse operino. Talvolta questo è ovvio, ma talvolta può risultare complicato. Se viene specificato il contesto sbagliato, non viene generato alcun messaggio di errore, ma l'azione desiderata non avrà luogo. Ad esempio, la proprietà `instrumentName` risiede ovviamente nel contesto `Staff`, poiché è il pentagramma a dover essere nominato. Nell'esempio seguente viene etichettato il primo pentagramma, ma non il secondo, perché abbiamo omesso il nome del contesto.

```
<<
  \new Staff \relative c'' {
    \set Staff.instrumentName = #"Soprano"
    c2 c
 }
  \new Staff \relative c' {
    \set instrumentName = #"Alto"  % Wrong!
    d2 d
 }
>>
```



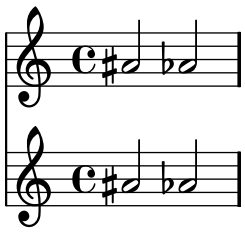
Ricorda che il nome di contesto predefinito è `Voice`, dunque il secondo comando `\set` imposta la proprietà `instrumentName` nel contesto `Voice` col valore "Alto", ma dato che LilyPond non trova tale proprietà nel contesto `Voice`, non ha avuto luogo alcuna altra azione. Questo non è un errore, e nessun messaggio di errore viene riportato nel file di log.

Analogamente, se il nome della proprietà è stato scritto male, non viene generato alcun messaggio di errore, e ovviamente l'azione desiderata non può essere eseguita. Infatti, col comando `\set` puoi impostare qualsiasi 'proprietà' (anche inventata) usando qualsiasi nome che ti piaccia in qualsiasi contesto esistente. Ma se LilyPond non conosce il nome, allora non verrà intrapresa alcuna azione. Alcuni editor di testo che hanno uno speciale supporto per i file di input di LilyPond presentano i nomi delle proprietà con una lista scorrevole al passaggio del mouse, come JEdit col plugin LilyPondTool, oppure evidenziano in modo diverso i nomi delle proprietà non conosciuti, come fa ConTEXT. Se non usi un editor con queste funzionalità, è consigliabile controllare il nome delle proprietà nella Guida al funzionamento interno: si veda Sezione "Tunable context properties" in *Guida al Funzionamento Interno*, o Sezione "Contexts" in *Guida al Funzionamento Interno*.

La proprietà `instrumentName` funzionerà soltanto se inserita nel contesto `Staff`, ma alcune proprietà possono essere collocate in più di un contesto. Ad esempio, la proprietà `extraNatural` è impostata di default sul valore ##t (vero) su tutti i righi. Se viene impostata su ##f (falso) in uno specifico contesto `Staff` si applica solo alle alterazioni presenti su quel rigo. Se impostata su falso nel contesto `Score` si applica a tutti i righi.

Quindi in questo modo si disattiva il bequadro su un rigo:
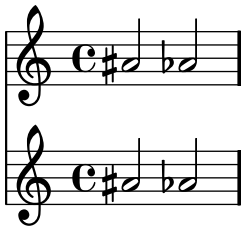
```
<<
  \new Staff \relative c'' {
    ais2 aes
 }
  \new Staff \relative c'' {
    \set Staff.extraNatural = ##f
    ais2 aes
 }
>>
```



e in questo modo si disattiva in tutti i righi:

```
<<
  \new Staff \relative c'' {
    ais2 aes
 }
  \new Staff \relative c'' {
    \set Score.extraNatural = ##f
    ais2 aes
```
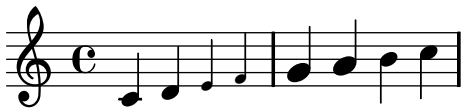
```
 }
>>
```



Un altro esempio: se la proprietà `clefOctavation` viene posta nel contesto `Score`, cambia immediatamente il valore dell'ottavazione in tutti i righi presenti e imposta un nuovo valore predefinito che sarà applicato a tutti i righi.

Il comando opposto, `\unset`, di fatto rimuove la proprietà dal contesto, e questo fa sì che molte proprietà tornino al valore predefinito. Solitamente `\unset` non è necessario dal momento che un nuovo comando `\set` permetterà di ottenere quanto si desidera.

I comandi `\set` e `\unset` possono trovarsi in qualsiasi punto del file di input e avranno effetto dal momento in cui si incontrano fino alla fine della partitura o finché la proprietà non viene attivata (`\set`) o disattivata (`\unset`) di nuovo. Proviamo a cambiare varie volte la dimensione del font, che influisce (tra le altre cose) sulla dimensione delle teste delle note. La modifica è relativa al valore predefinito, non all'ultimo valore impostato.

```
c4 d
% make note heads smaller
\set fontSize = #-4
e4 f |
% make note heads larger
\set fontSize = #2.5
g4 a
% return to default size
\unset fontSize
b4 c |
```



Abbiamo visto come impostare i valori di diversi tipi di proprietà. Si note che gli interi e i numeri sono sempre preceduti da un segno di cancelletto, `#`, mentre un valore vero o falso è specificato con `##t` e `##f`, con due cancelletti. Una proprietà testuale dovrebbe essere racchiusa tra virgolette, come abbiamo visto prima, sebbene vedremo in seguito che in realtà il testo può essere specificato in un modo molto più generale usando il potente comando `\markup`.
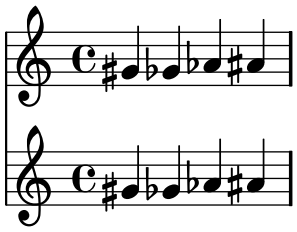
Impostare le proprietà di contesto con `\with`

Il valore predefinito delle proprietà di contesto possono essere impostate anche nel momento in cui il contesto viene creato. Talvolta questo è un modo più chiaro per specificare il valore della proprietà se questa deve rimanere fissa per la durata del contesto. Un contesto creato col comando `\new` può essere immediatamente seguito da un blocco `\with { .. }` dove vengono impostati i valori predefiniti della proprietà. Ad esempio, se vogliamo eliminare la stampa del bequadro per l'estensione di un rigo possiamo scrivere:

```
\new Staff \with { extraNatural = ##f }
```

così:

```
<<
  \new Staff {
    \relative c'' {
      gis4 ges aes ais
    }
  }
  \new Staff \with { extraNatural = ##f } {
    \relative c'' {
      gis4 ges aes ais
    }
  }
>>
```
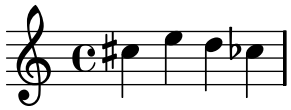


Si possono ancora cambiare dinamicamente le proprietà impostate in questo modo usando `\set`, mentre con `\unset` si possono riportare al valore predefinito impostato nel blocco `\with`.

Quindi se la proprietà `fontSize` viene inserita in una proposizione `\with`, imposta il valore predefinito della dimensione del font. Se viene in seguito modificato con `\set`, questo nuovo valore predefinito può essere recuperato col comando `\unset fontSize`.

Impostare le proprietà di contesto con `\context`

I valori delle proprietà di un contesto possono essere impostate in *tutti* i contesti di un particolare tipo, così come in tutti i contesti `Staff`, con un solo comando. Il tipo di contesto viene identificato attraverso il suo nome, come `Staff`, preceduto da una barra inversa (backslash): `\Staff`. La dichiarazione che imposta il valore della proprietà è la stessa che abbiamo visto nel blocco `\with`, introdotto prima. Viene posta in un blocco `\context` all'interno di un blocco `\layout`. Ogni blocco `\context` avrà effetto su tutti i contesti del tipo specificato nel blocco `\score` o `\book` nel quale il blocco `\layout` si trova. Ecco un esempio per mostrare la struttura:

```
\score {
  \new Staff {
    \relative c'' {
      cis4 e d ces
    }
  }
  \layout {
    \context {
      \Staff
      extraNatural = ##t
    }
  }
}
```



Le proprietà di contesto impostate in questo modo possono essere sovrascritte per alcuni particolari contesti attraverso asserzioni in un blocco `\with`, e tramite comandi `\set` incorporati nelle asserzioni musicali.

Vedi anche

Guida alla notazione: Sezione "Changing context default settings" in *Guida alla Notazione*, Sezione "The set command" in *Guida alla Notazione*.

Guida al funzionamento interno: Sezione "Contexts" in *Guida al Funzionamento Interno*, Sezione "Tunable context properties" in *Guida al Funzionamento Interno*.

3.3.5 Aggiungere e togliere gli incisori

Abbiamo visto che ciascuno dei contesti contiene vari incisori, ognuno dei quali ha il compito di produrre una parte specifica dell'output, come stanghette, righi, teste, gambi, etc.. Se un incisore viene rimosso da un contesto, non può più produrre il suo output. Si tratta di un metodo sbrigativo per modificare l'output, e talvolta può essere utile.

Cambiare un singolo contesto

Per rimuovere un incisore da un singolo contesto usiamo il comando `\with` posto subito dopo il comando di creazione del contesto, come esposto nella sezione precedente.

A titolo di esempio, ripetiamo un esempio della sezione precedente rimuovendo le linee del rigo. Ricorda che le linee del rigo vengono create dall'incisore `Staff_symbol_engraver`.

```
\new Staff \with {
  \remove Staff_symbol_engraver
}
\relative c' {
  c4 d
  \set fontSize = #-4  % make note heads smaller
  e4 f |
  \set fontSize = #2.5  % make note heads larger
  g4 a
  \unset fontSize  % return to default size
  b4 c |
}
```
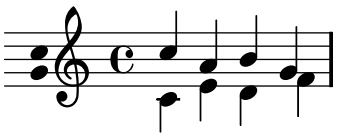


Gli incisori possono essere aggiunti anche a contesti individuali. Il comando per farlo è

`\consists` *`Engraver_name`*,

posto all'interno di un blocco `\with`. Alcune partiture vocali hanno un ambitus collocato all'inizio del pentagramma per indicare l'intervallo tra la nota più grave e quella più acuta in quella parte – si veda Sezione "ambitus" in *Glossario Musicale*. L'ambitus viene creato dall'incisore `Ambitus_engraver`, che di norma non è incluso in alcun contesto. Se lo aggiungiamo al contesto `Voice`, calcola l'intervallo di quella voce soltanto:
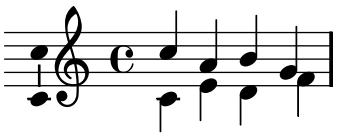
```
\new Staff <<
  \new Voice \with {
    \consists Ambitus_engraver
  } {
    \relative c'' {
```

```
      \voiceOne
      c4 a b g
    }
  }
  \new Voice {
    \relative c' {
      \voiceTwo
      c4 e d f
    }
  }
>>
```



ma se aggiungiamo l'incisore ambitus al contesto `Staff`, calcola l'intervallo di tutte le note in tutte le voci di quel pentagramma:

```
\new Staff \with {
  \consists Ambitus_engraver
}
<<
  \new Voice {
    \relative c'' {
      \voiceOne
      c4 a b g
    }
  }
  \new Voice {
    \relative c' {
      \voiceTwo
      c4 e d f
    }
  }
>>
```
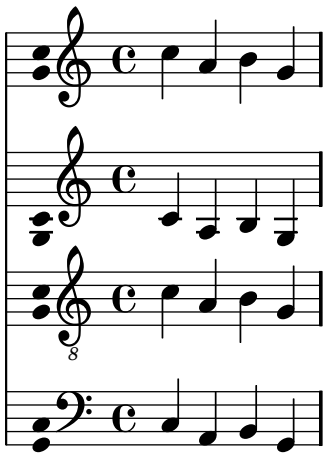


Modificare tutti i contesti dello stesso tipo

Gli esempi precedenti mostrano come rimuovere o aggiungere degli incisori a contesti individuali. È anche possibile rimuovere o aggiungere gli incisori per ogni contesto di un particolare tipo inserendo i comandi nel contesto appropriato in un blocco `\layout`. Ad esempio, se volessimo mostrare un ambitus per ogni pentagramma in una partitura di quattro pentagrammi, potremmo scrivere

```
\score {
  <<
    \new Staff {
      \relative c'' {
```

```
        c4 a b g
      }
    }
    \new Staff {
      \relative c' {
        c4 a b g
      }
    }
    \new Staff {
      \clef "G_8"
      \relative c' {
        c4 a b g
      }
    }
    \new Staff {
      \clef "bass"
      \relative c {
        c4 a b g
      }
    }
  >>
  \layout {
    \context {
      \Staff
      \consists Ambitus_engraver
    }
  }
}
```



I valori delle proprietà di un contesto possono essere impostati anche per tutti i contesti di un particolare tipo includendo il comando `\set` in un blocco `\context` nello stesso modo.

Vedi anche

Guida alla notazione: Sezione “Modifying context plug-ins” in *Guida alla Notazione*, Sezione “Changing context default settings” in *Guida alla Notazione*.

Problemi noti e avvertimenti

Gli incisori `Stem_engraver` e `Beam_engraver` fissano i propri oggetti alle teste delle note. Se si elimina `Note_heads_engraver` le teste delle note non vengono prodotte e di conseguenza non vengono creati nemmeno i gambi né le travature.

3.4 Estendere i modelli

Hai letto il tutorial, sai come scrivere la musica, comprendi i concetti fondamentali. Ma come puoi ottenere i pentagrammi che desideri? Puoi trovare molti modelli (vedi ⟨undefined⟩ [Templates], pagina ⟨undefined⟩) da cui partire. Ma se ti serve qualcosa che non è presente lì? Continua a leggere.

3.4.1 Soprano e violoncello

Parti dal modello che sembra più simile a ciò che vuoi ottenere. Diciamo che vuoi scrivere qualcosa per soprano e violoncello. In questo caso, inizieremmo col modello 'Note e testo' (per la parte di soprano).

```
\version "2.14.2"

melodia = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

testo = \lyricmode {
  Aaa Bee Cee Dee
}

\score {
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melodia
    }
    \new Lyrics \lyricsto "one" \testo
  >>
  \layout { }
  \midi { }
}
```

Ora vogliamo aggiungere una parte per violoncello. Vediamo l'esempio 'Solo note':

```
\version "2.14.2"

melodia = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

\score {
```

```
  \new Staff \melodia
  \layout { }
  \midi { }
}
```

Non ci servono due comandi `\version`. Ci servirà invece la sezione `melodia`. Non vogliamo due sezioni `\score` – se avessimo due `\score`, le due parti sarebbero separate. Ma le vogliamo insieme, come un duetto. All'interno della sezione `\score`, non ci servono due `\layout` o due `\midi`.

Se semplicemente tagliamo e incolliamo la sezione `melodia`, finiremo con l'avere due definizioni `melodia`. Questo non produrrebbe un errore, ma la seconda definizione sarebbe usata per entrambe le melodie. Dunque rinominiamole per distinguerle. Chiameremo la sezione per il soprano `musicaSoprano` e la sezione per il violoncello `musicaVioloncello`. Se facciamo questo, rinominiamo anche `testo` in `testoSoprano`. Ricorda di rinominare entrambe le occorrenze di tutti questi nomi – sia la definizione iniziale (la parte `melody = \relative c' {`) sia l'uso del nome (nella sezione `\score`).

Cambiamo anche il pentagramma della parte per violoncello – che normalmente usa la chiave di basso. Daremo anche al violoncello delle note diverse.

```
\version "2.14.2"

musicaSoprano = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

testoSoprano = \lyricmode {
  Aaa Bee Cee Dee
}

musicaVioloncello = \relative c {
  \clef "bass"
  \key c \major
  \time 4/4
  d4 g fis8 e d4
}

\score {
  <<
    \new Voice = "one" {
      \autoBeamOff
      \musicaSoprano
    }
    \new Lyrics \lyricsto "one" \testoSoprano
  >>
  \layout { }
  \midi { }
}
```

L'aspetto è promettente, ma la parte del violoncello non apparirà nella partitura – perché non l'abbiamo inserita nella sezione `\score`. Se vogliamo che la parte del violoncello compaia sotto quella del soprano, dobbiamo aggiungere

```
\new Staff \musicaVioloncello
```

sotto la parte del soprano. Dobbiamo anche aggiungere `<<` e `>>` intorno alla musica – che dice a LilyPond che c'è più di una cosa (in questo caso, due `Staff`) che sono simultanei. Lo `\score` ora appare così:

```
\score {
  <<
  <<
    \new Voice = "one" {
      \autoBeamOff
      \musicaSoprano
    }
    \new Lyrics \lyricsto "one" \testoSoprano
  >>
  \new Staff \musicaVioloncello
  >>
  \layout { }
  \midi { }
}
```

L'aspetto è un po' confuso; l'indentazione non è pulita. Ma si può correggere facilmente. Ecco il template completo per soprano e violoncello.

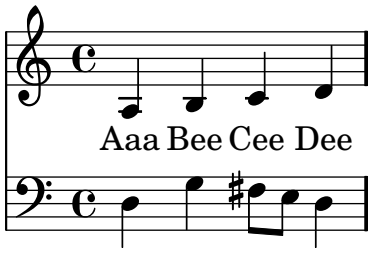
```
\version "2.14.2"
musicaSoprano = \relative c' {
  \clef "treble"
  \key c \major
  \time 4/4
  a4 b c d
}

testoSoprano = \lyricmode {
  Aaa Bee Cee Dee
}

musicaVioloncello = \relative c {
  \clef "bass"
  \key c \major
  \time 4/4
  d4 g fis8 e d4
}

\score {
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \musicaSoprano
      }
      \new Lyrics \lyricsto "one" \testoSoprano
    >>
    \new Staff \musicaVioloncello
  >>
  \layout { }
```

```
  \midi { }
}
```

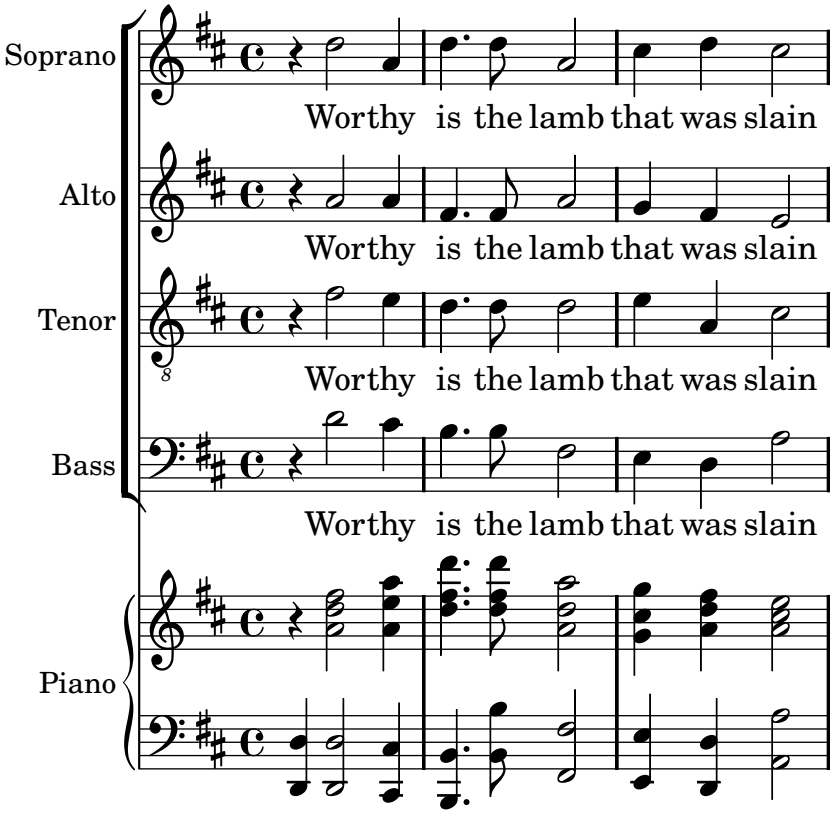


Vedi anche

I modelli da cui partire si trovano nell'appendice 'Modelli', si veda ⟨undefined⟩ [Single staff], pagina ⟨undefined⟩.

3.4.2 Partitura vocale a quattro parti SATB

Gran parte delle partiture vocali di musica scritta per coro misto a quattro voci con accompagnamento di orchestra, come l'Eliah di Mendelssohn o il Messiah di Handel, hanno la musica corale e le parole su quattro righi, una per ognuna delle voci SATB, con in basso una riduzione per pianoforte dell'accompagnamento orchestrale. Ecco un esempio tratto dal Messiah di Handel:



Nessun modello fornisce esattamente questo assetto. Quello più simile è 'Partitura vocale SATB e riduzione automatica per pianoforte' – si veda ⟨undefined⟩ [Vocal ensembles], pagina ⟨undefined⟩ – ma abbiamo bisogno di modificare l'assetto e aggiungere un accompagnamento per pianoforte che non sia derivato automaticamente dalle parti vocali. Le variabili che si riferiscono alla musica e alle parole per le parti vocali vanno bene, ma dovremo aggiungere le variabili per la riduzione per pianoforte.

L'ordine con cui i contesti appaiono nel ChoirStaff del modello non corrisponde all'ordine della partitura vocale mostrata sopra. Dobbiamo risistemarli in modo che ci siano quattro righi con le parole scritte direttamente sotto le note di ogni parte. Tutte le voci devono essere `\voiceOne`, che è l'impostazione predefinita, quindi i comandi `\voiceXXX` devono essere tolti. Dobbiamo anche specificare la chiave di tenore per i tenori. Il modo in cui il testo viene specificato nel modello non è stato ancora esaminato quindi dovremo usare un metodo con cui siamo già familiari. Aggiungeremo anche i nomi per ogni rigo.

In questo modo il nostro ChoirStaff avrà questo aspetto:

```
\new ChoirStaff <<
  \new Staff = "sopranos" <<
    \set Staff.instrumentName = #"Soprano"
    \new Voice = "sopranos" {
      \global
      \sopranoMusic
    }
  >>
  \new Lyrics \lyricsto "sopranos" {
    \sopranoWords
  }
  \new Staff = "altos" <<
    \set Staff.instrumentName = #"Alto"
    \new Voice = "altos" {
      \global
      \altoMusic
    }
  >>
  \new Lyrics \lyricsto "altos" {
    \altoWords
  }
  \new Staff = "tenors" <<
    \set Staff.instrumentName = #"Tenor"
    \new Voice = "tenors" {
      \global
      \tenorMusic
    }
  >>
  \new Lyrics \lyricsto "tenors" {
    \tenorWords
  }
  \new Staff = "basses" <<
    \set Staff.instrumentName = #"Bass"
    \new Voice = "basses" {
      \global
      \bassMusic
    }
  >>
  \new Lyrics \lyricsto "basses" {
    \bassWords
  }
>>  % end ChoirStaff
```

Poi dobbiamo lavorare sulla parte per pianoforte. Questo è facile - basta prendere la parte per pianoforte dal modello 'Pianoforte solista':

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano  "
  \new Staff = "upper" \upper
  \new Staff = "lower" \lower
>>
```

e aggiungere le definizioni delle variabili `upper` e `lower`.

ChoirStaff e PianoStaff devono essere uniti attraverso parentesi angolari, perché vogliamo che siano impilati uno sopra l'altro:

```
<<  % combine ChoirStaff and PianoStaff one above the other
  \new ChoirStaff <<
    \new Staff = "sopranos" <<
      \new Voice = "sopranos" {
        \global
        \sopranoMusic
      }
    >>
    \new Lyrics \lyricsto "sopranos" {
      \sopranoWords
     }
    \new Staff = "altos" <<
      \new Voice = "altos" {
        \global
        \altoMusic
      }
    >>
    \new Lyrics \lyricsto "altos" {
      \altoWords
    }
    \new Staff = "tenors" <<
      \clef "G_8"  % tenor clef
      \new Voice = "tenors" {
        \global
        \tenorMusic
      }
    >>
    \new Lyrics \lyricsto "tenors" {
      \tenorWords
    }
    \new Staff = "basses" <<
      \clef "bass"
      \new Voice = "basses" {
        \global
        \bassMusic
      }
    >>
    \new Lyrics \lyricsto "basses" {
      \bassWords
    }
  >>  % end ChoirStaff
```

```
  \new PianoStaff <<
    \set PianoStaff.instrumentName = #"Piano"
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
>>
```

Unendo tutto questo e aggiungendo la musica per le tre battute dell'esempio precedente, otteniamo:

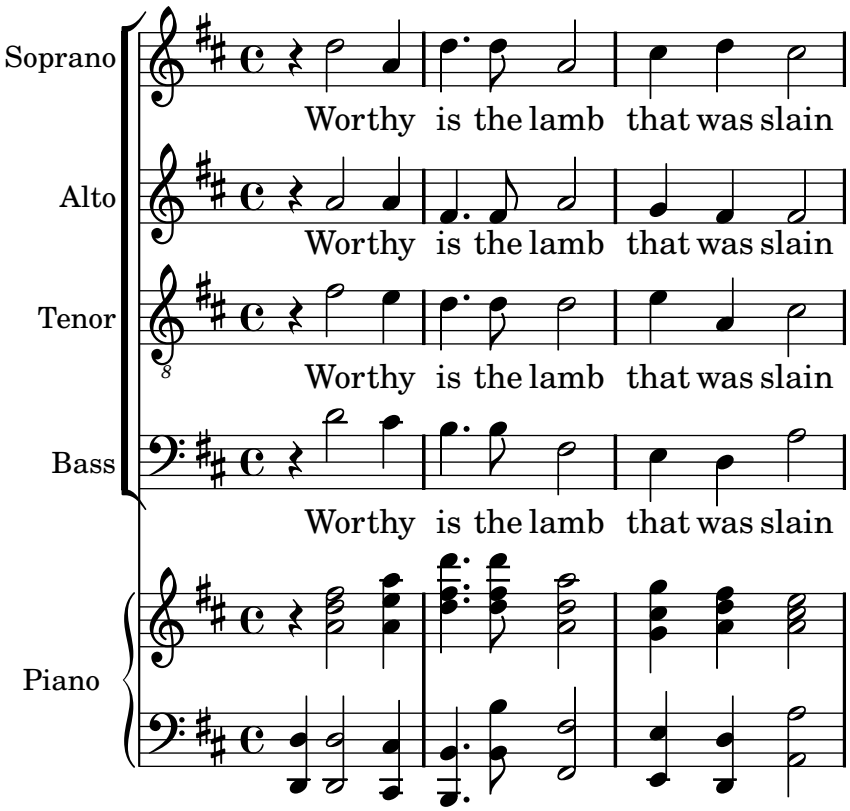
```
\version "2.14.2"

global = { \key d \major \time 4/4 }
sopranoMusic = \relative c'' {
  \clef "treble"
  r4 d2 a4 | d4. d8 a2 | cis4 d cis2 |
}
sopranoWords = \lyricmode {
  Wor -- thy | is the lamb | that was slain |
}
altoMusic = \relative a' {
  \clef "treble"
  r4 a2 a4 | fis4. fis8 a2 | g4 fis fis2 |
}
altoWords = \sopranoWords
tenorMusic = \relative c' {
  \clef "G_8"
  r4 fis2 e4 | d4. d8 d2 | e4 a, cis2 |
}
tenorWords = \sopranoWords
bassMusic = \relative c' {
  \clef "bass"
  r4 d2 cis4 | b4. b8 fis2 | e4 d a'2 |
}
bassWords = \sopranoWords
upper = \relative a' {
  \clef "treble"
  \global
  r4 <a d fis>2 <a e' a>4 |
  <d fis d'>4. <d fis d'>8 <a d a'>2 |
  <g cis g'>4 <a d fis> <a cis e>2 |
}
lower = \relative c, {
  \clef "bass"
  \global
  <d d'>4 <d d'>2 <cis cis'>4 |
  <b b'>4. <b' b'>8 <fis fis'>2 |
  <e e'>4 <d d'> <a' a'>2 |
}

\score {
  <<  % combine ChoirStaff and PianoStaff in parallel
    \new ChoirStaff <<
```

```
      \new Staff = "sopranos" <<
        \set Staff.instrumentName = #"Soprano"
        \new Voice = "sopranos" {
          \global
          \sopranoMusic
        }
      >>
      \new Lyrics \lyricsto "sopranos" {
        \sopranoWords
      }
      \new Staff = "altos" <<
        \set Staff.instrumentName = #"Alto"
        \new Voice = "altos" {
          \global
          \altoMusic
        }
      >>
      \new Lyrics \lyricsto "altos" {
        \altoWords
      }
      \new Staff = "tenors" <<
        \set Staff.instrumentName = #"Tenor"
        \new Voice = "tenors" {
          \global
          \tenorMusic
        }
      >>
      \new Lyrics \lyricsto "tenors" {
        \tenorWords
      }
      \new Staff = "basses" <<
        \set Staff.instrumentName = #"Bass"
        \new Voice = "basses" {
          \global
          \bassMusic
        }
      >>
      \new Lyrics \lyricsto "basses" {
        \bassWords
      }
    >>  % end ChoirStaff

    \new PianoStaff <<
      \set PianoStaff.instrumentName = #"Piano  "
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
}
```



3.4.3 Scrivere una partitura da zero

Dopo aver acquisito un po' di confidenza nella scrittura del codice LilyPond, potresti scoprire che è più facile costruire una partitura da zero piuttosto che modificare uno dei modelli. In questo modo puoi anche sviluppare il tuo stile per adattarlo al tipo di musica che vuoi. Come esempio, vediamo come mettere insieme la partitura di un preludio per organo.

Cominciamo con una sezione d'intestazione. Qui va il titolo, il nome del compositore, etc, poi vengono le varie definizioni, e infine il blocco della partitura. Spieghiamo questi prima a grandi linee e in seguito aggiungeremo i dettagli.

Useremo le prime due battute del preludio di Bach basato su *Jesu, meine Freude*, che è scritto per organo con due manuali e pedaliera. Puoi vedere queste due battute in fondo a questa sezione. La parte per il manuale superiore ha due voci, quella inferiore e la pedaliera ne hanno una. Abbiamo quindi bisogno di quattro definizioni musicali e di una definizione per stabilire il tempo e la tonalità:

```
\version "2.14.2"
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
keyTime = { \key c \minor \time 4/4 }
ManualOneVoiceOneMusic = { s1 }
ManualOneVoiceTwoMusic = { s1 }
ManualTwoMusic = { s1 }
PedalOrganMusic = { s1 }

\score {
}
```

Per ora abbiamo usato soltanto una nota spaziatrice, `s1`, invece di musica vera e propria. La aggiungeremo in seguito.

Ora vediamo cosa dovrebbe andare nel blocco della partitura. Mostriamo soltanto la struttura del pentagramma che vogliamo. La musica per organo di solito viene scritta su tre righi, uno per ogni mano e uno per i pedali. I righi della tastiera dovrebbero essere raggruppati insieme con una graffa, dunque dobbiamo usare PianoStaff per loro. La parte del primo manuale ha bisogno di due voci mentre la parte per il secondo manuale di una soltanto.

