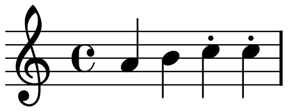
+.ly

0.1 Introduction

This document shows all kinds of tips and tricks, from simple to advanced. You may also find dirty tricks, or the very very latest features that have not been documented or fully implemented yet. This document is for LilyPond version 2.6.3.

add-staccato.ly

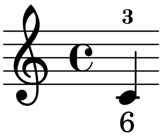
Using `make-music`, you can add various stuff to notes. In this example staccato dots are added to the notes. For this simple case, it is not necessary to use scm constructs (see `separate-staccato.ly`).



add-text-script.ly

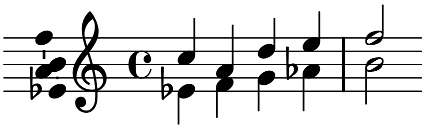
You can add various stuff to notes using `make-music`. In this example, an extra fingering is attached to a note.

In general, first do a `display` of the music you want to create, then write a function that will structure the music for you.