```
\new PianoStaff <<
  \new Staff = "ManualOne" <<
    \new Voice {
      \ManualOneVoiceOneMusic
    }
    \new Voice {
      \ManualOneVoiceTwoMusic
    }
  >>  % end ManualOne Staff context
  \new Staff = "ManualTwo" <<
    \new Voice {
      \ManualTwoMusic
    }
  >>  % end ManualTwo Staff context
>>  % end PianoStaff context
```

Poi dobbiamo aggiungere un rigo per i pedali. Questo va sotto il PianoStaff, ma deve svolgersi simultaneamente a quest'ultimo, quindi abbiamo bisogno delle parentesi angolari intorno ai due gruppi. Altrimenti, verrà generato un errore nel file di log. È un errore comune che farai prima o poi! Prova a copiare l'esempio finale alla fine di questa sezione, togli le parentesi angolari, e compilalo per vedere quali errori genera.

```
<<  % PianoStaff and Pedal Staff must be simultaneous
  \new PianoStaff <<
    \new Staff = "ManualOne" <<
      \new Voice {
        \ManualOneVoiceOneMusic
      }
      \new Voice {
        \ManualOneVoiceTwoMusic
      }
    >>  % end ManualOne Staff context
    \new Staff = "ManualTwo" <<
      \new Voice {
        \ManualTwoMusic
      }
    >>  % end ManualTwo Staff context
  >>  % end PianoStaff context
  \new Staff = "PedalOrgan" <<
    \new Voice {
      \PedalOrganMusic
    }
  >>
>>
```

Non è necessario usare il costrutto simultaneo `<< .. >>` per il rigo del secondo manuale e per quello della pedaliera, poiché contengono solo una espressione musicale, ma non è male usarlo comunque; usare sempre le parentesi angolari dopo `\new Staff` è una buona abitudine da coltivare nel caso ci sia più di una voce. Per le Voci, è vero l'opposto: queste devono essere

seguite regolarmente da parentesi graffe `{ .. }` nel caso in cui la musica sia composta da diverse variabili che devono essere eseguite consecutivamente.

Aggiungiamo questa struttura al blocco della partitura, e aggiustiamo l'indentazione. Aggiungiamo anche le chiavi appropriate, controlliamo che i gambi, le legature di portamento e quelle di valore in ogni voce del rigo superiore puntino nella direzione giusta usando `\voiceOne` e `\voiceTwo`, e inseriamo l'armatura di chiave e il tempo per ogni rigo attraverso la nostra variabile predefinita, `\keyTime`.

```
\score {
  <<  % PianoStaff and Pedal Staff must be simultaneous
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \keyTime  % set key and time signature
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManualOneVoiceOneMusic
        }
        \new Voice {
          \voiceTwo
          \ManualOneVoiceTwoMusic
        }
      >>  % end ManualOne Staff context
      \new Staff = "ManualTwo" <<
        \keyTime
        \clef "bass"
        \new Voice {
          \ManualTwoMusic
        }
      >>  % end ManualTwo Staff context
    >>  % end PianoStaff context
    \new Staff = "PedalOrgan" <<
      \keyTime
      \clef "bass"
      \new Voice {
        \PedalOrganMusic
      }
    >>  % end PedalOrgan Staff
  >>
}  % end Score context
```

L'aspetto dei righi dell'organo mostrati sopra è quasi perfetto; tuttavia c'è un piccolo difetto che non è visibile se si guarda un sistema singolo soltanto: La distanza tra il rigo della pedaliera e il rigo della mano sinistra dovrebbe essere all'incirca la stessa distanza tra il rigo della mano destra e quello della mano sinistra. In particolare, l'allungabilità dei righi in un contesto `PianoStaff` è limitata (in modo che la distanza tra i righi della mano destra e sinistra non possa diventare eccessiva), e il rigo della pedaliera dovrebbe comportarsi allo stesso modo.

L'allungabilità dei righi può essere controllata con la proprietà `staff-staff-spacing` dell' 'oggetto grafico' `VerticalAxisGroup` (gli oggetti grafici vengono comunemente chiamati 'grob' nella documentazione di lilypond) – non preoccuparti dei dettagli in questo momento; in seguito verrà fornita una spiegazione approfondita. I curiosi possono dare un'occhiata a Sezione "Overview of modifying properties" in *Guida alla Notazione*. In questo caso, vogliamo modificare soltanto la sottoproprietà `allungabilità`. Di nuovo, chi è curioso può trovare i valori predefini-

ti per la proprietà staff-staff-spacing nel file 'scm/define-grobs.scm' guardando la definizione del grob VerticalAxisGroup. Il valore dell' allungabilità viene preso dalla definizione del contesto PianoStaff (nel file 'ly/engraver-init.ly') così che i valori siano identici.

```
\score {
  <<  % PianoStaff e Pedal Staff devono essere simultanei
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \keyTime  % imposta l'armatura di chiave e il tempo
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManualOneVoiceOneMusic
        }
        \new Voice {
          \voiceTwo
          \ManualOneVoiceTwoMusic
        }
      >>  % fine del contesto ManualOne Staff
      \new Staff = "ManualTwo" \with {
        \override VerticalAxisGroup
          #'staff-staff-spacing  #'stretchability = 5
      } <<
        \keyTime
        \clef "bass"
        \new Voice {
          \ManualTwoMusic
        }
      >>  % fine del contesto ManualTwo Staff
    >>  % fine del contesto PianoStaff
    \new Staff = "PedalOrgan" <<
      \keyTime
      \clef "bass"
      \new Voice {
        \PedalOrganMusic
      }
    >>  % fine di PedalOrgan Staff
  >>
}  % fine del contesto Score
```

Questo completa la struttura. Qualsiasi musica per organo a tre righi avrà una struttura simile, sebbene il numero delle voci possa variare. Tutto ciò che resta da fare ora è aggiungere la musica, e combinare tutte le parti insieme.
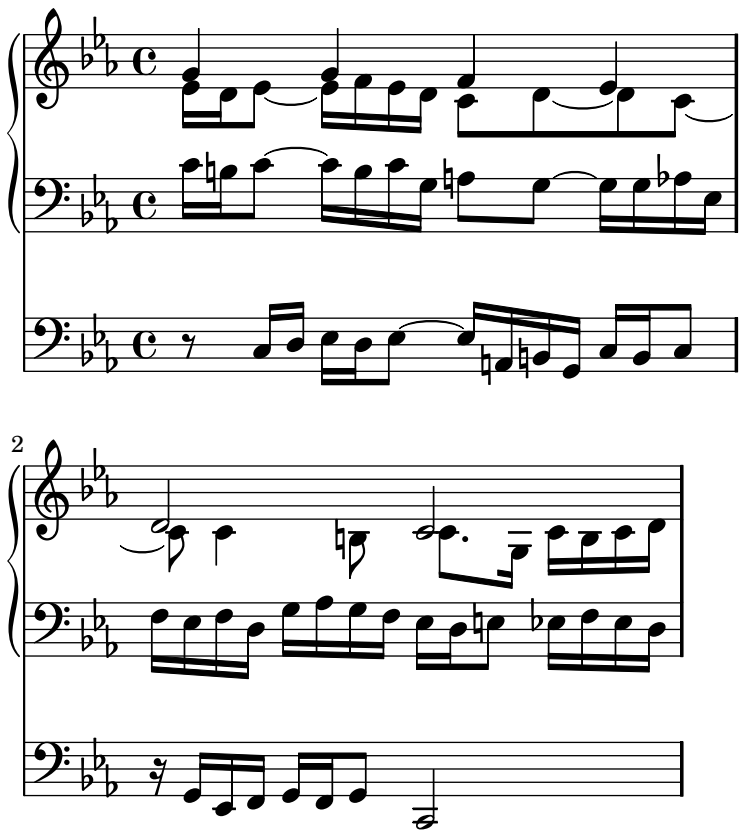
```
\version "2.14.2"
\header {
  title = "Jesu, meine Freude"
  composer = "J S Bach"
}
keyTime = { \key c \minor \time 4/4 }
ManualOneVoiceOneMusic = \relative g' {
  g4 g f ees |
  d2 c |
}
```

```
ManualOneVoiceTwoMusic = \relative c' {
  ees16 d ees8~ ees16 f ees d c8 d~ d c~ |
  c8 c4 b8 c8. g16 c b c d |
}
ManualTwoMusic = \relative c' {
  c16 b c8~ c16 b c g a8 g~ g16 g aes ees |
  f16 ees f d g aes g f ees d e8~ ees16 f ees d |
}
PedalOrganMusic = \relative c {
  r8 c16 d ees d ees8~ ees16 a, b g c b c8 |
  r16 g ees f g f g8 c,2 |
}

\score {
  <<  % PianoStaff and Pedal Staff must be simultaneous
    \new PianoStaff <<
      \new Staff = "ManualOne" <<
        \keyTime  % set key and time signature
        \clef "treble"
        \new Voice {
          \voiceOne
          \ManualOneVoiceOneMusic
        }
        \new Voice {
          \voiceTwo
          \ManualOneVoiceTwoMusic
        }
      >>  % end ManualOne Staff context
      \new Staff = "ManualTwo" \with {
        \override VerticalAxisGroup
          #'staff-staff-spacing #'stretchability = 5
      } <<
        \keyTime
        \clef "bass"
        \new Voice {
          \ManualTwoMusic
        }
      >>  % end ManualTwo Staff context
    >>  % end PianoStaff context
    \new Staff = "PedalOrgan" <<
      \keyTime
      \clef "bass"
      \new Voice {
        \PedalOrganMusic
      }
    >>  % end PedalOrgan Staff context
  >>
}  % end Score context
```

Jesu, meine Freude

J S Bach
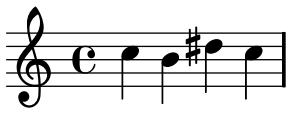


Vedi anche

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

3.4.4 Ridurre l’input grazie a variabili e funzioni

Finora hai visto questo tipo di cose:

```
hornNotes = \relative c'' { c4 b dis c }
\score {
  {
    \hornNotes
  }
}
```
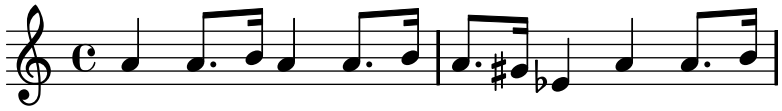


Potresti anche essere accorto che questo può essere utile nella musica minimalista:

```
fragmentA = \relative c'' { a4 a8. b16 }
fragmentB = \relative c'' { a8. gis16 ees4 }

violin = \new Staff {
  \fragmentA \fragmentA |
  \fragmentB \fragmentA |
}

\score {
  {
    \violin
  }
```
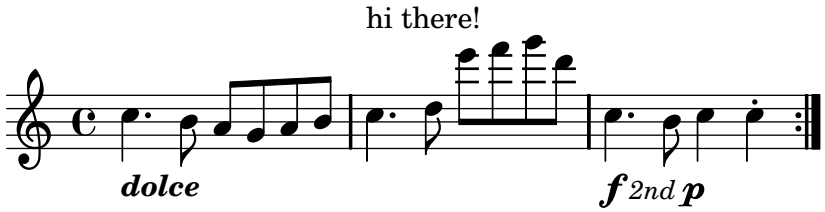
```
}
```



Tuttavia, puoi usare queste variabili (note anche come macro, o comandi definiti dall'utente) anche per le modifiche manuali:

```
dolce = \markup { \italic \bold dolce }

padText = { \once \override TextScript #'padding = #5.0 }
fthenp =_\markup {
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
}

violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}

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



Chiaramente queste variabili sono utili per ridurre la quantità di testo da scrivere. Ma vale la pena tenerle in considerazione anche se le usi una volta sola – perché riducono la complessità. Vediamo l'esempio precedente senza alcuna variabile. È molto difficile da leggere, soprattutto l'ultima linea.

```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup { \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup {
      \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p
    }
    b8 c4 c-. |
  }
```
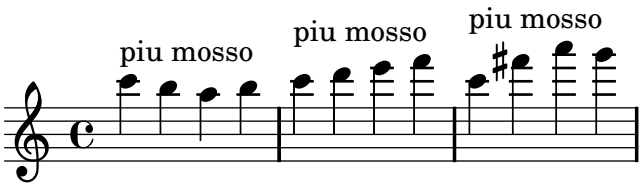
```
}
```

Finora abbiamo visto la sostituzione statica – quando LilyPond vede `\padText`, lo sostituisce con quel che noi abbiamo definito che sia (ovvero tutto ciò che sta a destra di `padtext=`).

LilyPond può gestire anche la sostituzione non statica (la puoi immaginare come una funzione).

```
padText =
#(define-music-function
     (parser location padding)
     (number?)
   #{
     \once \override TextScript #'padding = $padding
   #})

\relative c''' {
  c4^"piu mosso" b a b |
  \padText #1.8
  c4^"piu mosso" d e f |
  \padText #2.6
  c4^"piu mosso" fis a g |
}
```



L'uso di variabili è anche un buon modo per ridurre il lavoro quando la sintassi di input di LilyPond cambia (vedi Sezione "Aggiornare i file con convert-ly" in *Uso del Programma*). Se si ha un'unica definizione (come `\dolce`) per tutti i file di input (vedi Sezione 4.6.3 [Style sheets], pagina 135), allora se la sintassi cambia bisogna aggiornare soltanto la singola definizione `\dolce`, invece di dover modificare tutti i file '`.ly`'.

3.4.5 Partiture e parti

Nella musica orchestrale, tutte le note vengono stampate due volte. Una volta nella parte per i musicisti e una volta nella partitura completa per il direttore. Le variabili sono utili se si vuole evitare il doppio lavoro. La musica viene inserita una volta e salvata all'interno di una variabile. I contenuti di quella variabile vengono poi usati per generare sia la parte che l'intera partitura.

Conviene definire le note in un file speciale. Ad esempio, supponiamo che il file '`horn-music.ly`' contenga la seguente parte di un duetto per corno e fagotto

```
hornNotes = \relative c {
  \time 2/4
  r4 f8 a | cis4 f | e4 d |
}
```

Poi una parte individuale si ottiene inserendo il seguente codice in un file

```
\include "horn-music.ly"
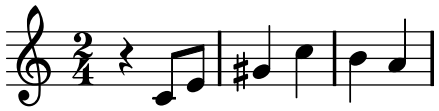
\header {
  instrument = "Horn in F"
}
```

```
{
 \transpose f c' \hornNotes
}
```

La linea

```
\include "horn-music.ly"
```

sostituisce i contenuti di '`horn-music.ly`' in quella posizione del file, quindi `hornNotes` è definito dopo. Il comando `\transpose f c'` indica che l'argomento, ovvero `\hornNotes`, deve essere trasposto di una quinta ascendente. La tonalità `f` è indicata dalla nota `c'`, che corrisponde all'accordatura di un normale corno francese in Fa. La trasposizione può essere vista nel seguente output
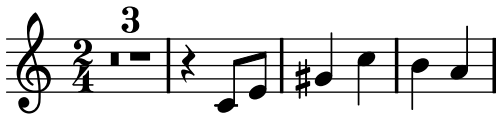


Nei brani di insieme, una delle voci non suona per molte misure. Questo viene indicato da una pausa speciale, la pausa multi-misura. Si inserisce con una `R` maiuscola seguita da una durata (`1` per una semibreve, `2` per una minima, etc.). Moltiplicando la durata, si possono costruire pause più lunghe. Ad esempio, questa pausa prende 3 misure in un tempo di 2/4

```
R2*3
```

Quando la parte viene stampata, le pause multiple devono essere condensate. Si può fare impostando una variabile run-time

```
\set Score.skipBars = ##t
```

Questo comando imposta la proprietà `skipBars` nel contesto `Score` su vero (`##t`). Aggiungendo la pausa e questa opzione alla musica precedente, si arriva al seguente risultato

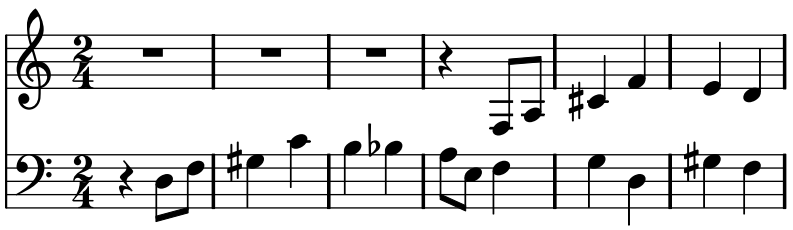


Lo spartito si forma combinando tutta la musica insieme. Assumendo che l'altra voce si trovi in `bassoonNotes` nel file '`bassoon-music.ly`', lo spartito sarà

```
\include "bassoon-music.ly"
\include "horn-music.ly"

<<
  \new Staff \hornNotes
  \new Staff \bassoonNotes
>>
```

ovvero



4 Tweaking output

This chapter discusses how to modify output. LilyPond is extremely configurable; virtually every fragment of output may be changed.

4.1 Tweaking basics

4.1.1 Introduction to tweaks

'Tweaking' is a LilyPond term for the various methods available to the user for modifying the actions taken during interpretation of the input file and modifying the appearance of the printed output. Some tweaks are very easy to use; others are more complex. But taken together the methods available for tweaking permit almost any desired appearance of the printed music to be achieved.

In this section we cover the basic concepts required to understand tweaking. Later we give a variety of ready-made commands which can simply be copied to obtain the same effect in your own scores, and at the same time we show how these commands may be constructed so that you may learn how to develop your own tweaks.

Before starting on this Chapter you may wish to review the section ⟨undefined⟩ [Contexts and engravers], pagina ⟨undefined⟩, as Contexts, Engravers, and the Properties contained within them are fundamental to understanding and constructing Tweaks.

4.1.2 Objects and interfaces

Tweaking involves modifying the internal operation and structures of the LilyPond program, so we must first introduce some terms which are used to describe those internal operations and structures.

The term 'Object' is a generic term used to refer to the multitude of internal structures built by LilyPond during the processing of an input file. So when a command like `\new Staff` is encountered a new object of type `Staff` is constructed. That `Staff` object then holds all the properties associated with that particular staff, for example, its name and its key signature, together with details of the engravers which have been assigned to operate within that staff's context. Similarly, there are objects to hold the properties of all other contexts, such as `Voice` objects, `Score` objects, `Lyrics` objects, as well as objects to represent all notational elements such as bar lines, note heads, ties, dynamics, etc. Every object has its own set of property values.

Some types of object are given special names. Objects which represent items of notation on the printed output such as note heads, stems, slurs, ties, fingering, clefs, etc are called 'Layout objects', often known as 'Graphical Objects', or 'Grobs' for short. These are still objects in the generic sense above, and so they too all have properties associated with them, such as their position, size, color, etc.

Some layout objects are still more specialized. Phrasing slurs, crescendo hairpins, ottava marks, and many other grobs are not localized in a single place – they have a starting point, an ending point, and maybe other properties concerned with their shape. Objects with an extended shape like these are called 'Spanners'.

It remains to explain what 'Interfaces' are. Many objects, even though they are quite different, share common features which need to be processed in the same way. For example, all grobs have a color, a size, a position, etc, and all these properties are processed in the same way during LilyPond's interpretation of the input file. To simplify these internal operations these common actions and properties are grouped together in an object called a `grob-interface`. There are many other groupings of common properties like this, each one given a name ending in `interface`. In total there are over 100 such interfaces. We shall see later why this is of interest and use to the user.

These, then, are the main terms relating to objects which we shall use in this chapter.

4.1.3 Naming conventions of objects and properties

We met some object naming conventions previously, in ⟨undefined⟩ [Contexts and engravers], pagina ⟨undefined⟩. Here for reference is a list of the most common object and property types together with the conventions for naming them and a couple of examples of some real names. We have used 'A' to stand for any capitalized alphabetic character and 'aaa' to stand for any number of lower-case alphabetic characters. Other characters are used verbatim.

Object/property type	**Naming convention**	**Examples**
Contexts	Aaaa or AaaaAaaaAaaa	Staff, GrandStaff
Layout Objects	Aaaa or AaaaAaaaAaaa	Slur, NoteHead
Engravers	Aaaa_aaa_engraver	Clef_engraver, Note_heads_engraver
Interfaces	aaa-aaa-interface	grob-interface, break-aligned-interface
Context Properties	aaa or aaaAaaaAaaa	alignAboveContext, skipBars
Layout Object Properties	aaa or aaa-aaa-aaa	direction, beam-thickness

As we shall see shortly, the properties of different types of object are modified by different commands, so it is useful to be able to recognize the type of object from the names of its properties.

4.1.4 Tweaking methods

\override command

We have already met the commands `\set` and `\with`, used to change the properties of **contexts** and to remove and add **engravers**, in ⟨undefined⟩ [Modifying context properties], pagina ⟨undefined⟩, and ⟨undefined⟩ [Adding and removing engravers], pagina ⟨undefined⟩. We must now introduce some more important commands.

The command to change the properties of **layout objects** is `\override`. Because this command has to modify internal properties deep within LilyPond its syntax is not as simple as the commands you have used so far. It needs to know precisely which property of which object in which context has to be modified, and what its new value is to be. Let's see how this is done.

The general syntax of this command is:

```
\override Context.LayoutObject #'layout-property =
#value
```

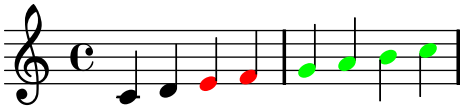
This will set the property with the name *layout-property* of the layout object with the name *LayoutObject*, which is a member of the *Context* context, to the value *value*.

The *Context* can be omitted (and usually is) when the required context is unambiguously implied and is one of lowest level contexts, i.e., `Voice`, `ChordNames` or `Lyrics`, and we shall omit it in many of the following examples. We shall see later when it must be specified.

Later sections deal comprehensively with properties and their values, but to illustrate the format and use of these commands we shall use just a few simple properties and values which are easily understood.

For now, don't worry about the `#'`, which must precede the layout property, and the `#`, which must precede the value. These must always be present in exactly this form. This is the most common command used in tweaking, and most of the rest of this chapter will be directed to presenting examples of how it is used. Here is a simple example to change the color of the note head:

```
c4 d
\override NoteHead #'color = #red
e4 f |
\override NoteHead #'color = #green
g4 a b c |
```



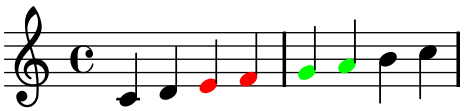
\revert command

Once overridden, the property retains its new value until it is overridden again or a `\revert` command is encountered. The `\revert` command has the following syntax and causes the value of the property to revert to its original default value; note, not its previous value if several `\override` commands have been issued.

```
\revert Context.LayoutObject #'layout-property
```

Again, just like *Context* in the `\override` command, *Context* is often not needed. It will be omitted in many of the following examples. Here we revert the color of the note head to the default value for the final two notes:

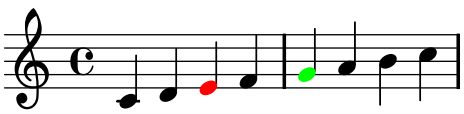
```
c4 d
\override NoteHead #'color = #red
e4 f |
\override NoteHead #'color = #green
g4 a
\revert NoteHead #'color
b4 c |
```



\once prefix

Both the `\override` and the `\set` commands may be prefixed by `\once`. This causes the following `\override` or `\set` command to be effective only during the current musical moment before the property reverts back to its default value. Using the same example, we can change the color of a single note like this:

```
c4 d
\once \override NoteHead #'color = #red
e4 f |
\once \override NoteHead #'color = #green
g4 a b c |
```



\overrideProperty command
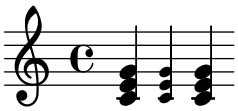
There is another form of the override command, `\overrideProperty`, which is occasionally required. We mention it here for completeness, but for details see Sezione “Difficult tweaks” in *Estendere*.

\tweak command

The final tweaking command which is available is `\tweak`. This should be used to change the properties of objects which occur at the same musical moment, such as the notes within a chord. Using `\override` would affect all the notes within a chord, whereas `\tweak` affects just the following item in the input stream.

Here's an example. Suppose we wish to change the size of the middle note head (the E) in a C major chord. Let's first see what `\once \override` would do:

```
<c e g>4
\once \override NoteHead #'font-size = #-3
<c e g>4
<c e g>4
```
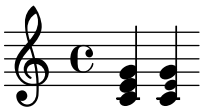


We see the override affects *all* the note heads in the chord. This is because all the notes of a chord occur at the same *musical moment*, and the action of `\once` is to apply the override to all layout objects of the type specified which occur at the same musical moment as the `\override` command itself.

The `\tweak` command operates in a different way. It acts on the immediately following item in the input stream. However, it is effective only on objects which are created directly from the input stream, essentially note heads and articulations; objects such as stems and accidentals are created later and cannot be tweaked in this way. Furthermore, when it is applied to note heads these *must* be within a chord, i.e., within single angle brackets, so to tweak a single note the `\tweak` command must be placed inside single angle brackets with the note.

So to return to our example, the size of the middle note of a chord would be changed in this way:

```
<c e g>4
<c \tweak #'font-size #-3 e g>4
```

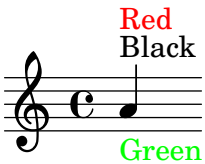


Note that the syntax of `\tweak` is different from that of the `\override` command. Neither the context nor the layout object should be specified; in fact, it would generate an error to do so. These are both implied by the following item in the input stream. Note also that an equals sign should not be present. So the general syntax of the `\tweak` command is simply

```
\tweak #'layout-property #value
```

A `\tweak` command can also be used to modify just one in a series of articulations, as shown here:

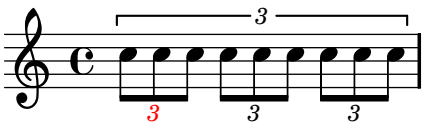
```
a4^"Black"
  -\tweak #'color #red ^"Red"
  -\tweak #'color #green _"Green"
```



Note that the `\tweak` command must be preceded by an articulation mark as if it were an articulation itself.
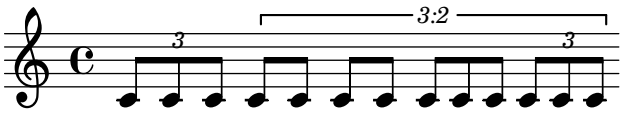
The `\tweak` command must also be used to change the appearance of one of a set of nested tuplets which begin at the same musical moment. In the following example, the long tuplet bracket and the first of the three short brackets begin at the same musical moment, so any `\override` command would apply to both of them. In the example, `\tweak` is used to distinguish between them. The first `\tweak` command specifies that the long tuplet bracket is to be placed above the notes and the second one specifies that the tuplet number is to be printed in red on the first short tuplet bracket.

```
\tweak #'direction #up
\times 4/3 {
  \tweak #'color #red
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
}
```



If nested tuplets do not begin at the same moment, their appearance may be modified in the usual way with `\override` commands:

```
\times 2/3 { c8[ c c] }
\once \override TupletNumber
  #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
  c8[ c]
  c8[ c]
  \once \override TupletNumber #'transparent = ##t
  \times 2/3 { c8[ c c] }
  \times 2/3 { c8[ c c] }
}
```



Vedi anche

Notation Reference: Sezione “The tweak command” in *Guida alla Notazione*.

4.2 The Internals Reference manual

4.2.1 Properties of layout objects

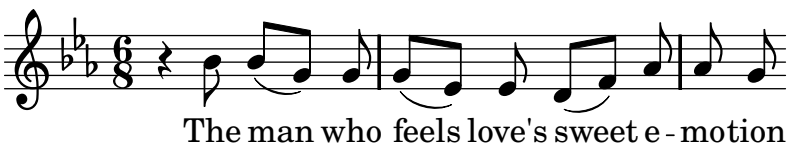
Suppose you have a slur in a score which, to your mind, appears too thin and you’d like to draw it a little heavier. How do you go about doing this? You know from the statements earlier about the flexibility of LilyPond that such a thing should be possible, and you would probably guess that an `\override` command would be needed. But is there a heaviness property for a slur, and if there is, how might it be modified? This is where the Internals Reference manual comes

in. It contains all the information you might need to construct this and all other `\override` commands.

Before we look at the Internals Reference a word of warning. This is a **reference** document, which means there is little or no explanation contained within it: its purpose is to present information precisely and concisely. This means it might look daunting at first sight. Don't worry! The guidance and explanation presented here will enable you to extract the information from the Internals Reference for yourself with just a little practice.

Let's use a concrete example with a simple fragment of real music:

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



Suppose now that we decide we would like the slurs to be a little heavier. Is this possible? The slur is certainly a layout object, so the question is, 'Is there a property belonging to a slur which controls the heaviness?' To answer this we must look in the Internals Reference, or IR for short.

The IR for the version of LilyPond you are using may be found on the LilyPond website at http://lilypond.org. Go to the documentation page and click on the Internals Reference link. For learning purposes you should use the standard HTML version, not the 'one big page' or the PDF. For the next few paragraphs to make sense you will need to actually do this as you read.

Under the heading **Top** you will see five links. Select the link to the *Backend*, which is where information about layout objects is to be found. There, under the heading **Backend**, select the link to *All layout objects*. The page that appears lists all the layout objects used in your version of LilyPond, in alphabetic order. Select the link to Slur, and the properties of Slurs are listed.

An alternative way of finding this page is from the Notation Reference. On one of the pages that deals with slurs you may find a link to the Internals Reference. This link will take you directly to this page, but if you have an idea about the name of the layout object to be tweaked, it is easier to go straight to the IR and search there.

This Slur page in the IR tells us first that Slur objects are created by the Slur_engraver. Then it lists the standard settings. Note these are **not** in alphabetic order. Browse down them looking for a property that might control the heaviness of slurs, and you should find

```
thickness (number)
     1.2
     Line thickness, generally measured in line-thickness
```

This looks a good bet to change the heaviness. It tells us that the value of `thickness` is a simple *number*, that the default value is 1.2, and that the units are in another property called `line-thickness`.

As we said earlier, there are few to no explanations in the IR, but we already have enough information to try changing the slur thickness. We see that the name of the layout object is `Slur`, that the name of the property to change is `thickness` and that the new value should be a number somewhat larger than 1.2 if we are to make slurs thicker.

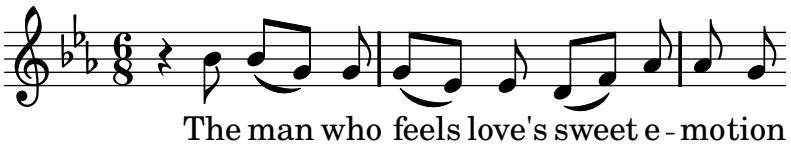
We can now construct the `\override` command by simply substituting the values we have found for the names, omitting the context. Let's use a very large value for the thickness at first, so we can be sure the command is working. We get:

```
\override Slur #'thickness = #5.0
```

Don't forget the `#'` preceding the property name and a `#` preceding the new value!

The final question is, 'Where should this command be placed?' While you are unsure and learning, the best answer is, 'Within the music, before the first slur and close to it.' Let's do that:

```
{
  \key es \major
  \time 6/8
  {
    % Increase thickness of all following slurs from 1.2 to 5.0
    \override Slur #'thickness = #5.0
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



and we see that the slur is indeed heavier.

So this is the basic way of constructing `\override` commands. There are a few more complications that we shall meet in later sections, but you now know all the essentials required to make up your own – but you will still need some practice. This is provided in the examples which follow.

Finding the context
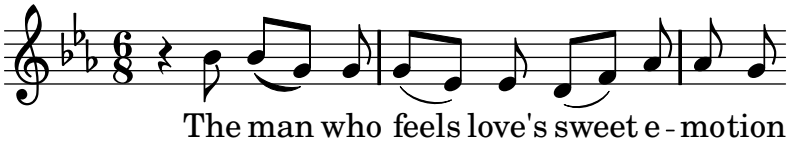
But first, what if we had needed to specify the Context? What should it be? We could guess that slurs are in the Voice context, as they are clearly closely associated with individual lines of music, but can we be sure? To find out, go back to the top of the IR page describing the Slur, where it says 'Slur objects are created by: Slur engraver'. So slurs will be created in whichever context the `Slur_engraver` is in. Follow the link to the `Slur_engraver` page. At the very bottom it tells us that `Slur_engraver` is part of five Voice contexts, including the standard voice context, `Voice`, so our guess was correct. And because `Voice` is one of the lowest level contexts which is implied unambiguously by the fact that we are entering notes, we can omit it in this location.

Overriding once only

As you can see, *all* the slurs are thicker in the final example above. But what if we wanted just the first slur to be thicker? This is achieved with the `\once` command. Placed immediately before

the `\override` command it causes it to change only the slur which begins on the **immediately following** note. If the immediately following note does not begin a slur the command has no effect at all – it is not remembered until a slur is encountered, it is simply discarded. So the command with `\once` must be repositioned as follows:

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    bes8[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



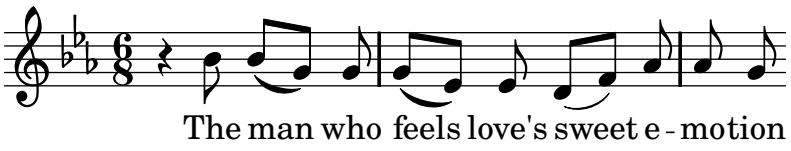
Now only the first slur is made heavier.

The `\once` command can also be used before the `\set` command.

Reverting

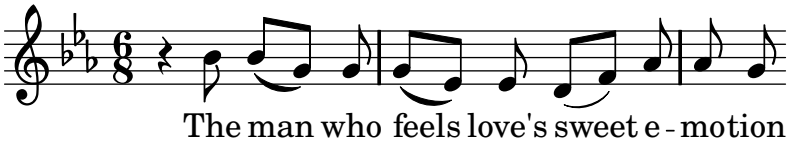
Finally, what if we wanted just the first two slurs to be heavier? Well, we could use two commands, each preceded by `\once` placed immediately before each of the notes where the slurs begin:

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    bes[( g]) g |
    % Increase thickness of immediately following slur only
    \once \override Slur #'thickness = #5.0
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



or we could omit the `\once` command and use the `\revert` command to return the `thickness` property to its default value after the second slur:

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8
    % Increase thickness of all following slurs from 1.2 to 5.0
    \override Slur #'thickness = #5.0
    bes[( g]) g |
    g8[( es]) es
    % Revert thickness of all following slurs to default of 1.2
    \revert Slur #'thickness
    d8[( f]) as |
    as8 g
  }
  \addlyrics {
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



The `\revert` command can be used to return any property changed with `\override` back to its default value. You may use whichever method best suits what you want to do.

That concludes our introduction to the IR, and the basic method of tweaking. Several examples follow in the later sections of this Chapter, partly to introduce you to some of the additional features of the IR, and partly to give you more practice in extracting information from it. These examples will contain progressively fewer words of guidance and explanation.

4.2.2 Properties found in interfaces

Suppose now that we wish to print the lyrics in italics. What form of `\override` command do we need to do this? We first look in the IR page listing 'All layout objects', as before, and look for an object that might control lyrics. We find `LyricText`, which looks right. Clicking on this shows the settable properties for lyric text. These include the `font-series` and `font-size`, but nothing that might give an italic shape. This is because the shape property is one that is common to all font objects, so, rather than including it in every layout object, it is grouped together with other similar common properties and placed in an **Interface**, the `font-interface`.

So now we need to learn how to find the properties of interfaces, and to discover what objects use these interface properties.

Look again at the IR page which describes LyricText. At the bottom of the page is a list of clickable interfaces which LyricText supports. The list has several items, including `font-interface`. Clicking on this brings up the properties associated with this interface, which are also properties of all the objects which support it, including LyricText.

Now we see all the user-settable properties which control fonts, including `font-shape(symbol)`, where `symbol` can be set to `upright`, `italics` or `caps`.

You will notice that `font-series` and `font-size` are also listed there. This immediately raises the question: Why are the common font properties `font-series` and `font-size` listed under `LyricText` as well as under the interface `font-interface` but `font-shape` is not? The answer is that `font-series` and `font-size` are changed from their global default values when a `LyricText` object is created, but `font-shape` is not. The entries in `LyricText` then tell you the values for those two properties which apply to `LyricText`. Other objects which support `font-interface` will set these properties differently when they are created.

Let's see if we can now construct the `\override` command to change the lyrics to italics. The object is `LyricText`, the property is `font-shape` and the value is `italic`. As before, we'll omit the context.
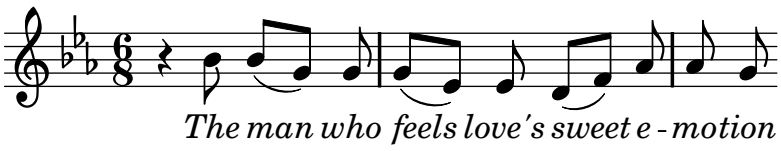
As an aside, although it is an important one, note that because the values of `font-shape` are symbols they must be introduced with a single apostrophe, `'`. That is why apostrophes are needed before `thickness` in the earlier example and `font-shape`. These are both symbols too. Symbols are then read internally by LilyPond. Some of them are the names of properties, like `thickness` or `font-shape`, others are used as values that can be given to properties, like `italic`. Note the distinction from arbitrary text strings, which would appear as `"a text string"`; for more details about symbols and strings, see Sezione "Scheme tutorial" in *Estendere*.

So we see that the `\override` command needed to print the lyrics in italics is:

```
\override LyricText #'font-shape = #'italic
```

This should be placed just in front of the lyrics we wish to affect, like so:

```
{
  \key es \major
  \time 6/8
  {
    r4 bes8 bes[( g]) g |
    g8[( es]) es d[( f]) as |
    as8 g
  }
  \addlyrics {
    \override LyricText #'font-shape = #'italic
    The man who | feels love's sweet e -- | mo -- tion
  }
}
```



and the lyrics are all printed in italics.

Specifying the context in lyric mode

In the case of lyrics, if you try specifying the context in the format given earlier the command will fail. A syllable entered in lyricmode is terminated by either a space, a newline or a digit. All other characters are included as part of the syllable. For this reason a space or newline must appear before the terminating `}` to prevent it being included as part of the final syllable. Similarly, spaces must be inserted before and after the period or dot, '.', separating the context name from the object name, as otherwise the two names are run together and the interpreter cannot recognize them. So the command should be:

```
\override Lyrics . LyricText #'font-shape = #'italic
```

> **Nota:** In lyrics always leave whitespace between the final syllable and the terminating brace.

> **Nota:** In overrides in lyrics always place spaces around the dot between the context name and the object name.

Vedi anche

Extending: Sezione "Scheme tutorial" in *Estendere*.

4.2.3 Types of properties

So far we have seen two types of property: `number` and `symbol`. To be valid, the value given to a property must be of the correct type and obey the rules for that type. The type of property is always shown in brackets after the property name in the IR. Here is a list of the types you may need, together with the rules for that type, and some examples. You must always add a hash symbol, `#`, of course, to the front of these values when they are entered in the `\override` command.

Property type	**Rules**	**Examples**
Boolean	Either True or False, represented by #t or #f	`#t`, `#f`
Dimension (in staff space)	A positive decimal number (in units of staff space)	`2.5`, `0.34`
Direction	A valid direction constant or its numerical equivalent (decimal values between -1 and 1 are allowed)	`LEFT`, `CENTER`, `UP`, `1`, `-1`
Integer	A positive whole number	`3`, `1`
List	A set of values separated by spaces, enclosed in parentheses and preceded by an apostrophe	`'(left-edge staff-bar)`, `'(1)`, `'(1.0 0.25 0.5)`
Markup	Any valid markup	`\markup { \italic "cresc." }`
Moment	A fraction of a whole note constructed with the make-moment function	`(ly:make-moment 1 4)`, `(ly:make-moment 3 8)`
Number	Any positive or negative decimal value	`3.5`, `-2.45`
Pair (of numbers)	Two numbers separated by a 'space . space' and enclosed in brackets preceded by an apostrophe	`'(2 . 3.5)`, `'(0.1 . -3.2)`
Symbol	Any of the set of permitted symbols for that property, preceded by an apostrophe	`'italic`, `'inside`
Unknown	A procedure, or `#f` to cause no action	`bend::print`, `ly:text-interface::print`, `#f`
Vector	A list of three items enclosed in parentheses and preceded by apostrophe-hash, `'#`.	`'#(#t #t #f)`

Vedi anche

Extending: Sezione “Scheme tutorial” in *Estendere*.

4.3 Appearance of objects

Let us now put what we have learned into practice with a few examples which show how tweaks may be used to change the appearance of the printed music.

4.3.1 Visibility and color of objects

In the educational use of music we might wish to print a score with certain elements omitted as an exercise for the student, who is required to supply them. As a simple example, let us suppose the exercise is to supply the missing bar lines in a piece of music. But the bar lines are normally inserted automatically. How do we prevent them printing?
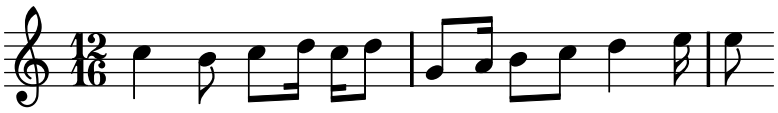
Before we tackle this, let us remember that object properties are grouped in what are called *interfaces* – see Sezione 4.2.2 [Properties found in interfaces], pagina 95. This is simply to group together those properties that may be used together to tweak a graphical object – if one of them is allowed for an object, so are the others. Some objects then use the properties in some interfaces, others use them from other interfaces. The interfaces which contain the properties used by a particular grob are listed in the IR at the bottom of the page describing that grob, and those properties may be viewed by looking at those interfaces.

We explained how to find information about grobs in Sezione 4.2.1 [Properties of layout objects], pagina 91. Using the same approach, we go to the IR to find the layout object which prints bar lines. Going via *Backend* and *All layout objects* we find there is a layout object called `BarLine`. Its properties include two that control its visibility: `break-visibility` and `stencil`. Barline also supports a number of interfaces, including the `grob-interface`, where we find the `transparent` and the `color` properties. All of these can affect the visibility of bar lines (and, of course, by extension, many other layout objects too.) Let’s consider each of these in turn.

stencil
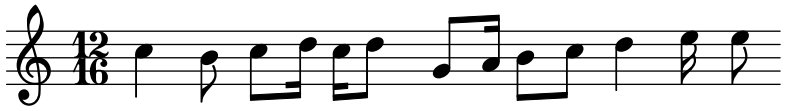
This property controls the appearance of the bar lines by specifying the symbol (glyph) which should be printed. In common with many other properties, it can be set to print nothing by setting its value to `#f`. Let’s try it, as before, omitting the implied Context, `Voice`:

```
{
  \time 12/16
  \override BarLine #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



The bar lines are still printed. What is wrong? Go back to the IR and look again at the page giving the properties of BarLine. At the top of the page it says “Barline objects are created by: Bar_engraver”. Go to the `Bar_engraver` page. At the bottom it gives a list of Contexts in which the bar engraver operates. All of them are of the type `Staff`, so the reason the `\override` command failed to work as expected is because `Barline` is not in the default `Voice` context. If the context is specified incorrectly, the command simply does not work. No error message is produced, and nothing is logged in the log file. Let’s try correcting it by adding the correct context:
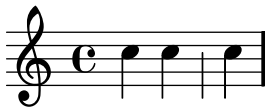
```
{
  \time 12/16
  \override Staff.BarLine #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Now the bar lines have vanished.

Note, though, that setting the `stencil` property to `#f` will cause errors when the dimensions of the object are required for correct processing. For example, errors will be generated if the `stencil` property of the `NoteHead` object is set to `#f`. If this is the case, you can instead use the `point-stencil` function, which sets the stencil to a object with zero size:
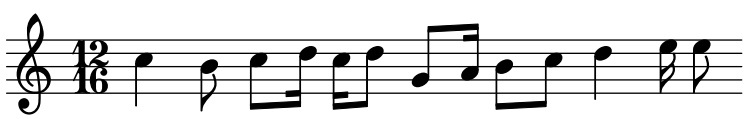
```
{
  c4 c
  \once \override NoteHead #'stencil = #point-stencil
  c4 c
}
```



break-visibility

We see from the `BarLine` properties in the IR that the `break-visibility` property requires a vector of three booleans. These control respectively whether bar lines are printed at the end of a line, in the middle of lines, and at the beginning of lines. For our example we want all bar lines to be suppressed, so the value we need is `'#(#f #f #f)`. Let's try that, remembering to include the `Staff` context. Note also that in writing this value we have `#'#` before the opening bracket. The `'#` is required as part of the value to introduce a vector, and the first `#` is required, as always, to precede the value itself in the `\override` command.

```
{
  \time 12/16
  \override Staff.BarLine #'break-visibility = #'#(#f #f #f)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
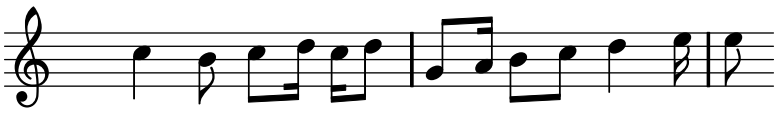


And we see this too removes all the bar lines.

transparent
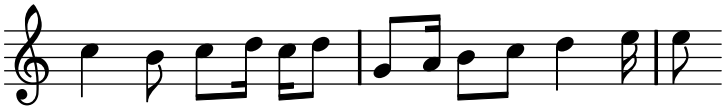
We see from the properties specified in the `grob-interface` page in the IR that the `transparent` property is a boolean. This should be set to `#t` to make the grob transparent. In this next example let us make the time signature invisible rather than the bar lines. To do this we need to find the grob name for the time signature. Back to the 'All layout objects' page in the IR to find the properties of the `TimeSignature` layout object. This is produced by the `Time_signature_engraver` which you can check also lives in the `Staff` context and also supports the `grob-interface`. So the command to make the time signature transparent is:

```
{
  \time 12/16
  \override Staff.TimeSignature #'transparent = ##t
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



The time signature is gone, but this command leaves a gap where the time signature should be. Maybe this is what is wanted for an exercise for the student to fill it in, but in other circumstances a gap might be undesirable. To remove it, the stencil for the time signature should be set to `#f` instead:

```
{
  \time 12/16
  \override Staff.TimeSignature #'stencil = ##f
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
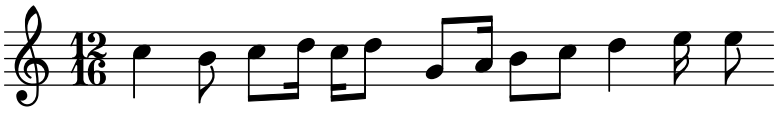


and the difference is obvious: setting the stencil to `#f` removes the object entirely; making the object `transparent` leaves it where it is, but makes it invisible.

color

Finally let us try making the bar lines invisible by coloring them white. (There is a difficulty with this in that the white bar line may or may not blank out the staff lines where they cross. You may see in some of the examples below that this happens unpredictably. The details of why this is so and how to control it are covered in Sezione "Painting objects white" in *Guida alla Notazione*. But at the moment we are learning about color, so please just accept this limitation for now.)

The `grob-interface` specifies that the color property value is a list, but there is no explanation of what that list should be. The list it requires is actually a list of values in internal units, but, to avoid having to know what these are, several ways are provided to specify colors. The first way is to use one of the 'normal' colors listed in the first table in Sezione "List of colors" in *Guida alla Notazione*. To set the bar lines to white we write:
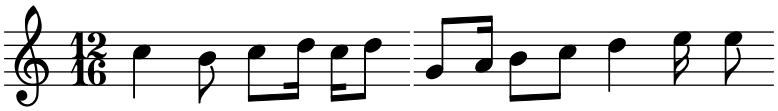
```
{
  \time 12/16
  \override Staff.BarLine #'color = #white
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



and again, we see the bar lines are not visible. Note that *white* is not preceded by an apostrophe – it is not a symbol, but a *function*. When called, it provides the list of internal values required to set the color to white. The other colors in the normal list are functions too. To convince yourself this is working you might like to change the color to one of the other functions in the list.

The second way of changing the color is to use the list of X11 color names in the second list in Sezione "List of colors" in *Guida alla Notazione*. However, these must be preceded by another function, which converts X11 color names into the list of internal values, `x11-color`, like this:
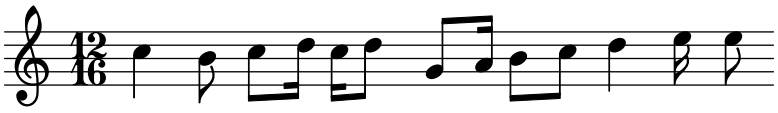
```
{
  \time 12/16
  \override Staff.BarLine #'color = #(x11-color 'white)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Note that in this case the function `x11-color` takes a symbol as an argument, so the symbol must be preceded by an apostrophe and the two enclosed in brackets.
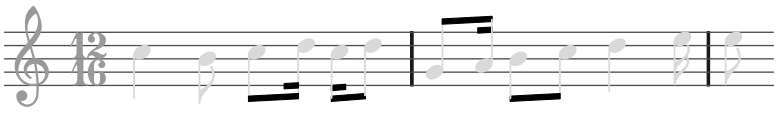
There is yet a third function, one which converts RGB values into internal colors – the `rgb-color` function. This takes three arguments giving the intensities of the red, green and blue colors. These take values in the range 0 to 1. So to set the color to red the value should be `(rgb-color 1 0 0)` and to white it should be `(rgb-color 1 1 1)`:

```
{
  \time 12/16
  \override Staff.BarLine #'color = #(rgb-color 1 1 1)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```



Finally, there is also a grey scale available as part of the X11 set of colors. These range from black, `'grey0'`, to white, `'grey100`, in steps of 1. Let's illustrate this by setting all the layout objects in our example to various shades of grey:

```
{
  \time 12/16
  \override Staff.StaffSymbol   #'color = #(x11-color 'grey30)
  \override Staff.TimeSignature #'color = #(x11-color 'grey60)
  \override Staff.Clef          #'color = #(x11-color 'grey60)
  \override Voice.NoteHead      #'color = #(x11-color 'grey85)
  \override Voice.Stem          #'color = #(x11-color 'grey85)
  \override Staff.BarLine       #'color = #(x11-color 'grey10)
  c4 b8 c d16 c d8 |
  g,8 a16 b8 c d4 e16 |
  e8
}
```
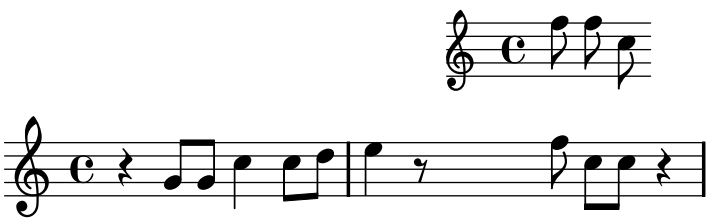


Note the contexts associated with each of the layout objects. It is important to get these right, or the commands will not work! Remember, the context is the one in which the appropriate engraver is placed. The default context for engravers can be found by starting from the layout object, going from there to the engraver which produces it, and on the engraver page in the IR it tells you in which context the engraver will normally be found.

4.3.2 Size of objects
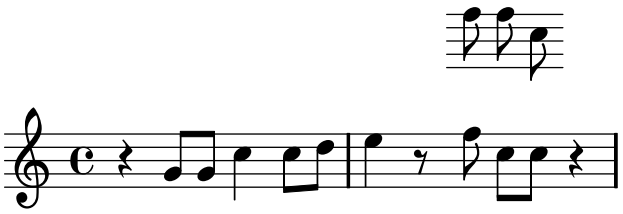
Let us begin by looking again at the earlier example see ⟨undefined⟩ [Nesting music expressions], pagina ⟨undefined⟩) which showed how to introduce a new temporary staff, as in an Sezione "ossia" in *Glossario Musicale*.

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main" }
      { f8 f c }
    >>
    r4 |
  }
}
```



Ossia are normally written without clef and time signature, and are usually printed slightly smaller than the main staff. We already know now how to remove the clef and time signature – we simply set the stencil of each to `#f`, as follows:

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
      }
      {
        \override Staff.Clef #'stencil = ##f
        \override Staff.TimeSignature #'stencil = ##f
        { f8 f c }
      }
    >>
    r4 |
  }
}
```



where the extra pair of braces after the `\with` clause are required to ensure the enclosed overrides and music are applied to the ossia staff.

But what is the difference between modifying the staff context by using `\with` and modifying the stencils of the clef and the time signature with \override? The main difference is that changes made in a `\with` clause are made at the time the context is created, and remain in force as the **default** values for the duration of that context, whereas `\set` or `\override` commands embedded in the music are dynamic – they make changes synchronized with a particular point in the music. If changes are unset or reverted using `\unset` or `\revert` they return to their default values, which will be the ones set in the `\with` clause, or if none have been set there, the normal default values.
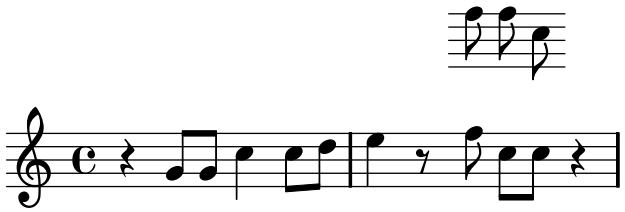
Some context properties can be modified only in `\with` clauses. These are those properties which cannot sensibly be changed after the context has been created. `alignAboveContext` and its partner, `alignBelowContext`, are two such properties – once the staff has been created its alignment is decided and it would make no sense to try to change it later.

The default values of layout object properties can also be set in `\with` clauses. Simply use the normal `\override` command leaving out the context name, since this is unambiguously defined as the context which the `\with` clause is modifying. If fact, an error will be generated if a context is specified in this location.

So we could replace the example above with

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
```

```
        \new Staff \with {
          alignAboveContext = #"main"
          % Don't print clefs in this staff
          \override Clef #'stencil = ##f
          % Don't print time signatures in this staff
          \override TimeSignature #'stencil = ##f
        }
        { f8 f c }
      >>
      r4 |
    }
}
```



Finally we come to changing the size of layout objects.

Some layout objects are created as glyphs selected from a typeface font. These include note heads, accidentals, markup, clefs, time signatures, dynamics and lyrics. Their size is changed by modifying the `font-size` property, as we shall shortly see. Other layout objects such as slurs and ties – in general, spanner objects – are drawn individually, so there is no `font-size` associated with them. These objects generally derive their size from the objects to which they are attached, so usually there is no need to change their size manually. Still other properties such as the length of stems and bar lines, thickness of beams and other lines, and the separation of staff lines all need to be modified in special ways.

Returning to the ossia example, let us first change the font-size. We can do this in two ways. We can either change the size of the fonts of each object type, like `NoteHead`s with commands like

```
\override NoteHead #'font-size = #-2
```

or we can change the size of all fonts by setting a special property, `fontSize`, using `\set`, or by including it in a `\with` clause (but without the `\set`).
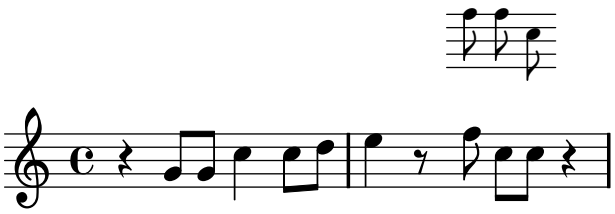
```
\set fontSize = #-2
```

Both of these statements would cause the font size to be reduced by 2 steps from its previous value, where each step reduces or increases the size by approximately 12%.

Let's try it in our ossia example:

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        % Reduce all font sizes by ~24%
```

```
        fontSize = #-2
      }
      { f8 f c }
    >>
    r4 |
  }
}
```
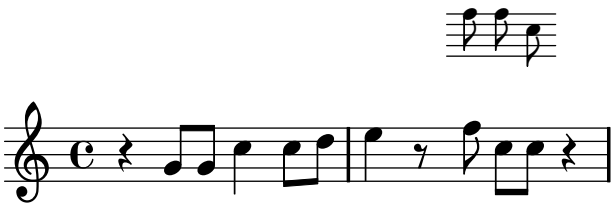


This is still not quite right. The note heads and flags are smaller, but the stems are too long in proportion and the staff lines are spaced too widely apart. These need to be scaled down in proportion to the font reduction. The next sub-section discusses how this is done.

4.3.3 Length and thickness of objects

Distances and lengths in LilyPond are generally measured in staff-spaces, the distance between adjacent lines in the staff, (or occasionally half staff spaces) while most `thickness` properties are measured in units of an internal property called `line-thickness`. For example, by default, the lines of hairpins are given a thickness of 1 unit of `line-thickness`, while the `thickness` of a note stem is 1.3. Note, though, that some thickness properties are different; for example, the thickness of beams is controlled by the value of the `beam-thickness` property, which is measured in staff-spaces.

So how are lengths to be scaled in proportion to the font size? This can be done with the help of a special function called `magstep` provided for exactly this purpose. It takes one argument, the change in font size (#-2 in the example above) and returns a scaling factor suitable for reducing other objects in proportion. It is used like this:

```
\new Staff ="main" {
  \relative g' {
    r4 g8 g c4 c8 d |
    e4 r8
    <<
      { f8 c c }
      \new Staff \with {
        alignAboveContext = #"main"
        \override Clef #'stencil = ##f
        \override TimeSignature #'stencil = ##f
        fontSize = #-2
        % Reduce stem length and line spacing to match
        \override StaffSymbol #'staff-space = #(magstep -2)
      }
      { f8 f c }
    >>
    r4 |
  }
}
```



Since the length of stems and many other length-related properties are always calculated relative to the value of the `staff-space` property these are automatically scaled down in length too. Note that this affects only the vertical scale of the ossia – the horizontal scale is determined by the layout of the main music in order to remain synchronized with it, so it is not affected by any of these changes in size. Of course, if the scale of all the main music were changed in this way then the horizontal spacing would be affected. This is discussed later in the layout section.

This, then, completes the creation of an ossia. The sizes and lengths of all other objects may be modified in analogous ways.

For small changes in scale, as in the example above, the thickness of the various drawn lines such as bar lines, beams, hairpins, slurs, etc does not usually require global adjustment. If the thickness of any particular layout object needs to be adjusted this can be best achieved by overriding its `thickness` property. An example of changing the thickness of slurs was shown above in Sezione 4.2.1 [Properties of layout objects], pagina 91. The thickness of all drawn objects (i.e., those not produced from a font) may be changed in the same way.

4.4 Placement of objects

4.4.1 Automatic behavior

There are some objects in musical notation that belong to the staff and there are other objects that should be placed outside the staff. These are called within-staff objects and outside-staff objects respectively.
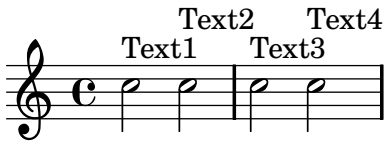
Within-staff objects are those that are located on the staff – note heads, stems, accidentals, etc. The positions of these are usually fixed by the music itself – they are vertically positioned on specific lines of the staff or are tied to other objects that are so positioned. Collisions of note heads, stems and accidentals in closely set chords are normally avoided automatically. There are commands and overrides which can modify this automatic behavior, as we shall shortly see.

Objects belonging outside the staff include things such as rehearsal marks, text and dynamic markings. LilyPond's rule for the vertical placement of outside-staff objects is to place them as close to the staff as possible but not so close that they collide with any other object. LilyPond uses the `outside-staff-priority` property to determine the order in which the objects should be placed, as follows.

First, LilyPond places all the within-staff objects. Then it sorts the outside-staff objects according to their `outside-staff-priority`. The outside-staff objects are taken one by one, beginning with the object with the lowest `outside-staff-priority`, and placed so that they do not collide with any objects that have already been placed. That is, if two outside-staff grobs are competing for the same space, the one with the lower `outside-staff-priority` will be placed closer to the staff. If two objects have the same `outside-staff-priority` the one encountered first will be placed closer to the staff.

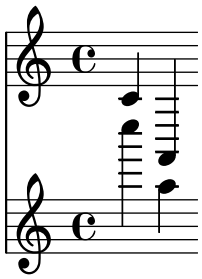
In the following example all the markup texts have the same priority (since it is not explicitly set). Note that 'Text3' is automatically positioned close to the staff again, nestling under 'Text2'.

```
c2^"Text1"
c2^"Text2" |
c2^"Text3"
c2^"Text4" |
```



Staves are also positioned, by default, as closely together as possible (subject to a minimum separation). If notes project a long way towards an adjacent staff they will force the staves further apart only if an overlap of the notation would otherwise occur. The following example demonstrates this 'nestling' of the notes on adjacent staves:

```
<<
  \new Staff {
    \relative c' { c4 a, }
  }
  \new Staff {
    \relative c'''' { c4 a, }
  }
>>
```
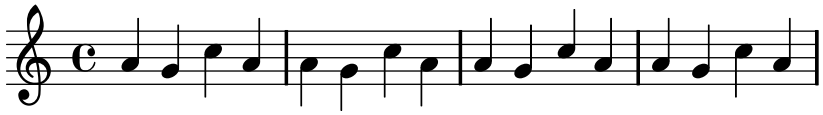


4.4.2 Within-staff objects

We have already seen how the commands `\voiceXXX` affect the direction of slurs, ties, fingering and everything else which depends on the direction of the stems. These commands are essential when writing polyphonic music to permit interweaving melodic lines to be distinguished. But occasionally it may be necessary to override this automatic behavior. This can be done for whole sections of music or even for an individual note. The property which controls this behavior is the `direction` property of each layout object. We first explain what this does, and then introduce a number of ready-made commands which avoid your having to code explicit overrides for the more common modifications.

Some layout objects like slurs and ties curve, bend or point either up or down; others like stems and flags also move to right or left when they point up or down. This is controlled automatically when `direction` is set.

The following example shows in bar 1 the default behavior of stems, with those on high notes pointing down and those on low notes pointing up, followed by four notes with all stems forced down, four notes with all stems forced up, and finally four notes reverted back to the default behavior.

```
a4 g c a |
\override Stem #'direction = #DOWN
a4 g c a |
\override Stem #'direction = #UP
a4 g c a |
\revert Stem #'direction
a4 g c a |
```



Here we use the constants `DOWN` and `UP`. These have the values `-1` and `+1` respectively, and these numerical values may be used instead. The value `0` may also be used in some cases. It is simply treated as meaning `UP` for stems, but for some objects it means 'center'. There is a constant, `CENTER` which has the value `0`.

However, these explicit overrides are not usually used, as there are simpler equivalent predefined commands available. Here is a table of the commonest. The meaning of each is stated where it is not obvious.

Down/Left	**Up/Right**	**Revert**	**Effect**
`\arpeggioArrowDown`	`\arpeggioArrowUp`	`\arpeggioNormal`	Arrow is at bottom, at top, or no arrow
`\dotsDown`	`\dotsUp`	`\dotsNeutral`	Direction of movement to avoid staff lines
`\dynamicDown`	`\dynamicUp`	`\dynamicNeutral`	
`\phrasingSlurDown`	`\phrasingSlurUp`	`\phrasingSlurNeutral`	Note: distinct from slur commands
`\slurDown`	`\slurUp`	`\slurNeutral`	
`\stemDown`	`\stemUp`	`\stemNeutral`	
`\textSpannerDown`	`\textSpannerUp`	`\textSpannerNeutral`	Text entered as spanner is below/above staff
`\tieDown`	`\tieUp`	`\tieNeutral`	
`\tupletDown`	`\tupletUp`	`\tupletNeutral`	Tuplets are below/above notes

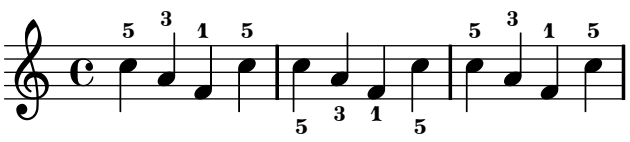
Note that these predefined commands may **not** be preceded by `\once`. If you wish to limit the effect to a single note you must either use the equivalent `\once \override` command or use the predefined command followed after the affected note by the corresponding `\xxxNeutral` command.

Fingering

The placement of fingering on single notes can also be controlled by the `direction` property, but changing `direction` has no effect on chords. As we shall see, there are special commands which allow the fingering of individual notes of chords to be controlled, with the fingering being placed above, below, to the left or to the right of each note.

First, here's the effect of `direction` on the fingering attached to single notes. The first bar shows the default behaviour, and the following two bars shows the effect of specifying `DOWN` and `UP`:
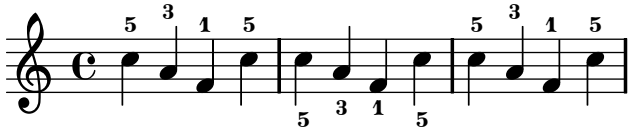
```
c4-5 a-3 f-1 c'-5 |
\override Fingering #'direction = #DOWN
c4-5 a-3 f-1 c'-5 |
\override Fingering #'direction = #UP
c4-5 a-3 f-1 c'-5 |
```



However, overriding the `direction` property is not the easiest way of manually setting the fingering above or below the notes; using `_` or `^` instead of `-` before the fingering number is usually preferable. Here is the previous example using this method:

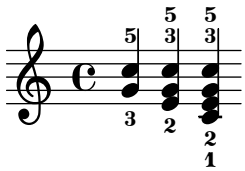
```
c4-5 a-3 f-1 c'-5 |
```

```
c4_5 a_3 f_1 c'_5 |
c4^5 a^3 f^1 c'^5 |
```



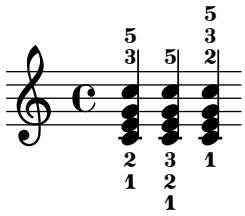
The `direction` property is ignored for chords, but the directional prefixes, `_` and `^` do work. By default, the fingering is automatically placed both above and below the notes of a chord, as shown:

```
<c-5 g-3>4
<c-5 g-3 e-2>4
<c-5 g-3 e-2 c-1>4
```



but this may be overridden to manually force all or any of the individual fingering numbers above or below:

```
<c-5 g-3 e-2 c-1>4
<c^5 g_3 e_2 c_1>4
<c^5 g^3 e^2 c_1>4
```



Even greater control over the placement of fingering of the individual notes in a chord is possible by using the `\set fingeringOrientations` command. The format of this command is:

```
\set fingeringOrientations = #'([up] [left/right] [down])
```

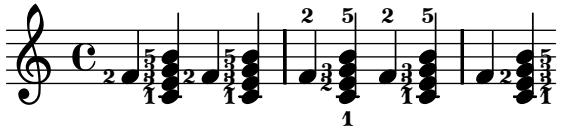
`\set` is used because `fingeringOrientations` is a property of the `Voice` context, created and used by the `New_fingering_engraver`.

The property may be set to a list of one to three values. It controls whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears, or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingering may be placed only on one side or the other, not both.

> **Nota:** To control the placement of the fingering of a single note using this command it is necessary to write it as a single note chord by placing angle brackets round it.

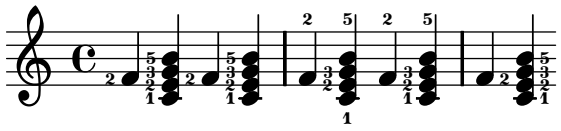
Here are a few examples:

```
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(up left down)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(up left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(right)
<f-2>4
<c-1 e-2 g-3 b-5>4
```



If the fingering seems a little crowded the `font-size` could be reduced. The default value can be seen from the `Fingering` object in the IR to be `-5`, so let's try `-7`:

```
\override Fingering #'font-size = #-7
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(up left down)
<f-2>4
<c-1 e-2 g-3 b-5>4
\set fingeringOrientations = #'(up left)
<f-2>4
<c-1 e-2 g-3 b-5>4 |
\set fingeringOrientations = #'(right)
<f-2>4
<c-1 e-2 g-3 b-5>4
```



4.4.3 Outside-staff objects

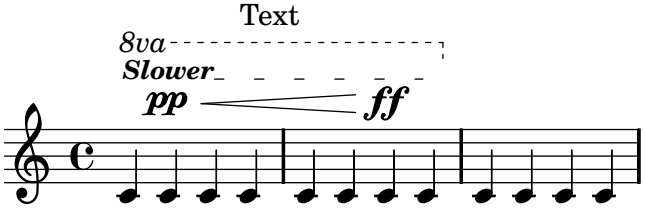
Outside-staff objects are automatically placed to avoid collisions. Objects with the lower value of the `outside-staff-priority` property are placed nearer to the staff, and other outside-staff objects are then raised as far as necessary to avoid collisions. The `outside-staff-priority` is defined in the `grob-interface` and so is a property of all layout objects. By default it is set to `#f` for all within-staff objects, and to a numerical value appropriate to each outside-staff object when the object is created. The following table shows the default numerical values for some of the commonest outside-staff objects.

Note the unusual names for some of the objects: spanner objects are automatically created to control the vertical positioning of grobs which (might) start and end at different musical moments, so changing the `outside-staff-priority` of the underlying grob will have no effect. For example, changing `outside-staff-priority` of the `Hairpin` object will have no effect on the vertical positioning of hairpins – you must change `outside-staff-priority` of the associated `DynamicLineSpanner` object instead. This override must be placed at the start of the spanner, which might include several linked hairpins and dynamics.

Layout Object	**Priority**	**Controls position of:**
`RehearsalMark`	1500	Rehearsal marks
`MetronomeMark`	1000	Metronome marks
`VoltaBracketSpanner`	600	Volta brackets
`TextScript`	450	Markup text
`MultiMeasureRestText`	450	Markup text over full-bar rests
`OttavaBracket`	400	Ottava brackets
`TextSpanner`	350	Text spanners
`DynamicLineSpanner`	250	All dynamic markings
`BarNumber`	100	Bar numbers
`TrillSpanner`	50	Spanning trills

Here is an example showing the default placement of some of these.

```
% Set details for later Text Spanner
\override TextSpanner #'(bound-details left text)
    = \markup { \small \bold Slower }
% Place dynamics above staff
\dynamicUp
% Start Ottava Bracket
\ottava #1
c'4 \startTextSpan
% Add Dynamic Text and hairpin
c4\pp\<
c4
% Add Text Script
c4^Text |
c4 c
% Add Dynamic Text and terminate hairpin
c4\ff c \stopTextSpan |
% Stop Ottava Bracket
\ottava #0
c,4 c c c |
```

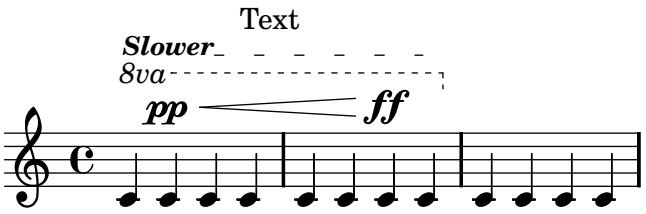


This example also shows how to create Text Spanners – text with extender lines above a section of music. The spanner extends from the `\startTextSpan` command to the `\stopTextSpan` command, and the format of the text is defined by the `\override TextSpanner` command. For more details see Sezione “Text spanners” in *Guida alla Notazione*.

It also shows how ottava brackets are created.

If the default values of `outside-staff-priority` do not give you the placing you want, the priority of any of the objects may be overridden. Suppose we would like the ottava bracket to be placed below the text spanner in the example above. All we need to do is to look up the priority of `OttavaBracket` in the IR or in the tables above, and reduce it to a value lower than that of a `TextSpanner`, remembering that `OttavaBracket` is created in the `Staff` context:

```
% Set details for later Text Spanner
\override TextSpanner #'(bound-details left text)
    = \markup { \small \bold Slower }
% Place dynamics above staff
\dynamicUp
% Place following Ottava Bracket below Text Spanners
\once \override Staff.OttavaBracket #'outside-staff-priority = #340
% Start Ottava Bracket
\ottava #1
c'4 \startTextSpan
% Add Dynamic Text
c4\pp
% Add Dynamic Line Spanner
c4\<
% Add Text Script
c4^Text |
c4 c
% Add Dynamic Text
c4\ff c \stopTextSpan |
% Stop Ottava Bracket
\ottava #0
c,4 c c c |
```
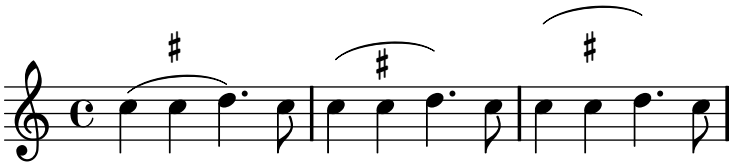


Note that some of these objects, in particular bar numbers, metronome marks and rehearsal marks, live by default in the `Score` context, so be sure to use the correct context when these are being overriden.

Slurs by default are classed as within-staff objects, but they often appear above the staff if the notes to which they are attached are high on the staff. This can push outside-staff objects such as articulations too high, as the slur will be placed first. The `avoid-slur` property of the articulation can be set to `'inside` to bring the articulation inside the slur, but the `avoid-slur` property is effective only if the `outside-staff-priority` is also set to `#f`. Alternatively, the `outside-staff-priority` of the slur can be set to a numerical value to cause it to be placed along with other outside-staff objects according to that value. Here's an example showing the effect of the two methods:

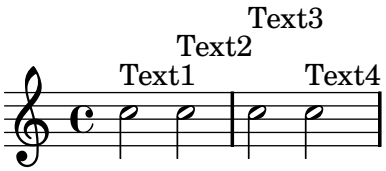
```
c4( c^\markup { \tiny \sharp } d4.) c8 |
c4(
\once \override TextScript #'avoid-slur = #'inside
\once \override TextScript #'outside-staff-priority = ##f
```

```
c4^\markup { \tiny \sharp } d4.) c8 |
\once \override Slur #'outside-staff-priority = #500
c4( c^\markup { \tiny \sharp } d4.) c8 |
```



Changing the `outside-staff-priority` can also be used to control the vertical placement of individual objects, although the results may not always be desirable. Suppose we would like "Text3" to be placed above "Text4" in the example under Automatic behavior, above (see Sezione 4.4.1 [Automatic behavior], pagina 106). All we need to do is to look up the priority of `TextScript` in the IR or in the tables above, and increase the priority of "Text3" to a higher value:

```
c2^"Text1"
c2^"Text2" |
\once \override TextScript #'outside-staff-priority = #500
c2^"Text3"
c2^"Text4" |
```
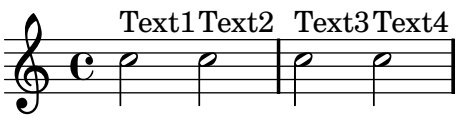


This certainly lifts "Text3" above "Text4" but it also lifts it above "Text2", and "Text4" now drops down. Perhaps this is not so good. What we would really like to do is to position all the annotation at the same distance above the staff. To do this, we clearly will need to space the notes out horizontally to make more room for the text. This is done using the `textLengthOn` command.

\textLengthOn

By default, text produced by markup takes up no horizontal space as far as laying out the music is concerned. The `\textLengthOn` command reverses this behavior, causing the notes to be spaced out as far as is necessary to accommodate the text:

```
\textLengthOn  % Cause notes to space out to accommodate text
c2^"Text1"
c2^"Text2" |
c2^"Text3"
c2^"Text4" |
```



The command to revert to the default behavior is `\textLengthOff`. Remember `\once` only works with `\override`, `\set`, `\revert` or `unset`, so cannot be used with `\textLengthOn`.

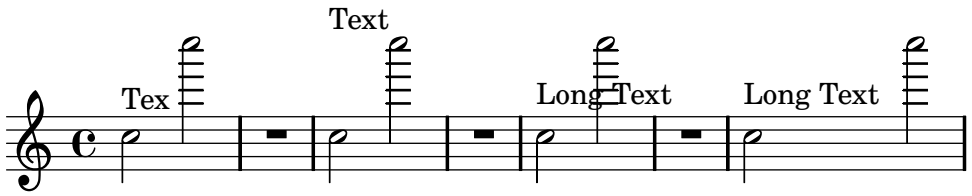
Markup text will also avoid notes which project above the staff. If this is not desired, the automatic displacement upwards may be turned off by setting the priority to `#f`. Here's an example to show how markup text interacts with such notes.

```
% This markup is short enough to fit without collision
c2^"Tex" c'' |
R1 |

% This is too long to fit, so it is displaced upwards
c,,2^"Text" c'' |
R1 |

% Turn off collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
c,,2^"Long Text   " c'' |
R1 |

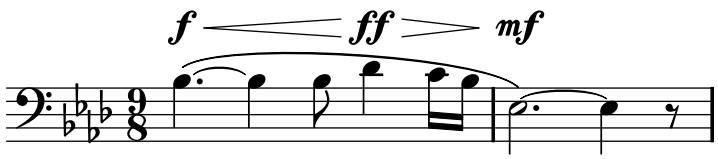
% Turn off collision avoidance
\once \override TextScript #'outside-staff-priority = ##f
\textLengthOn        % and turn on textLengthOn
c,,2^"Long Text   "  % Spaces at end are honored
c''2 |
```



Dynamics
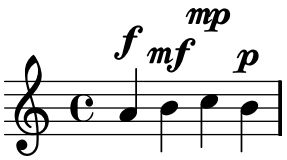
Dynamic markings will normally be positioned beneath the staff, but may be positioned above with the `dynamicUp` command. They will be positioned vertically relative to the note to which they are attached, and will float below (or above) all within-staff objects such as phrasing slurs and bar numbers. This can give quite acceptable results, as this example shows:

```
\clef "bass"
\key aes \major
\time 9/8
\dynamicUp
bes4.~\f\< \( bes4 bes8 des4\ff\> c16 bes\! |
ees,2.~\)\mf ees4 r8 |
```



However, if the notes and attached dynamics are close together the automatic placement will avoid collisions by displacing later dynamic markings further away, but this may not be the optimum placement, as this rather artificial example shows:

```
\dynamicUp
a4\f b\mf c\mp b\p
```



Should a similar situation arise in 'real' music, it may be preferable to space out the notes a little further, so the dynamic markings can all fit at the same vertical distance from the staff. We were able to do this for markup text by using the `\textLengthOn` command, but there is no equivalent command for dynamic marks. So we shall have to work out how to do this using `\override` commands.

Grob sizing
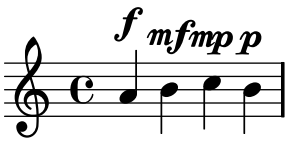
First we must learn how grobs are sized. All grobs have a reference point defined within them which is used to position them relative to their parent object. This point in the grob is then positioned at a horizontal distance, `X-offset`, and at a vertical distance, `Y-offset`, from its parent. The horizontal extent of the object is given by a pair of numbers, `X-extent`, which say where the left and right edges are relative to the reference point. The vertical extent is similarly defined by a pair of numbers, `Y-extent`. These are properties of all grobs which support the `grob-interface`.

By default, outside-staff objects are given a width of zero so that they may overlap in the horizontal direction. This is done by the trick of adding infinity to the leftmost extent and minus infinity to the rightmost extent by setting the `extra-spacing-width` to `'(+inf.0 . -inf.0)`. So to ensure they do not overlap in the horizontal direction we must override this value of `extra-spacing-width` to `'(0 . 0)` so the true width shines through. This is the command to do this for dynamic text:

```
\override DynamicText #'extra-spacing-width = #'(0 . 0)
```
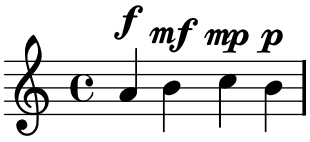
Let's see if this works in our previous example:

```
\dynamicUp
\override DynamicText #'extra-spacing-width = #'(0 . 0)
a4\f b\mf c\mp b\p |
```



Well, it has certainly stopped the dynamic marks being displaced, but two problems remain. The marks should be spaced a little further apart and it would be better if they were all the same distance from the staff. We can solve the first problem easily. Instead of making the `extra-spacing-width` zero we could add a little more to it. The units are the space between two staff lines, so moving the left edge half a unit to the left and the right edge half a unit to the right should do it:

```
\dynamicUp
% Extend width by 1 staff space
\override DynamicText #'extra-spacing-width = #'(-0.5 . 0.5)
a4\f b\mf c\mp b\p
```



This looks better, but maybe we would prefer the dynamic marks to be aligned along the same baseline rather than going up and down with the notes. The property to do this is `staff-padding` which is covered in the following section.

4.5 Collisions of objects

4.5.1 Moving objects

This may come as a surprise, but LilyPond is not perfect. Some notation elements can overlap. This is unfortunate, but in fact rather rare. Usually the need to move objects is for clarity or aesthetic reasons – they would look better with a little more or a little less space around them.

There are three main approaches to resolving overlapping notation. They should be considered in the following order:

1. The **direction** of one of the overlapping objects may be changed using the predefined commands listed above for within-staff objects (see Sezione 4.4.2 [Within-staff objects], pagina 107). Stems, slurs, beams, ties, dynamics, text and tuplets may be repositioned easily in this way. The limitation is that you have a choice of only two positions, and neither may be suitable.
2. The **object properties**, which LilyPond uses when positioning layout objects, may be modified using `\override`. The advantages of making changes to this type of property are (a) that some other objects will be moved automatically if necessary to make room and (b) the single override can apply to all instances of the same type of object. Such properties include:

   - `direction`

     This has already been covered in some detail – see Sezione 4.4.2 [Within-staff objects], pagina 107.

   - `padding`, `right-padding`, `staff-padding`

     As an object is being positioned the value of its `padding` property specifies the gap that must be left between itself and the nearest edge of the object against which it is being positioned. Note that it is the `padding` value of the object **being placed** that is used; the `padding` value of the object which is already placed is ignored. Gaps specified by `padding` can be applied to all objects which support the `side-position-interface`.

     Instead of `padding`, the placement of groups of accidentals is controlled by `right-padding`. This property is to be found in the `AccidentalPlacement` object which, note, lives in the **Staff** context. In the typesetting process the note heads are typeset first and then the accidentals, if any, are added to the left of the note heads using the `right-padding` property to determine the separation from the note heads and between individual accidentals. So only the `right-padding` property of the `AccidentalPlacement` object has any effect on the placement of the accidentals.

     The `staff-padding` property is closely related to the `padding` property: `padding` controls the minimum amount of space between any object which supports the `side-position-interface` and the nearest other object (generally the note or the staff lines); `staff-padding` applies only to those objects which are always set outside the staff – it controls the minimum amount of space that should be inserted between that object and the staff. Note that `staff-padding` has no effect on objects which are positioned relative to the note rather than the staff, even though it may be overridden without error for such objects – it is simply ignored.

     To discover which padding property is required for the object you wish to reposition, you need to return to the IR and look up the object's properties. Be aware that the padding properties might not be located in the obvious object, so look in objects that appear to be related.

     All padding values are measured in staff spaces. For most objects, this value is set by default to be around 1.0 or less (it varies with each object). It may be overridden if a larger (or smaller) gap is required.

- `self-alignment-X`

  This property can be used to align the object to the left, to the right, or to center it with respect to the parent object's reference point. It may be used with all objects which support the `self-alignment-interface`. In general these are objects that contain text. The values are `LEFT`, `RIGHT` or `CENTER`. Alternatively, a numerical value between `-1` and `+1` may be specified, where `-1` is left-aligned, `+1` is right-aligned, and numbers in between move the text progressively from left-aligned to right-aligned. Numerical values greater than `1` may be specified to move the text even further to the left, or less than `-1` to move the text even further to the right. A change of `1` in the value corresponds to a movement of half the text's length.

- `extra-spacing-width`

  This property is available for all objects which support the `item-interface`. It takes two numbers, the first is added to the leftmost extent and the second is added to the rightmost extent. Negative numbers move the edge to the left, positive to the right, so to widen an object the first number must be negative, the second positive. Note that not all objects honor both numbers. For example, the `Accidental` object only takes notice of the first (left edge) number.

- `staff-position`

  `staff-position` is a property of the `staff-symbol-referencer-interface`, which is supported by objects which are positioned relative to the staff. It specifies the vertical position of the object relative to the center line of the staff in half staff-spaces. It is useful in resolving collisions between layout objects like multi-measure rests, ties and notes in different voices.

- `force-hshift`

  Closely spaced notes in a chord, or notes occurring at the same time in different voices, are arranged in two, occasionally more, columns to prevent the note heads overlapping. These are called note columns, and an object called `NoteColumn` is created to lay out the notes in that column.

  The `force-hshift` property is a property of a `NoteColumn` (actually of the `note-column-interface`). Changing it permits a note column to be moved in units appropriate to a note column, viz. the note head width of the first voice note. It should be used in complex situations where the normal `\shiftOn` commands (see ⟨undefined⟩ [Explicitly instantiating voices], pagina ⟨undefined⟩) do not resolve the note conflict. It is preferable to the `extra-offset` property for this purpose as there is no need to work out the distance in staff-spaces, and moving the notes into or out of a `NoteColumn` affects other actions such as merging note heads.

3. Finally, when all else fails, objects may be manually repositioned relative to the staff center line vertically, or by displacing them by any distance to a new position. The disadvantages are that the correct values for the repositioning have to be worked out, often by trial and error, for every object individually, and, because the movement is done after LilyPond has placed all other objects, the user is responsible for avoiding any collisions that might ensue. But the main difficulty with this approach is that the repositioning values may need to be reworked if the music is later modified. The properties that can be used for this type of manual repositioning are:

   `extra-offset`

   This property applies to any layout object supporting the `grob-interface`. It takes a pair of numbers which specify the extra displacement in the horizontal and vertical directions. Negative numbers move the object to the left or down. The units are staff-spaces. The extra displacement is made after the typeset-

ting of objects is finished, so an object may be repositioned anywhere without affecting anything else.

`positions`
: This is most useful for manually adjusting the slope and height of beams, slurs, and tuplets. It takes a pair of numbers giving the position of the left and right ends of the beam, slur, etc. relative to the center line of the staff. Units are staff-spaces. Note, though, that slurs and phrasing slurs cannot be repositioned by arbitrarily large amounts. LilyPond first generates a list of possible positions for the slur and by default finds the slur that "looks best". If the `positions` property has been overridden the slur that is closest to the requested positions is selected from the list.

A particular object may not have all of these properties. It is necessary to go to the IR to look up which properties are available for the object in question.

Here is a list of the objects which are most likely to be involved in collisions, together with the name of the object which should be looked up in the IR in order to discover which properties should be used to move them.

Object type	**Object name**
Articulations	`Script`
Beams	`Beam`
Dynamics (vertically)	`DynamicLineSpanner`
Dynamics (horizontally)	`DynamicText`
Fingerings	`Fingering`
Rehearsal / Text marks	`RehearsalMark`
Slurs	`Slur`
Text e.g. `^"text"`	`TextScript`
Ties	`Tie`
Tuplets	`TupletBracket`

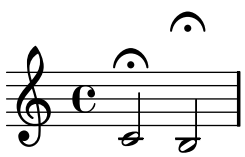
4.5.2 Fixing overlapping notation

Let's now see how the properties in the previous section can help to resolve overlapping notation.

padding property

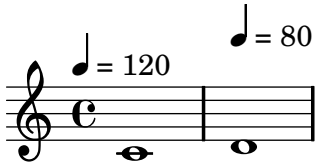
The `padding` property can be set to increase (or decrease) the distance between symbols that are printed above or below notes.

```
c2\fermata
\override Script #'padding = #3
b2\fermata
```



```
% This will not work, see below
\override MetronomeMark #'padding = #3
\tempo 4 = 120
c1 |
% This works
\override Score.MetronomeMark #'padding = #3
```

```
\tempo 4 = 80
d1 |
```
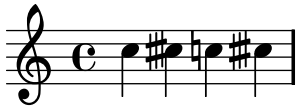


Note in the second example how important it is to figure out what context handles a certain object. Since the `MetronomeMark` object is handled in the `Score` context, property changes in the `Voice` context will not be noticed. For more details, see Sezione "Modifying properties" in *Guida alla Notazione*.

If the `padding` property of an object is increased when that object is in a stack of objects being positioned according to their `outside-staff-priority`, then that object and all objects outside it are moved.

right-padding

The `right-padding` property affects the spacing between the accidental and the note to which it applies. It is not often required, but the default spacing may be wrong for certain special accidental glyphs or combination of glyphs used in some microtonal music. These have to be entered by overriding the accidental stencil with a markup containing the desired symbol(s), like this:

```
sesquisharp = \markup { \sesquisharp }
\relative c'' {
  c4
  % This prints a sesquisharp but the spacing is too small
  \once \override Accidental
    #'stencil = #ly:text-interface::print
  \once \override Accidental #'text = #sesquisharp
  cis4 c
  % This improves the spacing
  \once \override Score.AccidentalPlacement #'right-padding = #0.6
  \once \override Accidental
    #'stencil = #ly:text-interface::print
  \once \override Accidental #'text = #sesquisharp
  cis4 |
}
```
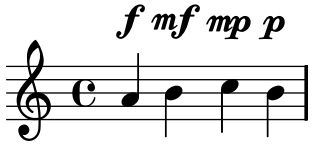


This necessarily uses an override for the accidental stencil which will not be covered until later. The stencil type must be a procedure, here changed to print the contents of the `text` property of `Accidental`, which itself is set to be a sesquisharp sign. This sign is then moved further away from the note head by overriding `right-padding`.

staff-padding property

`staff-padding` can be used to align objects such as dynamics along a baseline at a fixed height above the staff, rather than at a height dependent on the position of the note to which they are attached. It is not a property of `DynamicText` but of `DynamicLineSpanner`. This is because the

baseline should apply equally to **all** dynamics, including those created as extended spanners. So this is the way to align the dynamic marks in the example taken from the previous section:

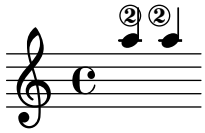
```
\dynamicUp
% Extend width by 1 unit
\override DynamicText #'extra-spacing-width = #'(-0.5 . 0.5)
% Align dynamics to a base line 2 units above staff
\override DynamicLineSpanner #'staff-padding = #2
a4\f b\mf c\mp b\p
```



self-alignment-X property

The following example shows how this can resolve the collision of a string fingering object with a note's stem by aligning the right edge with the reference point of the parent note:
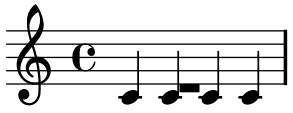
```
\voiceOne
<a\2>
\once \override StringNumber #'self-alignment-X = #RIGHT
<a\2>
```



staff-position property

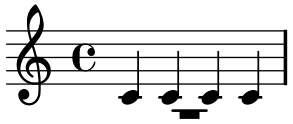
Multimeasure rests in one voice can collide with notes in another. Since these rests are typeset centered between the bar lines, it would require significant effort for LilyPond to figure out which other notes might collide with it, since all the current collision handling between notes and between notes and rests is done only for notes and rests that occur at the same time. Here's an example of a collision of this type:

```
<< { c4 c c c } \\ { R1 } >>
```



The best solution here is to move the multimeasure rest down, since the rest is in voice two. The default in `\voiceTwo` (i.e. in the second voice of a `<<{...} \\ {...}>>` construct) is that `staff-position` is set to -4 for MultiMeasureRest, so we need to move it, say, four half-staff spaces down to `-8`.

```
<<
  { c4 c c c }
  \\
  \override MultiMeasureRest #'staff-position = #-8
  { R1 }
>>
```
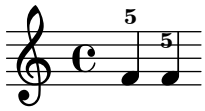


This is better than using, for example, `extra-offset`, because the ledger line above the rest is inserted automatically.

extra-offset property

The `extra-offset` property provides complete control over the positioning of an object both horizontally and vertically.

In the following example, the second fingering is moved a little to the left, and 1.8 staff space downwards:

```
\stemUp
f4-5
\once \override Fingering #'extra-offset = #'(-0.3 . -1.8)
f4-5
```



positions property
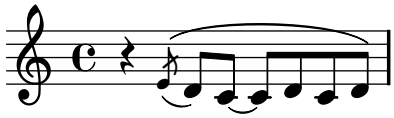
The `positions` property allows the position and slope of tuplets, slurs, phrasing slurs and beams to be controlled manually. Here's an example which has an ugly phrasing slur due to its trying to avoid the slur on the acciaccatura.

```
r4 \acciaccatura e8\( d8 c~ c d c d\)
```
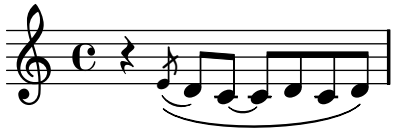


We could simply move the phrasing slur above the notes, and this would be the preferred solution:

```
r4
\phrasingSlurUp
\acciaccatura e8\( d8 c~ c d c d\)
```
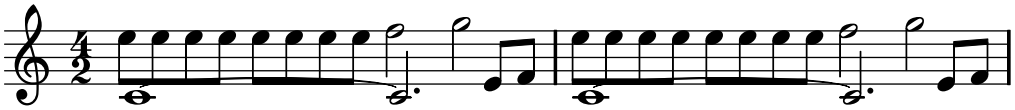


But if there were some reason why this could not be done the other alternative would be to move the left end of the phrasing slur down a little using the `positions` property. This also resolves the rather nasty shape.

```
r4
\once \override PhrasingSlur #'positions = #'(-4 . -3)
\acciaccatura e8\( d8 c~ c d c d\)
```
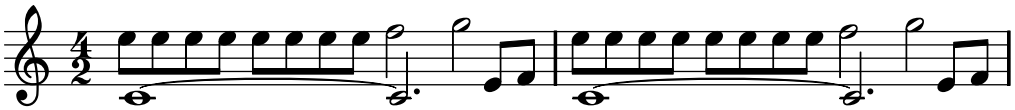


Here's a further example. We see that the beams collide with the ties:

```
{
  \time 4/2
  <<
    { c'1 ~ c'2. e'8 f' }
    \\
    { e''8 e'' e'' e''  e'' e'' e'' e''  f''2 g'' }
  >>
  <<
    { c'1 ~ c'2. e'8 f' }
    \\
    { e''8 e'' e'' e''  e'' e'' e'' e''  f''2 g'' }
  >>
}
```



This can be resolved by manually moving both ends of the beam up from their position at 1.81 staff-spaces below the center line to, say, 1:

```
{
  \time 4/2
  <<
    { c'1 ~ c'2. e'8 f' }
    \\
    {
      \override Beam #'positions = #'(-1 . -1)
      e''8 e'' e'' e''  e'' e'' e'' e''  f''2 g''
    }
  >>
  <<
    { c'1 ~ c'2. e'8 f' }
    \\
    { e''8 e'' e'' e''  e'' e'' e'' e''  f''2 g'' }
  >>
}
```
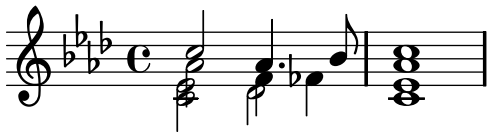


Note that the override continues to apply in the first voice of the second measure of eighth notes, but not to any of the beams in the second voice.

force-hshift property

We can now see how to apply the final corrections to the Chopin example introduced at the end of ⟨undefined⟩ [I'm hearing Voices], pagina ⟨undefined⟩, which was left looking like this:

```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 }
    \\
```
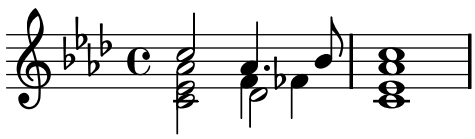
```
    { aes2 f4 fes }
    \\
    {
      \voiceFour
      <ees c>2 des
    }
  >> |
  <c ees aes c>1 |
}
```



The lower two notes of the first chord (i.e, those in the third voice) should not be shifted away from the note column of the higher two notes. To correct this we set `force-hshift`, which is a property of `NoteColumn`, of these notes to zero. The lower note of the second chord is best placed just to the right of the higher notes. We achieve this by setting `force-hshift` of this note to 0.5, ie half a note head's width to the right of the note column of the higher notes.

Here's the final result:

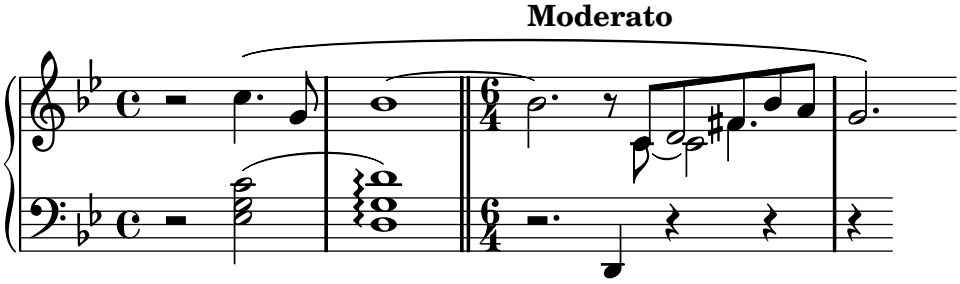
```
\new Staff \relative c'' {
  \key aes \major
  <<
    { c2 aes4. bes8 }
    \\
    { aes2 f4 fes }
    \\
    {
      \voiceFour
      \once \override NoteColumn #'force-hshift = #0
      <ees c>2
      \once \override NoteColumn #'force-hshift = #0.5
      des2
    }
  >> |
  <c ees aes c>1 |
}
```



4.5.3 Real music example

We end this section on Tweaks by showing the steps to be taken to deal with a tricky example which needs several tweaks to produce the desired output. The example has been deliberately chosen to illustrate the use of the Notation Reference to resolve unusual problems with notation. It is not representative of the more usual engraving process, so please do not let these difficulties put you off! Fortunately, difficulties like these are not very common!

The example is from Chopin's Première Ballade, Op. 23, bars 6 to 9, the transition from the opening Lento to Moderato. Here, first, is what we want the output to look like, but to avoid over-complicating the example too much we have left out the dynamics, fingering and pedalling.



We note first that the right hand part in the third bar requires four voices. These are the five beamed eighth notes, the tied C, the half-note D which is merged with the eighth note D, and the dotted quarter note F-sharp, which is also merged with the eighth note at the same pitch. Everything else is in a single voice, so the easiest way is to introduce these extra three voices temporarily at the time they are needed. If you have forgotten how to do this, look at ⟨undefined⟩ [I'm hearing Voices], pagina ⟨undefined⟩ and ⟨undefined⟩ [Explicitly instantiating voices], pagina ⟨undefined⟩. Here we choose to use explicitly instantiated voices for the polyphonic passage, as LilyPond is better able to avoid collisions if all voices are instantiated explicitly in this way.

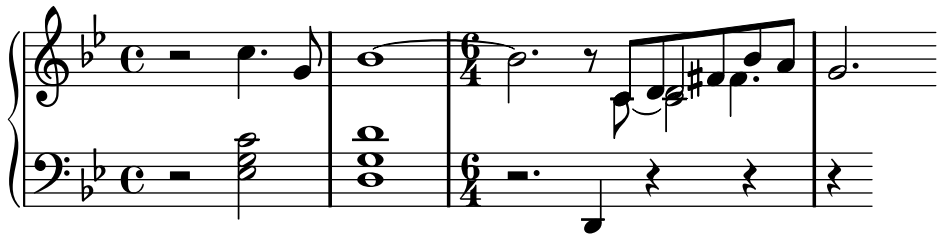
So let us begin by entering the notes as two variables, setting up the staff structure in a score block, and seeing what LilyPond produces by default:

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4. g8 |
    bes1~ |
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    g2.  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2 |
```

```
  <d g, d>1 |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



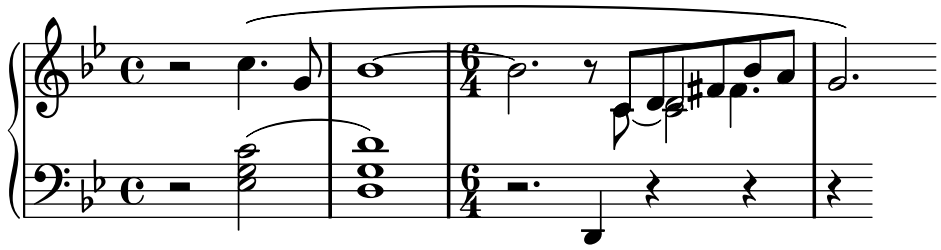
All the notes are right, but the appearance is far from satisfactory. The tie collides with the change in time signature, some notes are not merged together, and several notation elements are missing. Let's first deal with the easier things. We can easily add the left hand slur and the right hand phrasing slur, since these were all covered in the Tutorial. Doing this gives:

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    bes1~ |
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
```

```
    >> |
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1) |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



The first bar is now correct. The second bar contains an arpeggio and is terminated by a double bar line. How do we do these, as they have not been mentioned in this Learning Manual? This is where we need to turn to the Notation Reference. Looking up 'arpeggio' and 'bar line' in the index quickly shows us that an arpeggio is produced by appending `\arpeggio` to a chord, and a double bar line is produced by the `\bar "||"` command. That's easily done. We next need to correct the collision of the tie with the time signature. This is best done by moving the tie upwards. Moving objects was covered earlier in Sezione 4.5.1 [Moving objects], pagina 116, which says that objects positioned relative to the staff can be moved vertically by overriding their `staff-position` property, which is specified in half staff spaces relative to the center line of the staff. So the following override placed just before the first tied note would move the tie up to 3.5 half staff spaces above the center line:

```
\once \override Tie #'staff-position = #3.5
```
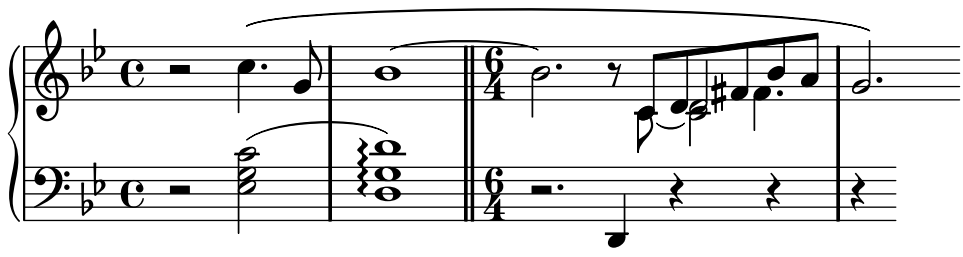
This completes bar two, giving:

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
```

```
    bes1~ |
    \bar "||"
    \time 6/4
    bes2. r8
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~ c2
      }
      \new Voice {
        \voiceThree
        s8 d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



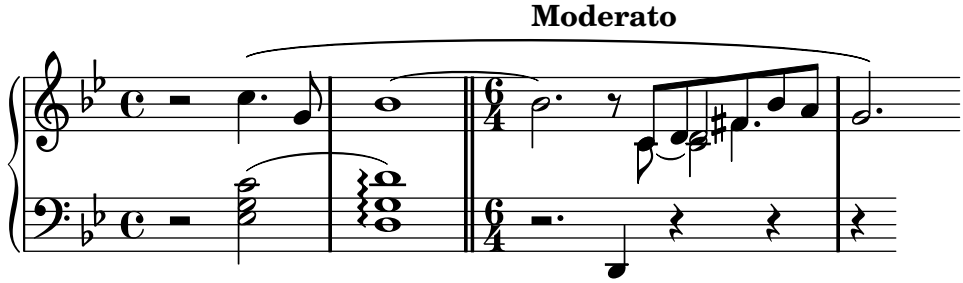
On to bar three and the start of the Moderato section. The tutorial showed how to add bold text with the `\markup` command, so adding "Moderato" in bold is easy. But how do we merge notes in different voices together? This is where we need to turn again to the Notation Reference for help. A search for "merge" in the Notation Reference index quickly leads us to the commands for merging differently headed and differently dotted notes in Sezione "Collision resolution" in *Guida alla Notazione*. In our example we need to merge both types of note for the duration of the polyphonic section in bar 3, so using the information we find in the Notation Reference we add

```
\mergeDifferentlyHeadedOn
\mergeDifferentlyDottedOn
```

to the start of that section and

```
\mergeDifferentlyHeadedOff
\mergeDifferentlyDottedOff
```

to the end, giving:



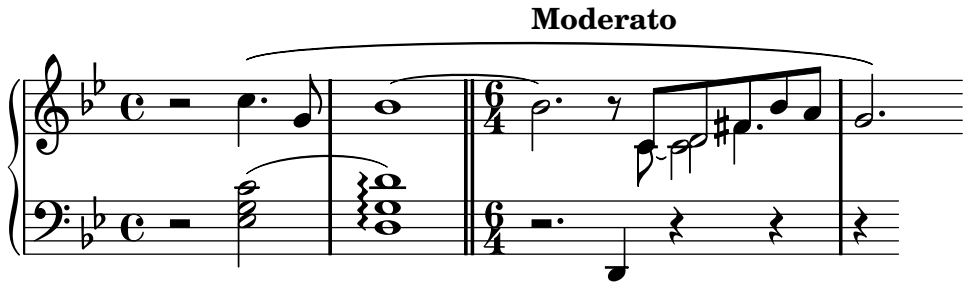
These overrides have merged the two F-sharp notes, but not the two on D. Why not? The answer is there in the same section in the Notation Reference – notes being merged must have stems in opposite directions and two notes cannot be merged successfully if there is a third note in the same note column. Here the two D's both have upward stems and there is a third note – the C. We know how to change the stem direction using `\stemDown`, and the Notation Reference also says how to move the C – apply a shift using one of the `\shift` commands. But which one? The C is in voice two which has shift off, and the two D's are in voices one and three, which have shift off and shift on, respectively. So we have to shift the C a further level still using `\shiftOnn` to avoid it interfering with the two D's. Applying these changes gives:

```
rhMusic = \relative c'' {
  \new Voice {
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2.^\markup { \bold "Moderato" } r8
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        % Move the c2 out of the main note column so the merge will work
        c,8~ \shiftOnn c2
      }
```

```
      \new Voice {
        \voiceThree
        % Stem on the d2 must be down to permit merging
        s8 \stemDown d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    \mergeDifferentlyHeadedOff
    \mergeDifferentlyDottedOff
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```



Nearly there. Only two problems remain: The downward stem on the merged D should not be there, and the C would be better positioned to the right of the D's. We know how to do both of these from the earlier tweaks: we make the stem transparent, and move the C with the `force-hshift` property. Here's the final result:

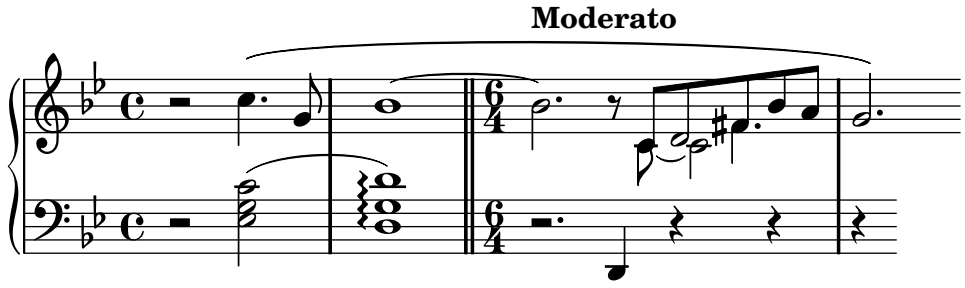
```
rhMusic = \relative c'' {
  \new Voice {
```

```
    r2 c4.\( g8 |
    \once \override Tie #'staff-position = #3.5
    bes1~ |
    \bar "||"
    \time 6/4
    bes2.^\markup { \bold "Moderato" } r8
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    % Start polyphonic section of four voices
    <<
      { c,8 d fis bes a }  % continuation of main voice
      \new Voice {
        \voiceTwo
        c,8~
        % Reposition the c2 to the right of the merged note
        \once \override NoteColumn #'force-hshift = #1.0
        % Move the c2 out of the main note column so the merge will work
        \shiftOnn
        c2
      }
      \new Voice {
        \voiceThree
        s8
        % Stem on the d2 must be down to permit merging
        \stemDown
        % Stem on the d2 should be invisible
        \once \override Stem #'transparent = ##t
        d2
      }
      \new Voice {
        \voiceFour
        s4 fis4.
      }
    >> |
    \mergeDifferentlyHeadedOff
    \mergeDifferentlyDottedOff
    g2.\)  % continuation of main voice
  }
}

lhMusic = \relative c' {
  r2 <c g ees>2( |
  <d g, d>1)\arpeggio |
  r2. d,,4 r4 r |
  r4
}

\score {
  \new PianoStaff <<
    \new Staff = "RH"  <<
      \key g \minor
      \rhMusic
```

```
    >>
    \new Staff = "LH" <<
      \key g \minor
      \clef "bass"
      \lhMusic
    >>
  >>
}
```
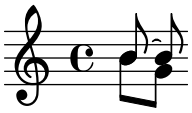


4.6 Further tweaking

4.6.1 Other uses for tweaks
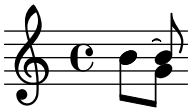
Tying notes across voices

The following example demonstrates how to connect notes in different voices using ties. Normally, only two notes in the same voice can be connected with ties. By using two voices, with the tied notes in one of them



and blanking the first up-stem in that voice, the tie appears to cross voices:
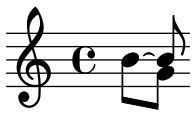
```
<<
  {
    \once \override Stem #'transparent = ##t
    b8~ b\noBeam
  }
\\
  { b8[ g] }
>>
```



To make sure that the just-blanked stem doesn't squeeze the tie too much, we can lengthen the stem by setting the `length` to `8`,

```
<<
  {
    \once \override Stem #'transparent = ##t
    \once \override Stem #'length = #8
```

```
    b8~ b\noBeam
  }
\\
  { b8[ g] }
>>
```

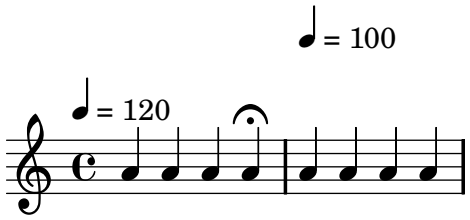


Simulating a fermata in MIDI

For outside-staff objects it is usually better to override the object's `stencil` property rather than its `transparent` property when you wish to remove it from the printed output. Setting the `stencil` property of an object to `#f` will remove that object entirely from the printed output. This means it has no effect on the placement of other objects placed relative to it.
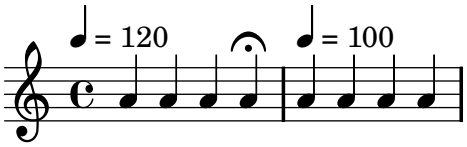
For example, if we wished to change the metronome setting in order to simulate a fermata in the MIDI output we would not want the metronome markings to appear in the printed output, and we would not want it to influence the spacing between the two systems or the positions of adjacent annotations on the staff. So setting its `stencil` property to `#f` would be the best way. We show here the effect of the two methods:

```
\score {
  \relative c'' {
    % Visible tempo marking
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'transparent = ##t
    % Invisible tempo marking to lengthen fermata in MIDI
    \tempo 4=80
    a4\fermata |
    % New tempo for next section
    \tempo 4=100
    a4 a a a |
  }
  \layout { }
  \midi { }
}
```




```
\score {
  \relative c'' {
    % Visible tempo marking
    \tempo 4=120
    a4 a a
    \once \override Score.MetronomeMark #'stencil = ##f
    % Invisible tempo marking to lengthen fermata in MIDI
```

```
    \tempo 4=80
    a4\fermata |
    % New tempo for next section
    \tempo 4=100
    a4 a a a |
  }
  \layout { }
  \midi { }
}
```



Both methods remove the metronome mark which lengthens the fermata from the printed output, and both affect the MIDI timing as required, but the transparent metronome mark in the first line forces the following tempo indication too high while the second (with the stencil removed) does not.

Vedi anche

Music Glossary: Sezione “system” in *Glossario Musicale*.

4.6.2 Using variables for tweaks

Override commands are often long and tedious to type, and they have to be absolutely correct. If the same overrides are to be used many times it may be worth defining variables to hold them.

Suppose we wish to emphasize certain words in lyrics by printing them in bold italics. The `\italic` and `\bold` commands only work within lyrics if they are embedded, together with the word or words to be modified, within a `\markup` block, which makes them tedious to enter. The need to embed the words themselves prevents their use in simple variables. As an alternative can we use `\override` and `\revert` commands?

```
\override Lyrics . LyricText #'font-shape = #'italic
\override Lyrics . LyricText #'font-series = #'bold

\revert Lyrics . LyricText #'font-shape
\revert Lyrics . LyricText #'font-series
```

These would also be extremely tedious to enter if there were many words requiring emphasis. But we *can* define these as two variables and use those to bracket the words to be emphasized. Another advantage of using variables for these overrides is that the spaces around the dot are not necessary, since they are not being interpreted in `\lyricmode` directly. Here’s an example of this, although in practice we would choose shorter names for the variables to make them quicker to type:

```
emphasize = {
  \override Lyrics.LyricText #'font-shape = #'italic
  \override Lyrics.LyricText #'font-series = #'bold
}

normal = {
  \revert Lyrics.LyricText #'font-shape
  \revert Lyrics.LyricText #'font-series
}
```

```
global = { \key c \major \time 4/4 \partial 4 }

SopranoMusic = \relative c' { c4 | e4. e8 g4 g    | a4   a   g  }
AltoMusic    = \relative c' { c4 | c4. c8 e4 e    | f4   f   e  }
TenorMusic   = \relative c  { e4 | g4. g8 c4.  b8 | a8 b c d e4 }
BassMusic    = \relative c  { c4 | c4. c8 c4 c    | f8 g a b c4 }

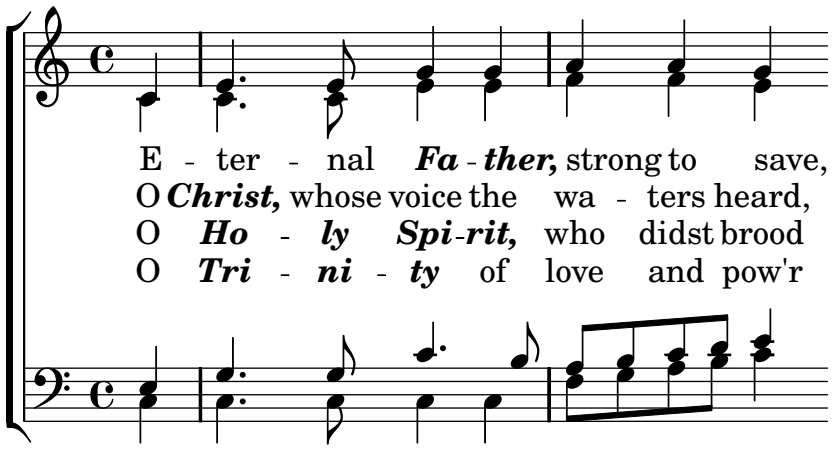
VerseOne = \lyrics {
  E -- | ter -- nal \emphasize Fa -- ther, | \normal strong to save,
}

VerseTwo = \lyricmode {
  O | \emphasize Christ, \normal whose voice the | wa -- ters heard,
}

VerseThree = \lyricmode {
  O | \emphasize Ho -- ly Spi -- rit, | \normal who didst brood
}

VerseFour = \lyricmode {
  O | \emphasize Tri -- ni -- ty \normal of | love and pow'r
}

\score {
  \new ChoirStaff <<
    \new Staff <<
      \clef "treble"
      \new Voice = "Soprano"  { \voiceOne \global \SopranoMusic }
      \new Voice = "Alto" { \voiceTwo \AltoMusic }
      \new Lyrics \lyricsto "Soprano" { \VerseOne }
      \new Lyrics \lyricsto "Soprano" { \VerseTwo }
      \new Lyrics \lyricsto "Soprano" { \VerseThree }
      \new Lyrics \lyricsto "Soprano" { \VerseFour }
    >>
    \new Staff <<
      \clef "bass"
      \new Voice = "Tenor" { \voiceOne \TenorMusic }
      \new Voice = "Bass"  { \voiceTwo \BassMusic }
    >>
  >>
}
```



4.6.3 Style sheets
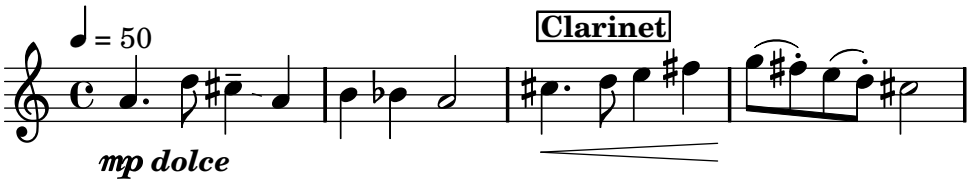
The output that LilyPond produces can be heavily modified; see Capitolo 4 [Tweaking output], pagina 87, for details. But what if you have many input files that you want to apply your tweaks to? Or what if you simply want to separate your tweaks from the actual music? This is quite easy to do.

Let's look at an example. Don't worry if you don't understand the parts with all the `#()`. This is explained in Sezione 4.6.5 [Advanced tweaks with Scheme], pagina 140.

```
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a |
  b4 bes a2 |
  \inst "Clarinet"
  cis4.\< d8 e4 fis |
  g8(\! fis)-. e( d)-. cis2 |
}
```



There are some problems with overlapping output; we'll fix those using the techniques in Sezione 4.5.1 [Moving objects], pagina 116. But let's also do something about the `mpdolce` and

`inst` definitions. They produce the output we desire, but we might want to use them in another piece. We could simply copy-and-paste them at the top of every file, but that's an annoyance. It also leaves those definitions in our input files, and I personally find all the `#()` somewhat ugly. Let's hide them in another file:

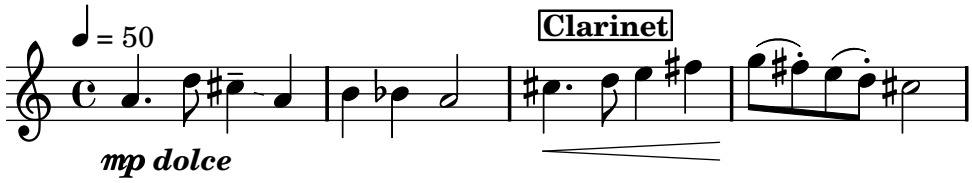
```
%%% save this to a file called "definitions.ily"
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))
```

We will refer to this file using the `\include` command near the top of the music file. (The extension '`.ily`' is used to distinguish this included file, which is not meant to be compiled on its own, from the main file.) Now let's modify our music (let's save this file as '`music.ly`').

```
\include "definitions.ily"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a |
  b4 bes a2 |
  \inst "Clarinet"
  cis4.\< d8 e4 fis |
  g8(\! fis)-. e( d)-. cis2 |
}
```



That looks better, but let's make a few changes. The glissando is hard to see, so let's make it thicker and closer to the note heads. Let's put the metronome marking above the clef, instead of over the first note. And finally, my composition professor hates 'C' time signatures, so we'd better make that '4/4' instead.

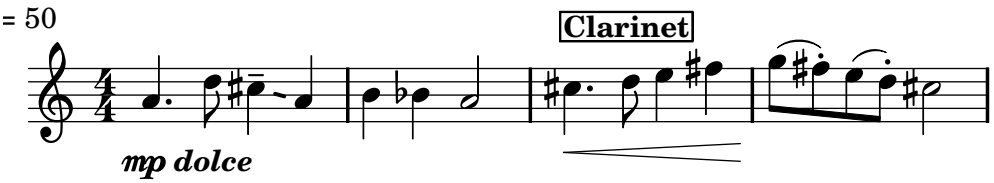
Don't change '`music.ly`', though. Replace our '`definitions.ily`' with this:

```
%%%  definitions.ily
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
```

```
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
   (make-music
    'TextScriptEvent
    'direction UP
    'text (markup #:bold (#:box string))))

\layout{
  \context {
    \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context {
    \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context {
    \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



That looks nicer! But now suppose that I want to publish this piece. My composition professor doesn't like 'C' time signatures, but I'm somewhat fond of them. Let's copy the current '`definitions.ily`' to '`web-publish.ily`' and modify that. Since this music is aimed at producing a pdf which will be displayed on the screen, we'll also increase the overall size of the output.
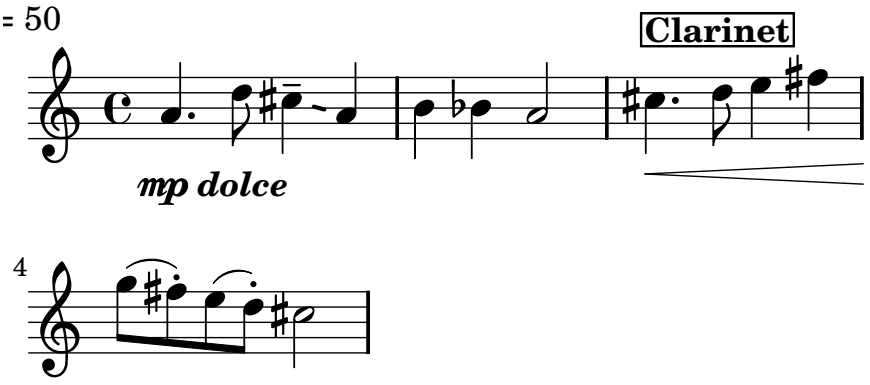
```
%%%  definitions.ily
mpdolce =
#(make-dynamic-script
  (markup #:hspace 0
          #:translate '(5 . 0)
          #:line (#:dynamic "mp"
                  #:text #:italic "dolce")))

inst =
#(define-music-function
     (parser location string)
     (string?)
```

```
    (make-music
     'TextScriptEvent
     'direction UP
     'text (markup #:bold (#:box string))))

#(set-global-staff-size 23)

\layout{
  \context {
    \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context {
    \Staff
  }
  \context {
    \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



Now in our music, I simply replace `\include "definitions.ily"` with `\include "web-publish.ily"`. Of course, we could make this even more convenient. We could make a '`definitions.ily`' file which contains only the definitions of `mpdolce` and `inst`, a '`web-publish.ily`' file which contains only the `\layout` section listed above, and a '`university.ily`' file which contains only the tweaks to produce the output that my professor prefers. The top of '`music.ly`' would then look like this:

```
\include "definitions.ily"

%%%  Only uncomment one of these two lines!
\include "web-publish.ily"
%\include "university.ily"
```

This approach can be useful even if you are only producing one set of parts. I use half a dozen different 'style sheet' files for my projects. I begin every music file with `\include "../global.ily"`, which contains

```
%%%   global.ily
\version "2.14.2"
```

```
#(ly:set-option 'point-and-click #f)

\include "../init/init-defs.ly"
\include "../init/init-layout.ly"
\include "../init/init-headers.ly"
\include "../init/init-paper.ly"
```

4.6.4 Other sources of information

The Internals Reference documentation contains a lot of information about LilyPond, but even more information can be gathered by looking at the internal LilyPond files. To explore these, you must first find the directory appropriate to your system. The location of this directory depends (a) on whether you obtained LilyPond by downloading a precompiled binary from lilypond.org or whether you installed it from a package manager (i.e. distributed with Linux, or installed under fink or cygwin) or compiled it from source, and (b) on which operating system it is being used:

Downloaded from lilypond.org

- Linux

  Navigate to '*INSTALLDIR*`/lilypond/usr/share/lilypond/current/`'
- MacOS X

  Navigate to '*INSTALLDIR*`/LilyPond.app/Contents/Resources/share/lilypond/current/`' by either `cd`-ing into this directory from the Terminal, or control-clicking on the LilyPond application and selecting 'Show Package Contents'.
- Windows

  Using Windows Explorer, navigate to '*INSTALLDIR*`/LilyPond/usr/share/lilypond/current/`'

Installed from a package manager or compiled from source

Navigate to '*PREFIX*`/share/lilypond/`*X.Y.Z*`/`', where *PREFIX* is set by your package manager or `configure` script, and *X.Y.Z* is the LilyPond version number.

Within this directory the two interesting subdirectories are

- '`ly/`' - contains files in LilyPond format
- '`scm/`' - contains files in Scheme format

Let's begin by looking at some files in '`ly/`'. Open '`ly/property-init.ly`' in a text editor. The one you normally use for `.ly` files will be fine. This file contains the definitions of all the standard LilyPond predefined commands, such as `\stemUp` and `\slurDotted`. You will see that these are nothing more than definitions of variables containing one or a group of `\override` commands. For example, `/tieDotted` is defined to be:

```
tieDotted = {
  \override Tie #'dash-period = #0.75
  \override Tie #'dash-fraction = #0.1
}
```

If you do not like the default values these predefined commands can be redefined easily, just like any other variable, at the head of your input file.

The following are the most useful files to be found in '`ly/`':

Filename	**Contents**

‘ly/engraver-init.ly’	Definitions of engraver Contexts
‘ly/paper-defaults-init.ly’	Specifications of paper-related defaults
‘ly/performer-init.ly’	Definitions of performer Contexts
‘ly/property-init.ly’	Definitions of all common predefined commands
‘ly/spanner-init.ly’	Definitions of spanner-related predefined commands

Other settings (such as the definitions of markup commands) are stored as ‘.scm’ (Scheme) files. The Scheme programming language is used to provide a programmable interface into LilyPond internal operation. Further explanation of these files is currently outside the scope of this manual, as a knowledge of the Scheme language is required. Users should be warned that a substantial amount of technical knowledge or time is required to understand Scheme and these files (see Sezione “Scheme tutorial” in *Estendere*).

If you have this knowledge, the Scheme files which may be of interest are:

Filename	**Contents**
‘scm/auto-beam.scm’	Sub-beaming defaults
‘scm/define-grobs.scm’	Default settings for grob properties
‘scm/define-markup-commands.scm’	Specify all markup commands
‘scm/midi.scm’	Default settings for MIDI output
‘scm/output-lib.scm’	Settings that affect appearance of frets, colors, accidentals, bar lines, etc
‘scm/parser-clef.scm’	Definitions of supported clefs
‘scm/script.scm’	Default settings for articulations
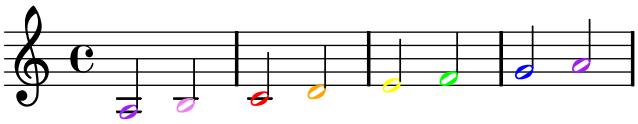
4.6.5 Advanced tweaks with Scheme

Although many things are possible with the \override and \tweak commands, an even more powerful way of modifying the action of LilyPond is available through a programmable interface to the LilyPond internal operation. Code written in the Scheme programming language can be incorporated directly in the internal operation of LilyPond. Of course, at least a basic knowledge of programming in Scheme is required to do this, and an introduction is provided in the Sezione “Scheme tutorial” in *Estendere*.

As an illustration of one of the many possibilities, instead of setting a property to a constant it can be set to a Scheme procedure which is then called whenever that property is accessed by LilyPond. The property can then be set dynamically to a value determined by the procedure at the time it is called. In this example we color the note head in accordance with its position on the staff.

```
#(define (color-notehead grob)
   "Color the notehead according to its position on the staff."
   (let ((mod-position (modulo (ly:grob-property grob 'staff-position)
                               7)))
     (case mod-position
       ;;   Return rainbow colors
       ((1) (x11-color 'red    ))  ; for C
       ((2) (x11-color 'orange ))  ; for D
       ((3) (x11-color 'yellow ))  ; for E
       ((4) (x11-color 'green  ))  ; for F
       ((5) (x11-color 'blue   ))  ; for G
       ((6) (x11-color 'purple ))  ; for A
       ((0) (x11-color 'violet ))  ; for B
       )))
```

```
\relative c' {
  % Arrange to obtain color from color-notehead procedure
  \override NoteHead #'color = #color-notehead
  a2 b | c2 d | e2 f | g2 a |
}
```



Further examples showing the use of these programmable interfaces can be found in Sezione "Callback functions" in *Estendere*.

Appendice A Modelli

Questa sezione del manuale contiene dei modelli con la struttura del file LilyPond già preimpostata. Non ti resta che aggiungere le note, eseguire LilyPond e goderti dei belli spartiti!

A.1 Rigo singolo
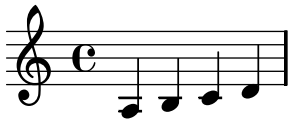
A.1.1 Solo note

Questo modello molto semplice mette a disposizione un rigo con delle note ed è quindi adatto per uno strumento non accompagnato o per un frammento melodico. Copialo e incollalo in un file, aggiungi le note e hai finito!

```
\version "2.14.2"
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

\score {
  \new Staff \melody
  \layout { }
  \midi { }
}
```



A.1.2 Note e testo

Questo piccolo modello presenta una semplice linea melodica con un testo. Copialo e incollalo, aggiungi le note e le parole. Questo esempio disabilita la disposizione automatica delle travature, come è consuetudine per le parti vocali. Per usare la disposizione automatica delle travature, cambia o commenta la relativa linea di codice.

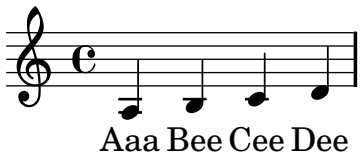
```
\version "2.14.2"
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
```

```
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melody
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



A.1.3 Note e accordi
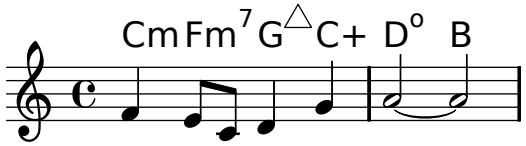
Vuoi preparare uno spartito semplificato (lead sheet) con melodia e accordi? La tua ricerca è finita!

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  f4 e8[ c] d4 g
  a2 ~ a
}

harmonies = \chordmode {
  c4:m f:min7 g:maj c:aug
  d2:dim b:sus
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Staff \melody
  >>
  \layout{ }
  \midi { }
}
```



A.1.4 Note, testo e accordi.

Ecco il modello di un comune spartito semplificato (lead sheet): include linea melodica, testo vocale, sigle degli accordi e relativi diagrammi per chitarra.

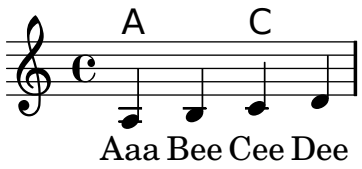
```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



A.2 Modelli per pianoforte

A.2.1 Solo pianoforte

Ecco un comune doppio pentagramma per pianoforte con un po' di note.
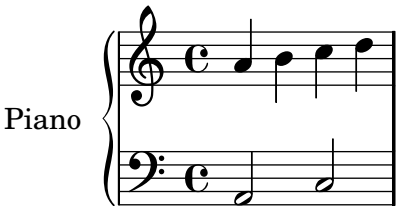
```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}
```

```
lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  \new PianoStaff <<
    \set PianoStaff.instrumentName = #"Piano  "
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
  \layout { }
  \midi { }
}
```



A.2.2 Pianoforte e melodia con testo

Ecco un tipico formato per canzoni: un rigo con linea melodica e testo, e sotto l'accompagnamento per pianoforte.

```
melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

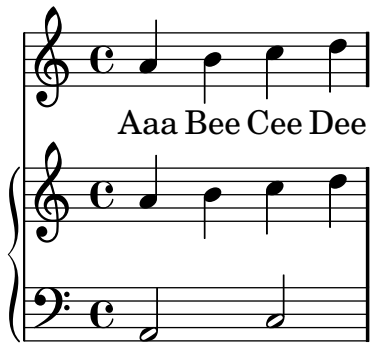
  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
```

```
  \time 4/4

  a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}
```



A.2.3 Pianoforte con testo al centro

Invece di destinare un rigo a parte alla linea melodica e al suo testo, è possibile collocare il testo al centro di un doppio pentagramma per pianoforte.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

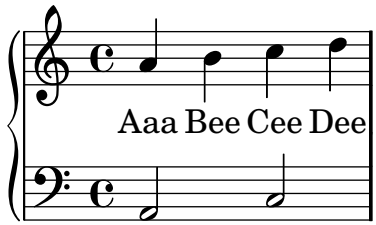
lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

text = \lyricmode {
```

```
  Aaa Bee Cee Dee
}

\score {
  \new GrandStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout {
    \context {
      \GrandStaff
      \accepts "Lyrics"
    }
    \context {
      \Lyrics
      \consists "Bar_engraver"
    }
  }
  \midi { }
}
```



A.2.4 Pianoforte con dinamiche al centro

Molti spartiti per pianoforte hanno le dinamiche poste al centro dei due righi. Il contesto `Dynamics`, se collocato tra i righi, posiziona automaticamente le dinamiche in modo corretto.

```
global = {
  \key c \major
  \time 4/4
}

upper = \relative c'' {
  \clef treble
  a4 b c d
}

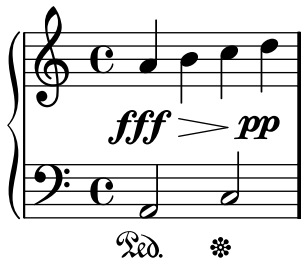
lower = \relative c {
  \clef bass
  a2 c
}

dynamics = {
  s2\fff\> s4 s\!\pp
}
```

```
pedal = {
  s2\sustainOn s\sustainOff
}

\score {
  \new PianoStaff = "PianoStaff_pf" <<
    \new Staff = "Staff_pfUpper" << \global \upper >>
    \new Dynamics = "Dynamics_pf" \dynamics
    \new Staff = "Staff_pfLower" << \global \lower >>
    \new Dynamics = "pedal" \pedal
  >>
  \layout { }
}

\score {
  \new PianoStaff = "PianoStaff_pf" <<
    \new Staff = "Staff_pfUpper" << \global \upper \dynamics \pedal >>
    \new Staff = "Staff_pfLower" << \global \lower \dynamics \pedal >>
  >>
  \midi { }
}
```



A.3 Quartetto d'archi

A.3.1 Quartetto d'archi

Questo modello presenta un semplice quartetto d'archi. Impiega anche una sezione `\global` per definire il tempo e l'armatura di chiave.

```
global= {
  \time 4/4
  \key c \major
}

violinOne = \new Voice \relative c'' {
  \set Staff.instrumentName = #"Violin 1 "

  c2 d
  e1

  \bar "|."
}

violinTwo = \new Voice \relative c'' {
```

```
  \set Staff.instrumentName = #"Violin 2 "

  g2 f
  e1

  \bar "|."
}

viola = \new Voice \relative c' {
  \set Staff.instrumentName = #"Viola "
  \clef alto

  e2 d
  c1

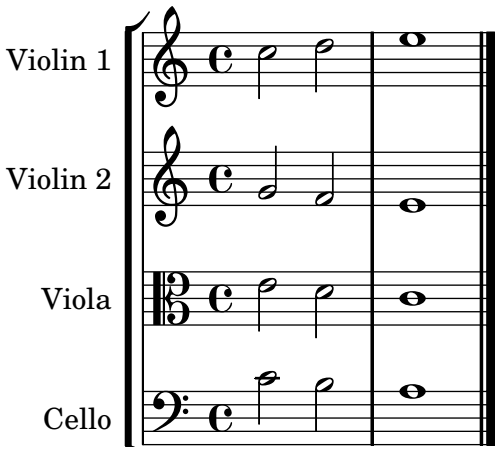
  \bar "|."
}

cello = \new Voice \relative c' {
  \set Staff.instrumentName = #"Cello "
  \clef bass

  c2 b
  a1

  \bar "|."
}

\score {
  \new StaffGroup <<
    \new Staff << \global \violinOne >>
    \new Staff << \global \violinTwo >>
    \new Staff << \global \viola >>
    \new Staff << \global \cello >>
  >>
  \layout { }
  \midi { }
}
```



A.3.2 Parti di un quartetto d'archi

The "String quartet template" snippet produces a nice string quartet, but what if you needed to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

You need to split this template into separate files; the filenames are contained in comments at the beginning of each file. `piece.ly` contains all the music definitions. The other files – `score.ly`, `vn1.ly`, `vn2.ly`, `vla.ly`, and `vlc.ly` – produce the appropriate part.

Do not forget to remove specified comments when using separate files!

```
%%%%% piece.ly
%%%%% (This is the global definitions file)

global= {
  \time 4/4
  \key c \major
}

Violinone = \new Voice { \relative c''{
  \set Staff.instrumentName = #"Violin 1 "

  c2 d e1

\bar "|." }}   %*********************************
Violintwo = \new Voice { \relative c''{
  \set Staff.instrumentName = #"Violin 2 "

  g2 f e1

\bar "|." }}   %*********************************
Viola = \new Voice { \relative c' {
  \set Staff.instrumentName = #"Viola "
  \clef alto

  e2 d c1

\bar "|." }}   %*********************************
Cello = \new Voice { \relative c' {
  \set Staff.instrumentName = #"Cello "
  \clef bass

  c2 b a1
```

```
\bar "|."}}   %**********************************

music = {
  <<
    \tag #'score \tag #'vn1 \new Staff { << \global \Violinone >> }
    \tag #'score \tag #'vn2 \new Staff { << \global \Violintwo>> }
    \tag #'score \tag #'vla \new Staff { << \global \Viola>> }
    \tag #'score \tag #'vlc \new Staff { << \global \Cello>> }
  >>
}

%%% These are the other files you need to save on your computer

%%%%% score.ly
%%%%% (This is the main file)


%\include "piece.ly"             %%% uncomment this line when using a separate file
#(set-global-staff-size 14)
\score {
  \new StaffGroup \keepWithTag #'score \music
  \layout { }
  \midi { }
}


%{ Uncomment this block when using separate files

%%%%% vn1.ly
%%%%% (This is the Violin 1 part file)

\include "piece.ly"
\score {
  \keepWithTag #'vn1 \music
  \layout { }
}


%%%%% vn2.ly
%%%%% (This is the Violin 2 part file)

\include "piece.ly"
\score {
  \keepWithTag #'vn2 \music
  \layout { }
}

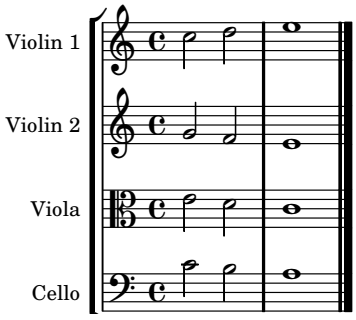

%%%%% vla.ly
%%%%% (This is the Viola part file)
```

```
\include "piece.ly"
\score {
  \keepWithTag #'vla \music
  \layout { }
}


%%%%% vlc.ly
%%%%% (This is the Cello part file)

\include "piece.ly"
\score {
  \keepWithTag #'vlc \music
  \layout { }
}

%}
```



A.4 Gruppi vocali

A.4.1 Partitura vocale SATB

Ecco una tipica partitura corale a quattro parti, SATB. Se il complesso è più ampio, è spesso comodo scrivere gli elementi comuni in un'unica sezione, che verrà poi inclusa in tutte le parti. Ad esempio, l'indicazione di tempo e l'armatura di chiave sono quasi sempre le stesse per tutte le parti. Come nel modello dell'"Inno", le quattro voci sono ripartite in due soli righi.

```
\paper {
  top-system-spacing #'basic-distance = #10
  score-system-spacing #'basic-distance = #20
  system-system-spacing #'basic-distance = #20
  last-bottom-spacing #'basic-distance = #10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
```

```
sopWords = \lyricmode {
  hi hi hi hi
}
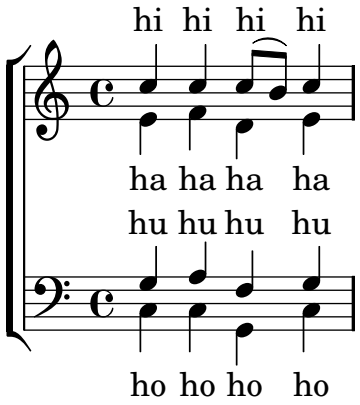
altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Lyrics = "sopranos" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \new Staff = "women" <<
      \new Voice = "sopranos" {
        \voiceOne
        << \global \sopMusic >>
      }
      \new Voice = "altos" {
        \voiceTwo
        << \global \altoMusic >>
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup #'staff-affinity = #DOWN
    }
    \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" \sopWords
    \context Lyrics = "altos" \lyricsto "altos" \altoWords
    \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
    \context Lyrics = "basses" \lyricsto "basses" \bassWords
  >>
}
```



A.4.2 Partitura vocale SATB e automatica riduzione per pianoforte

Questo modello aggiunge una riduzione automatica per pianoforte alla tipica partitura vocale SATB illustrata in "Modello per complesso vocale". Si dimostra così uno dei punti di forza di LilyPond – è possibile usare una definizione musicale più di una volta. Qualsiasi modifica venga fatta alle note delle voci (ad esempio, `tenorMusic`) verrà applicata anche alla riduzione per pianoforte.

```
\paper {
  top-system-spacing #'basic-distance = #10
  score-system-spacing #'basic-distance = #20
  system-system-spacing #'basic-distance = #20
  last-bottom-spacing #'basic-distance = #10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
```

```
  e4 f d e
}
altoWords =\lyricmode {
  ha ha ha ha
}

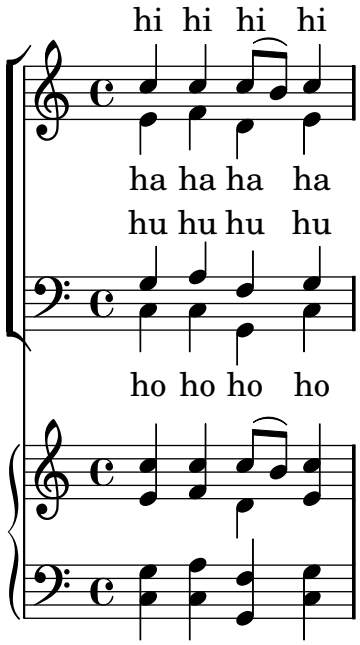
tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup #'staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"
      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup #'staff-affinity = #DOWN
      }

      \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" \sopWords
      \context Lyrics = "altos" \lyricsto "altos" \altoWords
      \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
      \context Lyrics = "basses" \lyricsto "basses" \bassWords
    >>
    \new PianoStaff <<
      \new Staff <<
        \set Staff.printPartCombineTexts = ##f
        \partcombine
```

```
        << \global \sopMusic >>
        << \global \altoMusic >>
      >>
      \new Staff <<
        \clef bass
        \set Staff.printPartCombineTexts = ##f
        \partcombine
        << \global \tenorMusic >>
        << \global \bassMusic >>
      >>
    >>
  >>
}
```



A.4.3 SATB con contesti allineati

Questo modello è fondamentalmente analogo al semplice modello "Complesso vocale", con l'unica differenza che qui tutti i versi del testo sono posizionati usando `alignAboveContext` e `alignBelowContext`.

```
global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
```
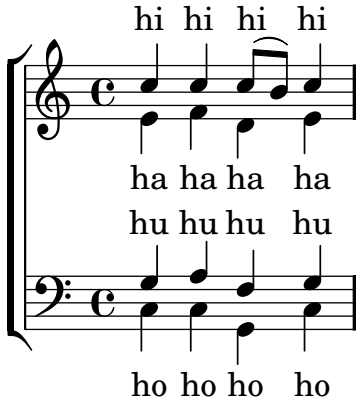
```
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Staff = "women" <<
      \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
      \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = #"women" } \lyricsto "sopranos" \sopWords
    \new Lyrics \with { alignBelowContext = #"women" } \lyricsto "altos" \altoWords
    % we could remove the line about this with the line below, since we want
    % the alto lyrics to be below the alto Voice anyway.
    % \new Lyrics \lyricsto "altos" \altoWords

    \new Staff = "men" <<
      \clef bass
      \new Voice = "tenors" { \voiceOne << \global \tenorMusic >> }
      \new Voice = "basses" { \voiceTwo << \global \bassMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = #"men" } \lyricsto "tenors" \tenorWords
    \new Lyrics \with { alignBelowContext = #"men" } \lyricsto "basses" \bassWords
    % again, we could replace the line above this with the line below.
    % \new Lyrics \lyricsto "basses" \bassWords
  >>
}
```
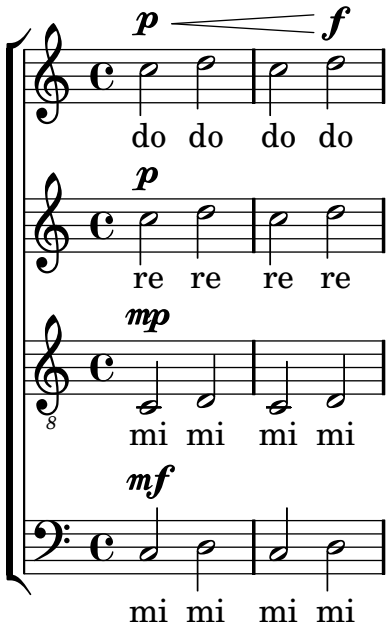


A.4.4 SATB su quattro righi

Modello per coro SATB (quattro righi)

```
global = {
  \key c \major
  \time 4/4
  \dynamicUp
}
sopranonotes = \relative c'' {
  c2 \p \< d c d \f
}
sopranowords = \lyricmode { do do do do }
altonotes = \relative c'' {
  c2\p d c d
}
altowords = \lyricmode { re re re re }
tenornotes = {
  \clef "G_8"
  c2\mp d c d
}
tenorwords = \lyricmode { mi mi mi mi }
bassnotes = {
  \clef bass
  c2\mf d c d
}
basswords = \lyricmode { mi mi mi mi }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "soprano" <<
        \global
        \sopranonotes
      >>
      \lyricsto "soprano" \new Lyrics \sopranowords
    >>
    \new Staff <<
      \new Voice = "alto" <<
        \global
        \altonotes
      >>
      \lyricsto "alto" \new Lyrics \altowords
```

```
    >>
    \new Staff <<
      \new Voice = "tenor" <<
        \global
        \tenornotes
      >>
      \lyricsto "tenor" \new Lyrics \tenorwords
    >>
    \new Staff <<
      \new Voice = "bass" <<
        \global
        \bassnotes
      >>
      \lyricsto "bass" \new Lyrics \basswords
    >>
  >>
}
```



A.4.5 Strofa sola e ritornello a due parti

Questo modello crea una partitura che inizia con una sezione solistica e prosegue in un ritornello a due voci. Illustra anche l'uso delle pause spaziatrici all'interno della variabile `\global` per definire i cambi di tempo (e altri elementi comuni a tutte le parti) nel corso di tutta la partitura.

```
global = {
  \key g \major

  % verse
  \time 3/4
  s2.*2
  \break

  % refrain
  \time 2/4
```

```
  s2*2
  \bar "|."
}

SoloNotes = \relative g' {
  \clef "treble"

  % verse
  g4 g g |
  b4 b b |

  % refrain
  R2*2 |
}

SoloLyrics = \lyricmode {
  One two three |
  four five six |
}

SopranoNotes = \relative c'' {
  \clef "treble"

  % verse
  R2.*2 |

  % refrain
  c4 c |
  g4 g |
}

SopranoLyrics = \lyricmode {
  la la |
  la la |
}

BassNotes = \relative c {
  \clef "bass"

  % verse
  R2.*2 |
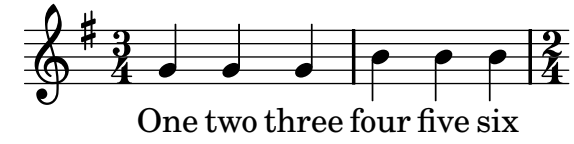
  % refrain
  c4 e |
  d4 d |
}

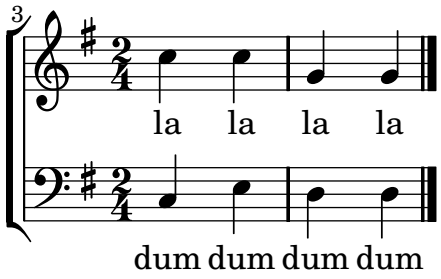
BassLyrics = \lyricmode {
  dum dum |
  dum dum |
}
```

```
\score {
  <<
    \new Voice = "SoloVoice" << \global \SoloNotes >>
    \new Lyrics \lyricsto "SoloVoice" \SoloLyrics

    \new ChoirStaff <<
      \new Voice = "SopranoVoice" << \global \SopranoNotes >>
      \new Lyrics \lyricsto "SopranoVoice" \SopranoLyrics

      \new Voice = "BassVoice" << \global \BassNotes >>
      \new Lyrics \lyricsto "BassVoice" \BassLyrics
    >>
  >>
  \layout {
    ragged-right = ##t
    \context { \Staff
      % these lines prevent empty staves from being printed
      \RemoveEmptyStaves
      \override VerticalAxisGroup #'remove-first = ##t
    }
  }
}
```





A.4.6 Inni

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
}

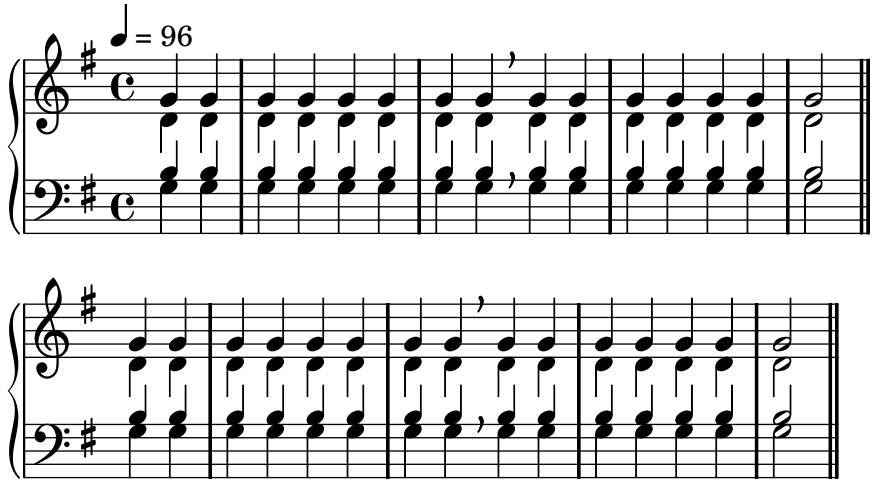
global = {
 \key g \major
}

\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

A.4.7 Salmi

Questo modello presenta un modo di impostare un salmo anglicano. Mostra anche come le strofe possono essere aggiunte come testo separato al di sotto della musica. Le due strofe sono scritte con stili diversi per illustrare le varie possibilità.

```
SopranoMusic = \relative g' {
  g1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative c' {
  e1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

TenorMusic = \relative a {
  c1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
}

BassMusic =  \relative c {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
}

dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}

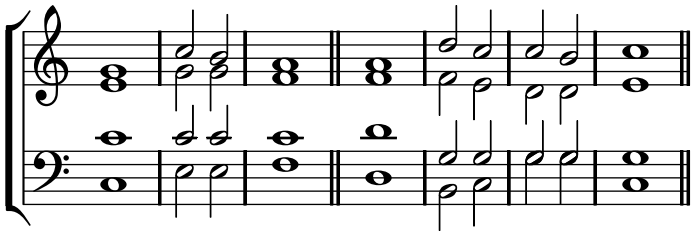
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score {  % centered
      <<
        \new ChoirStaff <<
          \new Staff <<
            \global
            \clef "treble"
            \new Voice = "Soprano" <<
              \voiceOne
              \SopranoMusic
            >>
            \new Voice = "Alto" <<
              \voiceTwo
              \AltoMusic
            >>
          >>
          \new Staff <<
            \clef "bass"
```

```
            \global
            \new Voice = "Tenor" <<
              \voiceOne
              \TenorMusic
            >>
            \new Voice = "Bass" <<
              \voiceTwo
              \BassMusic
            >>
          >>
        >>
      >>
      \layout {
        \context {
          \Score
          \override SpacingSpanner
          #'base-shortest-duration = #(ly:make-moment 1 2)
        }
        \context {
          \Staff
          \remove "Time_signature_engraver"
        }
      }
    }  % End score
  }
}  % End markup

\markup {
  \fill-line {
    \column {
      \left-align {
        \null \null \null
        \line {
          \fontsize #5 O
          \fontsize #3 come
          let us \bold sing | unto \dot the | Lord : let
        }
        \line {
          us heartily
          \concat { re \bold joice }
          in the | strength of | our
        }
        \line {
          sal | vation.
        }
        \null
        \line {
          \hspace #2.5 8. Today if ye will hear his voice *
        }
        \line {
          \concat { \bold hard en }
          \tick not your \tick hearts : as in the pro-
```

```
        }
        \line {
          vocation * and as in the \bold day of tempt- \tick
        }
        \line {
          -ation \tick in the \tick wilderness.
        }
      }
    }
  }
}
```



O come let us **sing** | unto • the | Lord : let us heartily re**joice** in the | strength of | our sal | vation.

8. Today if ye will hear his voice * **hard**en ' not your ' hearts : as in the provocation * and as in the **day** of tempt- ' -ation ' in the ' wilderness.

A.5 Modelli per orchestra

A.5.1 Orchestra, coro e pianoforte

Questo modello mostra come usare i contesti annidati `StaffGroup` e `GrandStaff` per creare sottogruppi degli strumenti dello stesso tipo. Mostra anche come usare `\transpose` in modo che le variabili mantengano la musica per gli strumenti traspositori nell'intonazione reale.

```
#(set-global-staff-size 17)
\paper {
  indent = 3.0\cm  % space for instrumentName
  short-indent = 1.5\cm  % space for shortInstrumentName
}

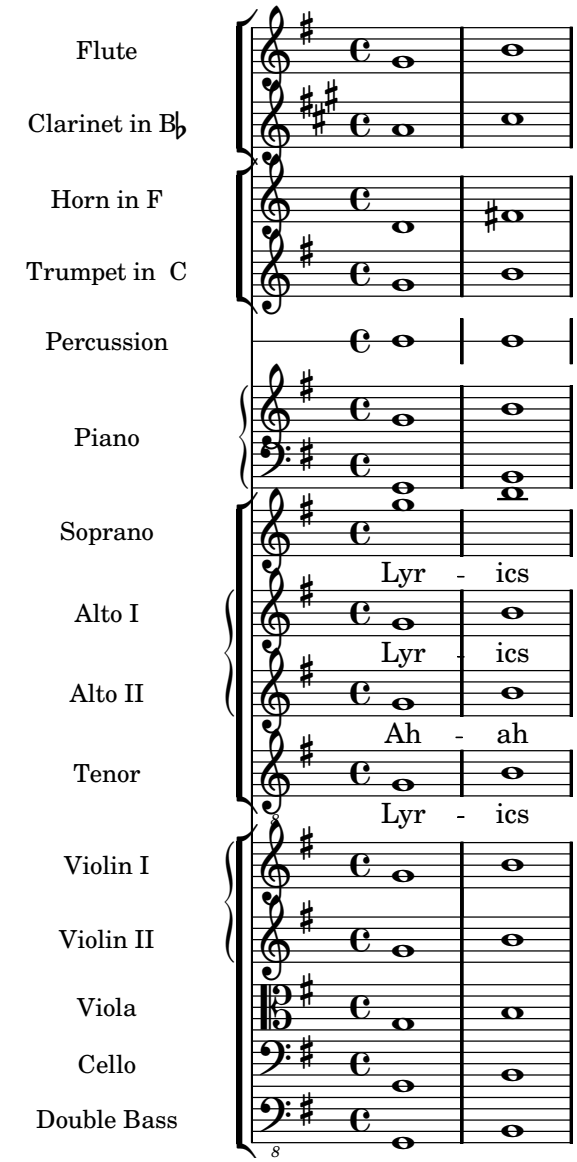
fluteMusic = \relative c' { \key g \major g'1 b }
% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.
clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }
trumpetMusic = \relative c { \key g \major g''1 b }
% Key signature is often omitted for horns
```

```
hornMusic = \transpose c' f
  \relative c { d'1 fis }
percussionMusic = \relative c { \key g \major g1 b }
sopranoMusic = \relative c'' { \key g \major g'1 b }
sopranoLyrics = \lyricmode { Lyr -- ics }
altoIMusic = \relative c' { \key g \major g'1 b }
altoIIMusic = \relative c' { \key g \major g'1 b }
altoILyrics =  \sopranoLyrics
altoIILyrics = \lyricmode { Ah -- ah }
tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }
tenorLyrics = \sopranoLyrics
pianoRHMusic = \relative c { \key g \major g''1 b }
pianoLHMusic = \relative c { \clef bass \key g \major g1 b }
violinIMusic = \relative c' { \key g \major g'1 b }
violinIIMusic = \relative c' { \key g \major g'1 b }
violaMusic = \relative c { \clef alto \key g \major g'1 b }
celloMusic = \relative c { \clef bass \key g \major g1 b }
bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\score {
  <<
    \new StaffGroup = "StaffGroup_woodwinds" <<
      \new Staff = "Staff_flute" {
        \set Staff.instrumentName = #"Flute"
        % shortInstrumentName, midiInstrument, etc.
        % may be set here as well
        \fluteMusic
      }
      \new Staff = "Staff_clarinet" {
        \set Staff.instrumentName =
        \markup { \concat { "Clarinet in B" \flat } }
        % Declare that written Middle C in the music
        %  to follow sounds a concert B flat, for
        %  output using sounded pitches such as MIDI.
        \transposition bes
        % Print music for a B-flat clarinet
        \transpose bes c' \clarinetMusic
      }
    >>
    \new StaffGroup = "StaffGroup_brass" <<
      \new Staff = "Staff_hornI" {
        \set Staff.instrumentName = #"Horn in F"
        \transposition f
        \transpose f c' \hornMusic
      }
      \new Staff = "Staff_trumpet" {
        \set Staff.instrumentName = #"Trumpet in  C"
        \trumpetMusic
      }
    >>
    \new RhythmicStaff = "RhythmicStaff_percussion" <<
      \set RhythmicStaff.instrumentName = #"Percussion"
```

```
  \percussionMusic
>>
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff { \pianoRHMusic }
  \new Staff { \pianoLHMusic }
>>
\new ChoirStaff = "ChoirStaff_choir" <<
  \new Staff = "Staff_soprano" {
    \set Staff.instrumentName = #"Soprano"
    \new Voice = "soprano"
    \sopranoMusic
  }
  \new Lyrics \lyricsto "soprano" { \sopranoLyrics }
  \new GrandStaff = "GrandStaff_altos"
  \with { \accepts Lyrics } <<
    \new Staff = "Staff_altoI"  {
      \set Staff.instrumentName = #"Alto I"
      \new Voice = "altoI"
      \altoIMusic
    }
    \new Lyrics \lyricsto "altoI" { \altoILyrics }
    \new Staff = "Staff_altoII" {
      \set Staff.instrumentName = #"Alto II"
      \new Voice = "altoII"
      \altoIIMusic
    }
    \new Lyrics \lyricsto "altoII" { \altoIILyrics }
  >>
  \new Staff = "Staff_tenor" {
    \set Staff.instrumentName = #"Tenor"
    \new Voice = "tenor"
    \tenorMusic
  }
  \new Lyrics \lyricsto "tenor" { \tenorLyrics }
>>
\new StaffGroup = "StaffGroup_strings" <<
  \new GrandStaff = "GrandStaff_violins" <<
    \new Staff = "Staff_violinI" {
      \set Staff.instrumentName = #"Violin I"
      \violinIMusic
    }
    \new Staff = "Staff_violinII" {
      \set Staff.instrumentName = #"Violin II"
      \violinIIMusic
    }
  >>
  \new Staff = "Staff_viola" {
    \set Staff.instrumentName = #"Viola"
    \violaMusic
  }
  \new Staff = "Staff_cello" {
```

```
        \set Staff.instrumentName = #"Cello"
        \celloMusic
      }
      \new Staff = "Staff_bass" {
        \set Staff.instrumentName = #"Double Bass"
        \bassMusic
      }
    >>
  >>
  \layout { }
}
```



A.6 Modelli per notazione antica

A.6.1 Transcrizione di musica mensurale

Quando si trascrive musica mensurale, può essere utile inserire all'inizio del pezzo un incipit che indichi l'intonazione e il tempo originali. Le stanghette di battuta, a cui i musicisti di oggi

sono abituati e che aiutano a riconoscere più velocemente gli schemi ritmici, durante l'epoca della musica mensurale non erano ancora state introdotte; infatti il metro cambiava spesso ogni poche note. Come compromesso, le stanghette vengono spesso inserite tra i righi invece che al loro interno.

```
global = {
  \set Score.skipBars = ##t

  % incipit
  \once \override Score.SystemStartBracket #'transparent = ##t
  \override Score.SpacingSpanner #'spacing-increment = #1.0 % tight spacing
  \key f \major
  \time 2/2
  \once \override Staff.TimeSignature #'style = #'neomensural
  \override Voice.NoteHead #'style = #'neomensural
  \override Voice.Rest #'style = #'neomensural
  \set Staff.printKeyCancellation = ##f
  \cadenzaOn % turn off bar lines
  \skip 1*10
  \once \override Staff.BarLine #'transparent = ##f
  \bar "||"
  \skip 1*1 % need this extra \skip such that clef change comes
            % after bar line
  \bar ""

  % main
  \revert Score.SpacingSpanner #'spacing-increment % CHECK: no effect?
  \cadenzaOff % turn bar lines on again
  \once \override Staff.Clef #'full-size-change = ##t
  \set Staff.forceClef = ##t
  \key g \major
  \time 4/4
  \override Voice.NoteHead #'style = #'default
  \override Voice.Rest #'style = #'default

  % FIXME: setting printKeyCancellation back to #t must not
  % occur in the first bar after the incipit.  Dto. for forceClef.
  % Therefore, we need an extra \skip.
  \skip 1*1
  \set Staff.printKeyCancellation = ##t
  \set Staff.forceClef = ##f

  \skip 1*7 % the actual music

  % let finis bar go through all staves
  \override Staff.BarLine #'transparent = ##f

  % finis bar
  \bar "|."
}

discantusNotes = {
```

```
  \transpose c' c'' {
    \set Staff.instrumentName = #"Discantus  "

    % incipit
    \clef "neomensural-c1"
    c'1. s2   % two bars
    \skip 1*8 % eight bars
    \skip 1*1 % one bar

    % main
    \clef "treble"
    d'2. d'4 |
    b e' d'2 |
    c'4 e'4.( d'8 c' b |
    a4) b a2 |
    b4.( c'8 d'4) c'4 |
    \once \override NoteHead #'transparent = ##t c'1 |
    b\breve |
  }
}

discantusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- |
  la -- te De -- |
  o, om --
  nis ter -- |
  ra, __ om- |
  "..." |
  -us. |
}

altusNotes = {
  \transpose c' c'' {
    \set Staff.instrumentName = #"Altus  "

    % incipit
    \clef "neomensural-c3"
    r1        % one bar
    f1. s2    % two bars
    \skip 1*7 % seven bars
    \skip 1*1 % one bar

    % main
    \clef "treble"
    r2 g2. e4 fis g | % two bars
    a2 g4 e |
    fis g4.( fis16 e fis4) |
    g1 |
```

```
    \once \override NoteHead #'transparent = ##t g1 |
    g\breve |
  }
}

altusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- la -- te | % two bars
  De -- o, om -- |
  nis ter -- ra, |
  "..." |
  -us. |
}

tenorNotes = {
  \transpose c' c' {
    \set Staff.instrumentName = #"Tenor  "

    % incipit
    \clef "neomensural-c4"
    r\longa   % four bars
    r\breve   % two bars
    r1        % one bar
    c'1. s2   % two bars
    \skip 1*1 % one bar
    \skip 1*1 % one bar

    % main
    \clef "treble_8"
    R1 |
    R1 |
    R1 |
    r2 d'2. d'4 b e' | % two bars
    \once \override NoteHead #'transparent = ##t e'1 |
    d'\breve |
  }
}

tenorLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi -- la -- te | % two bars
  "..." |
  -us. |
}

bassusNotes = {
```

```
  \transpose c' c' {
    \set Staff.instrumentName = #"Bassus  "

    % incipit
    \clef "bass"
    r\maxima  % eight bars
    f1. s2    % two bars
    \skip 1*1 % one bar

    % main
    \clef "bass"
    R1 |
    R1 |
    R1 |
    R1 |
    g2. e4 |
    \once \override NoteHead #'transparent = ##t e1 |
    g\breve |
  }
}

bassusLyrics = \lyricmode {
  % incipit
  IV-

  % main
  Ju -- bi- |
  "..." |
  -us. |
}

\score {
  \new StaffGroup = choirStaff <<
    \new Voice =
      "discantusNotes" << \global \discantusNotes >>
    \new Lyrics =
      "discantusLyrics" \lyricsto discantusNotes { \discantusLyrics }
    \new Voice =
      "altusNotes" << \global \altusNotes >>
    \new Lyrics =
      "altusLyrics" \lyricsto altusNotes { \altusLyrics }
    \new Voice =
      "tenorNotes" << \global \tenorNotes >>
    \new Lyrics =
      "tenorLyrics" \lyricsto tenorNotes { \tenorLyrics }
    \new Voice =
      "bassusNotes" << \global \bassusNotes >>
    \new Lyrics =
      "bassusLyrics" \lyricsto bassusNotes { \bassusLyrics }
  >>
  \layout {
    \context {
```

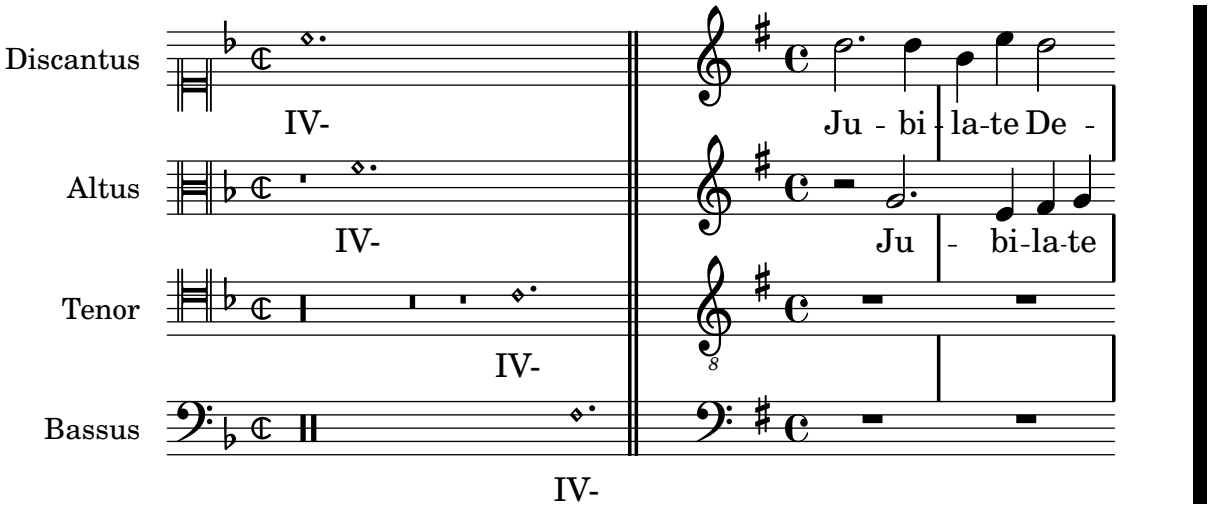
```
      \Score
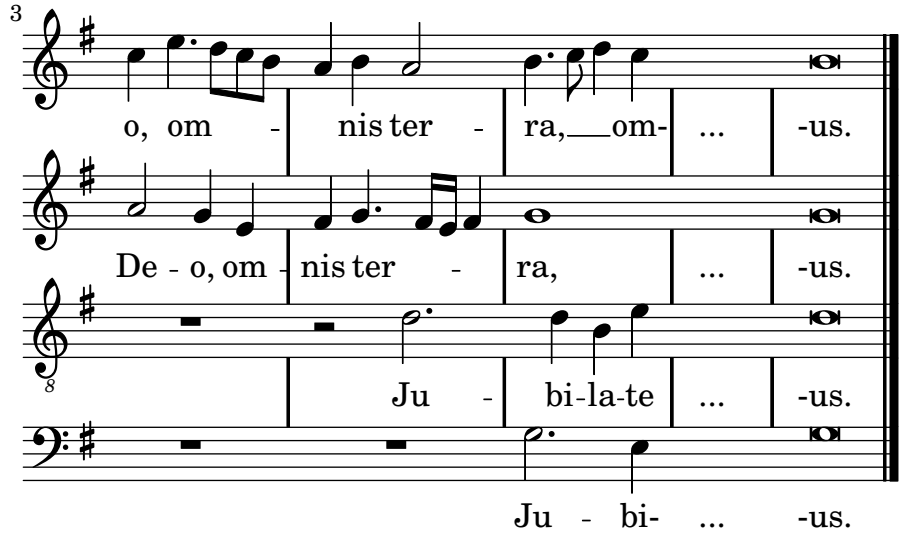
      % no bars in staves
      \override BarLine #'transparent = ##t

      % incipit should not start with a start delimiter
      \remove "System_start_delimiter_engraver"
    }
    \context {
      \Voice

      % no slurs
      \override Slur #'transparent = ##t

      % Comment in the below "\remove" command to allow line
      % breaking also at those barlines where a note overlaps
      % into the next bar.  The command is commented out in this
      % short example score, but especially for large scores, you
      % will typically yield better line breaking and thus improve
      % overall spacing if you comment in the following command.
      %\remove "Forbid_line_break_engraver"
    }
  }
}
```





A.6.2 Trascrizione di musica Gregoriana

Questo esempio mostra come realizzare una trascrizione moderna di musica gregoriana. La musica gregoriana non presenta suddivisione in misure né gambi; utilizza soltanto le teste della minima e della semiminima, e dei segni appositi che indicano pause di diversa lunghezza.
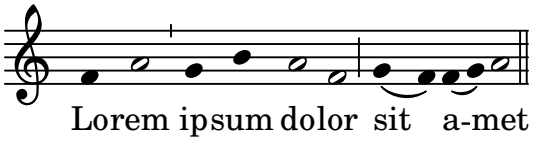
```
\include "gregorian.ly"

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"
      \override Stem #'transparent = ##t
    }
    \context {
      \Voice
      \override Stem #'length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
```

```
}
```



A.7 Altri modelli

A.7.1 Combo jazz

Ecco un modello piuttosto complesso, per un gruppo jazz. Si noti che tutti gli strumenti sono in `\key c \major`. Si tratta della tonalità reale; sarà trasposta automaticamente includendo la musica all'interno di una sezione `\transpose`.

```
\header {
  title = "Song"
  subtitle = "(tune)"
  composer = "Me"
  meter = "moderato"
  piece = "Swing"
  tagline = \markup {
    \column {
      "LilyPond example file by Amelie Zapf,"
      "Berlin 07/07/2003"
    }
  }
}

%#(set-global-staff-size 16)
\include "english.ly"

%%%%%%%%%% Some macros %%%%%%%%%%%%%%%%%%

sl = {
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
}
nsl = {
  \revert NoteHead #'style
  \revert Stem #'transparent
}
crOn = \override NoteHead #'style = #'cross
crOff = \revert NoteHead #'style

%% insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%% Keys'n'thangs %%%%%%%%%%%%%%%%%

global = { \time 4/4 }
```

```
Key = { \key c \major }

% ############ Horns ############

% ------ Trumpet ------
trpt = \transpose c d \relative c'' {
  \Key
  c1 | c | c |
}
trpHarmony = \transpose c' d {
  \jazzChords
}
trumpet = {
  \global
  \set Staff.instrumentName = #"Trumpet"
  \clef treble
  <<
    \trpt
  >>
}

% ------ Alto Saxophone ------
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \set Staff.instrumentName = #"Alto Sax"
  \clef treble
  <<
    \alto
  >>
}

% ------ Baritone Saxophone ------
bari = \transpose c a' \relative c {
  \Key
  c1
  c1
  \sl
  d4^"Solo" d d d
  \nsl
}
bariHarmony = \transpose c' a \chordmode {
  \jazzChords s1 s d2:maj e:m7
}
bariSax = {
```

```
  \global
  \set Staff.instrumentName = #"Bari Sax"
  \clef treble
  <<
    \bari
  >>
}

% ------ Trombone ------
tbone = \relative c {
  \Key
  c1 | c | c
}
tboneHarmony = \chordmode {
  \jazzChords
}
trombone = {
  \global
  \set Staff.instrumentName = #"Trombone"
  \clef bass
  <<
    \tbone
  >>
}

% ############ Rhythm Section ##############

% ------ Guitar ------
gtr = \relative c'' {
  \Key
  c1
  \sl
  b4 b b b
  \nsl
  c1
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 c2:min7+ d2:maj9
}
guitar = {
  \global
  \set Staff.instrumentName = #"Guitar"
  \clef treble
  <<
    \gtr
  >>
}

%% ------ Piano ------
rhUpper = \relative c'' {
  \voiceOne
```

```
  \Key
  c1 | c | c
}
rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g
}
lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c
}

PianoRH = {
  \clef treble
  \global
  \set Staff.midiInstrument = #"acoustic grand"
  <<
    \new Voice = "one" \rhUpper
    \new Voice = "two" \rhLower
  >>
}
PianoLH = {
  \clef bass
  \global
  \set Staff.midiInstrument = #"acoustic grand"
  <<
    \new Voice = "one" \lhUpper
    \new Voice = "two" \lhLower
  >>
}

piano = {
  <<
    \set PianoStaff.instrumentName = #"Piano"
    \new Staff = "upper" \PianoRH
    \new Staff = "lower" \PianoLH
  >>
}

% ------ Bass Guitar ------
Bass = \relative c {
  \Key
  c1 | c | c
}
```

```
bass = {
  \global
  \set Staff.instrumentName = #"Bass"
  \clef bass
  <<
    \Bass
  >>
}

% ------ Drums ------
up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn>
  hh4 <hh sn> hh <hh sn>
  hh4 <hh sn> hh <hh sn>
}
down = \drummode {
  \voiceTwo
  bd4 s bd s
  bd4 s bd s
  bd4 s bd s
}

drumContents = {
  \global
  <<
    \set DrumStaff.instrumentName = #"Drums"
    \new DrumVoice \up
    \new DrumVoice \down
  >>
}
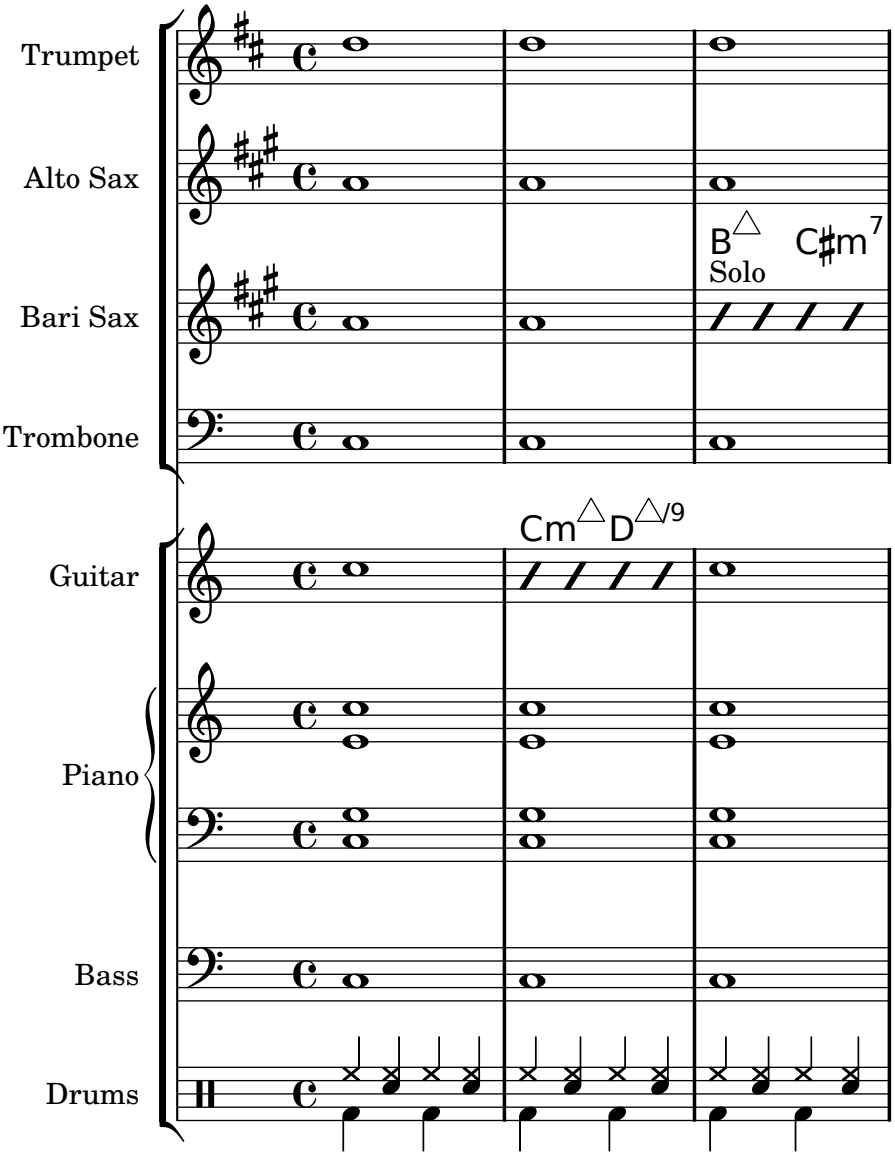
%%%%%%%% It All Goes Together Here %%%%%%%%%%%%%%%%%%%%%

\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \trumpet
      \new Staff = "altosax" \altoSax
      \new ChordNames = "barichords" \bariHarmony
      \new Staff = "barisax" \bariSax
      \new Staff = "trombone" \trombone
    >>

    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \gtrHarmony
      \new Staff = "guitar" \guitar
      \new PianoStaff = "piano" \piano
      \new Staff = "bass" \bass
      \new DrumStaff \drumContents
    >>
  >>
```

```
  \layout {
    \context { \Staff \RemoveEmptyStaves }
    \context {
      \Score
      \override BarNumber #'padding = #3
      \override RehearsalMark #'padding = #2
      skipBars = ##t
    }
  }
  \midi { }
}
```




Trumpet
Alto Sax
Bari Sax
Trombone
Guitar
Piano
Bass
Drums
B△
Solo
C♯m7
Cm△
D△/9

Appendice B GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendice C Indice di LilyPond

!

! 25

%

% 18
%{ ... %} 18

'

' 14

(

(...) 23

,

, ... 14

.

... . 18

<

< 25, 31
< ... > 31
<< 28, 31
<< ... >> 28
<< ... \\ ... >> 31
<< \\ >> 48

>

> 25, 31
>> 28, 31

[

[..... 26
[...] 26

]

] 26

^

^ 24

_

_ 24

\

\! 25
\(... \) 23
\< 25
\> 25
\\ 31, 48
\acciaccatura 27
\addlyrics 32
\addlyrics example 92
\addlyrics, example 96
\appoggiatura 27
\autoBeamOff 26, 56
\autoBeamOn 26
\book 41, 42
\clef 17
\consists 67
\context 66
\f 25
\ff 25
\grace 27
\header 38, 42
\key 22
\layout 42, 68
\lyricmode 56
\lyricsto 56
\major 22
\markup 25
\mf 25
\midi 42
\minor 22
\mp 25
\new 29, 60
\new ChoirStaff 56
\new Lyrics 56
\new Staff 29
\new Voice 52
\once 89, 93
\oneVoice 52
\override 88
\overrideProperty 89
\p 25
\partial 26
\pp 25
\relative 14
\remove 67
\revert 89, 94
\score 41, 43
\set 63
\set, example of using 109
\shiftOff 55
\shiftOn 55
\shiftOnn 55
\shiftOnnn 55
\startTextSpan 111
\stopTextSpan 111
\tempo 16
\textLengthOff 113
\textLengthOn 113
\time 16
\times 27
\tweak 90
\tweak, example 90, 91
\unset 63
\version 18

\voiceFour ... 52
\voiceFourStyle ... 50
\voiceNeutralStyle ... 50
\voiceOne ... 52
\voiceOneStyle ... 50
\voiceThree ... 52
\voiceThreeStyle ... 50
\voiceTwo ... 52
\voiceTwoStyle ... 50
\with ... 65
\with, example ... 102, 103, 104, 105

~

~ ... 23

A

abbellimenti ... 27
accento ... 24
acciaccatura ... 27
Accidental, example of overriding ... 119
AccidentalPlacement, example of overriding ... 119
accollatura ... 30
accordi ... 31
accordi vs. voci ... 48
addlyrics ... 32
aggiungere gli incisori ... 67
aggiungere testo ... 25
alignAboveContext property, example ... 102, 103, 104, 105
aligning objects on a baseline ... 119
allineare il testo ... 32
allungabilità dei righi ... 80
alterazioni e armature di chiave ... 21, 22
alterazioni e modo relativo ... 14
alterazioni, accidenti ... 21
altezze ... 14
altezze, valori assoluti ... 39
ambitus, incisore dell' ... 67
anacrusi ... 26
annidare i costrutti simultanei ... 54
annidare le espressioni musicali ... 54
annidare le voci ... 54
apostrofo ... 14
appoggiatura ... 27
armatura di chiave ... 22
armatura di chiave, impostare ... 21
articolazione ... 24
articulations and slurs ... 112
assegnare le variabili ... 37
autoBeamOff ... 26, 56
autoBeamOn ... 26

B

bar numbers, tweaking placement ... 112
BarLine, example of overriding ... 98, 99, 100, 101
basso ... 17
battuta parziale ... 26
Beam, example of overriding ... 122
beams, controlling manually ... 121
bemolle ... 21
bemolle, doppio ... 21
blocco dell'intestazione ... 38
blocco, commento di ... 18
book ... 41, 42
book (libro) ... 41
bound-details property, example ... 111, 112
bracket, triplet ... 91
bracket, tuplet ... 91
break-visibility property ... 99
break-visibility property, example ... 99

C

canzoni ... 32
caratteri permessi nelle variabili ... 37
center ... 107
changing size of objects ... 102
chiave ... 17
chiave di violino ... 17
ChoirStaff ... 30, 56
ChordNames ... 29
clef ... 17
Clef, example of overriding ... 101, 102, 103, 104, 105
cliccabili, esempi ... 19
code automatiche ... 26
code manuali ... 26
code, a mano ... 26
code, automatiche ... 26
code, manuali ... 26
collisioni di note ... 55
colonna delle note ... 55
color property ... 100
color property, example ... 88, 89, 90, 91, 100, 101
color property, setting to Scheme procedure ... 140
color, rgb ... 101
color, X11 ... 101
comandi di spostamento ... 55
combinare le espressioni in parallelo ... 28
come leggere il manuale ... 19
commenti ... 18
commento di blocco ... 18
commento di linea ... 18
compilazione ... 1
composta, espressione musicale ... 43
consigli su come costruire i file ... 19
consists ... 67
contenuto di un blocco score ... 43
contenuto vs. layout ... 22
contesti della voce, creazione dei ... 52
contesti impliciti ... 41
contesti, creazione di ... 60
contesti, dare un nome ... 61
contesti, impliciti ... 41
contesti, spiegazione dei ... 59
contesto ... 29
contesto della notazione ... 29
contesto Voice (voce) ... 48
contesto, notazione ... 29
contesto, proprietà del ... 63
contesto, proprietà del, impostare con \context ... 66
contesto, proprietà del, impostare con \with ... 65
contesto, proprietà del, modificare ... 63
context ... 66
context, finding ... 93
context, identifying correct ... 93
context, specifying in lyric mode ... 96

contralto ... 17
controlling tuplets, slurs, phrasing slurs, and beams manually ... 121
coro, pentagramma per ... 30
costruire i file, consigli ... 19
creazione di contesti ... 60
crescendo ... 25

D

dare un nome ai contesti ... 61
decrescendo ... 25
default properties, reverting to ... 94
diesis ... 21
diesis, doppio ... 21
dinamica ... 25
direction property, example ... 91, 107, 108
distances ... 105
diteggiatura ... 24
doppio bemolle ... 21
doppio diesis ... 21
down ... 107
durate ... 15
durate della nota ... 15
durate delle note negli accordi ... 31
DynamicLineSpanner, example of overriding ... 120
dynamics, tweaking placement ... 114
DynamicText, example of overriding ... 115, 120

E

`Easier editing` ... 2, 6
`Easier editing` ... 13
`Editing facilitato` ... 1
editor di testo ... 1
errori comuni ... 19
`es` ... 21
eseguire LilyPond in MacOS X ... 2
eseguire LilyPond in Unix ... 13
eseguire LilyPond in Windows ... 6
esempi cliccabili ... 19
esempio di scrittura di una partitura ... 78
esempio iniziale ... 1
esempio, iniziale ... 1
`eses` ... 21
espressione musicale ... 27
espressione musicale composta ... 27, 43
espressioni ... 18
espressioni parallele ... 28
espressioni parallele e note relative ... 28
extra-offset property ... 117
extra-offset property, example ... 121
`extra-spacing-width` ... 115
extra-spacing-width property ... 117
extra-spacing-width property, example ... 115, 120

F

fermata, implementing in MIDI ... 132
file, consigli per costruirli ... 19
fingering example ... 108, 109
fingering, chords ... 108
Fingering, example of overriding ... 108, 121
fingering, placement ... 108
fingeringOrientations property, example ... 109
fixing overlapping notation ... 118
font-series property, example ... 133
font-shape property, example ... 96, 133
font-size property, example ... 90
fontSize property, example ... 104, 105
fontSize, impostazione predefinita e impostazione manuale ... 66
force-hshift property ... 117
force-hshift property, example ... 123, 129
formato di input ... 41

G

gambi e direzione delle voci ... 52
gambo in giù ... 52
gambo in su ... 52
`grace` ... 27
graffe, parentesi ... 18
grafici, oggetti ... 80
`GrandStaff` ... 30
grob ... 80, 87
grob sizing ... 115
grobs, moving colliding ... 116
grobs, positioning ... 121
grobs, properties of ... 91
gruppi irregolari ... 27
gruppo di pentagrammi ... 30

H

`header` ... 38, 42
hiding objects ... 131

I

identificatori ... 37
implicito, blocco del libro ... 42
impostare le proprietà all'interno dei contesti ... 63
incisori ... 62
incisori, aggiungere ... 67
incisori, rimuovere ... 67
indicazioni di tempo ... 16
indicazioni metronomiche ... 16
insensibile agli spazi ... 18
interface ... 87, 95
interface properties ... 95
Internals Reference ... 91
Internals Reference manual ... 91
Internals Reference, example of using ... 92
intestazioni ... 38
invisible objects ... 131
`is` ... 21
`isis` ... 21
italic, example ... 96

K

`key` ... 22

L

`layout` ... 42, 68
layout object ... 87

layout objects, properties of ... 91
layout vs. contenuto ... 22
layout, effetto della posizione del blocco ... 42
legatura di fraseggio ... 23
legatura di portamento ... 23
legatura di portamento, fraseggio ... 23
legatura di valore ... 23
legature che attraversano le parentesi ... 49
legature di portamento vs. legature di valore ... 24
leggere il manuale ... 19
length ... 105
libro ... 41
libro, blocco implicito ... 42
linea di estensione ... 32
linea, commento di ... 18
livelli ... 48
lyric mode, specifying context ... 96
`lyricmode` ... 56
`Lyrics` ... 29, 56
Lyrics, creazione di un contesto ... 56
`lyricsto` ... 56
LyricText, example of overriding ... 96, 133

M

MacOS X, eseguire LilyPond ... 2
macro ... 37
maggiore ... 22
magstep ... 105
magstep function, example of using ... 105
maiuscole, sensibile alle ... 1
`major` ... 22
manuale, leggere ... 19
`Manuali` ... 1
manually controlling tuplets, slurs, phrasing slurs, and beams ... 121
`markup` ... 25
markup example ... 106
markup text, allowing collisions ... 113
melisma ... 32
metronome mark, tweaking placement ... 112
MetronomeMark, example of overriding ... 118, 132
`midi` ... 42
minima ... 15
`minor` ... 22
minore ... 22
modelli ... 19
modelli, SATB ... 73
modello, modificare ... 70
modificare i modelli ... 70
modificare le proprietà del contesto ... 63
modo assoluto ... 39
modo relativo ... 14
modo relativo e polifonia ... 50
modo relativo, e alterazioni ... 14
moving colliding grobs ... 116
moving colliding objects ... 116
moving overlapping objects ... 116
MultiMeasureRest, example of overriding ... 120
multipli, righi ... 28, 29
musica simultanea ... 48
musica simultanea e note relative ... 28
musica sincrona ... 48

N

naming conventions for objects ... 88
naming conventions for properties ... 88
neutral ... 107
`new` ... 29, 60
`new Staff` ... 29
nomi assoluti delle note ... 39
nomi delle note ... 39
nomi delle note, assoluti ... 39
nota puntata ... 15
notazione delle durate ... 15
notazione delle pause ... 16
notazione semplice ... 13
notazione, semplice ... 13
note relative e espressioni parallele ... 28
note relative e musica simultanea ... 28
note spaziatrici ... 55
note, collisioni di ... 55
NoteColumn, example of overriding ... 123, 129
NoteHead, example of overriding ... 88, 89, 90, 101, 140
notes, spreading out with text ... 113
numero di versione ... 18
nuovi contesti ... 60

O

object ... 87
object collision within a staff ... 120
object properties ... 87
object, layout ... 87
objects, aligning on a baseline ... 119
objects, changing size of ... 102
objects, hiding ... 131
objects, invisible ... 131
objects, making invisible ... 131
objects, moving colliding ... 116
objects, naming conventions ... 88
objects, outside-staff ... 106
objects, positioning ... 121
objects, removing ... 131
objects, size of ... 102
objects, within-staff ... 106
oggetti grafici ... 80
`once` ... 89, 93
once override ... 93
`oneVoice` ... 52
ossia ... 45
ottava bracket ... 111
outside-staff objects ... 106
outside-staff-priority property, example ... 113
overlapping notation ... 118
`override` ... 88
override command ... 88
override example ... 92
override syntax ... 88
`overrideProperty` ... 89
overrideProperty command ... 89
overriding once only ... 93

P

padding ... 116, 118
padding property ... 116

padding property, example 118
parallele, espressioni 28
parentesi graffe 18
parentesi, annidare 47
parentesi, racchiudere vs. contrassegnare 47
parole con sillabe multiple nel testo 32
`partial` 26
partitura 41, 43
partitura, esempio di scrittura 78
partiture, multiple 42
pausa 16
pausa spaziatrice 31
PDF file 1
pentagramma per coro 30
pentagramma per piano 30
phrasing slurs, controlling manually 121
PhrasingSlur, example of overriding 121
piano, pentagramma per 30
`PianoStaff` 30
polifonia 28, 31, 48
polifonia e modo relativo 50
polifonia su un singolo rigo 31
positioning grobs 121
positioning objects 121
positions property 118
positions property, example 121, 122
properties in interfaces 95
properties of grobs 91
properties of layout objects 91
properties, naming conventions 88
properties, object 87
property types 97
proprietà che operano nei contesti 63
proprietà, sottoproprietà 80

R

raggruppamento 26
rehearsal marks, tweaking placement 112
`relative` 14
`remove` 67
removing objects 131
`revert` 89, 94
revert command 89
rgb colors 101
`rgb-color` 101
righi multipli 28, 29
righi multipli e testo 36
righi temporanei 45
righi, allungabilità 80
right-padding property 116, 119
right-padding property, example 119
rigo per un coro 56
rigo, posizionamento del 46
rimuovere gli incisori 67
risoluzione dei problemi 19
ritmi 15

S

SATB, modelli 73
SATB, struttura di 57
`score` 41, 43
`Score` 29
score (partitura) 41
score, contenuto del blocco 43
Script, example of overriding 118
scrivere una partitura, esempio 78
self-alignment-X property 117
self-alignment-X property, example 120
semibreve 15
semiminima 15
sensibile alle maiuscole 18
`set` 63
`shiftOff` 55
`shiftOn` 55
`shiftOnn` 55
`shiftOnnn` 55
simultanea, musica 48
sincrona, musica 48
size of objects 102
size, changing 105
sizing grobs 115
Slur example of overriding 93
Slur, example of overriding 94, 95
slurs and articulations 112
slurs and outside-staff-priority 112
slurs, controlling manually 121
sottoproprietà 80
spanner 87
spanners 110
spaziatrice, pausa 31
staccato 24
`Staff` 29
staff line spacing, changing 105
staff-padding property 116
staff-padding property, example 120
staff-position property 117
staff-position property, example 120, 128
staff-space property, example 105
StaffSymbol, example of overriding 101, 105
`startTextSpan` 111
stem length, changing 105
Stem, example of overriding 101, 107, 129, 131
stencil property 98
stencil property, example 98, 100, 102, 105, 119, 132
stencil property, use of 132
`stopTextSpan` 111
StringNumber, example of overriding 120
strofe, vocali, varie 57
struttura del file 41
struttura di un inno 57
struttura di una partitura vocale 56

T

template, scrivere il tuo 78
`tempo` 16
tempo, indicazione di 16
temporanei, righi 45
tenore 17
terzine 27
testi 32
testo e righi multipli 36
testo e travatura 56
testo, aggiungere 25
testo, allineare 32
testo, collegare a una voce 56

testo, creazione di un contesto ... 56
testo, parole polisillabiche ... 32
text property, example ... 91, 119
text spanner ... 111
`textLengthOff` ... 113
`textLengthOn` ... 113
TextScript, example of overriding ... 113
TextSpanner, example of overriding ... 111, 112
thickness ... 105
thickness property, example ... 93, 94, 95
Tie, example of overriding ... 128
`time` ... 16
`times` ... 27
TimeSignature, example of overriding ... 100, 101, 102, 103, 104, 105
tipi di parentesi ... 47
titolo ... 38
tornare alla voce singola ... 53
transparency ... 100
transparent property ... 100
transparent property, example ... 91, 100, 129, 131, 132
transparent property, use of ... 131
trattini ... 32
trattino basso ... 32
travatura e testo ... 56
triplet bracket ... 91
triplets, nested ... 91
tuplet beams, controlling manually ... 121
tuplet bracket ... 91
tuplet-number function, example ... 91
`TupletBracket` ... 91
TupletNumber, example of overriding ... 91
tuplets, nested ... 91
`tweak` ... 90
tweak command ... 90
tweaking bar number placement ... 112
tweaking dynamics placement ... 114
tweaking methods ... 88
tweaking metronome mark placement ... 112
tweaking rehearsal mark placement ... 112
tweaks, using variables for ... 133
tying notes across voices ... 131

U

Unix, eseguire LilyPond ... 13
`unset` ... 63
up ... 107
usare le variabili ... 37
using variables for tweaks ... 133

V

valori assoluti per le altezze ... 39
variabili ... 37, 43, 83
variabili, caratteri permessi nelle ... 37
variabili, definire ... 37
variabili, usare ... 37
variables, using for tweaks ... 133
varie strofe vocali ... 57
varie voci ... 48
vedere la musica ... 1
versionamento ... 18
versione ... 18
virgola ... 14
vocale, partitura, varie strofe ... 57
voci che attraversano le parentesi ... 49
voci e direzione dei gambi ... 52
voci multiple ... 31
voci temporanee ... 54
voci vs. accordi ... 48
voci, annidare ... 54
voci, nome delle ... 49
voci, più su un rigo ... 31
voci, tornare alla voce singola ... 53
voci, varie ... 48
`Voice` ... 29
Voice (voce), contesto di ... 48
`voiceFour` ... 52
`voiceOne` ... 52
`voiceThree` ... 52
`voiceTwo` ... 52

W

Windows, eseguire LilyPond ... 6
`with` ... 65
within-staff objects ... 106

X

X11 colors ... 101
`x11-color` ... 101
x11-color function, example of using ... 140
x11-color, example of using ... 101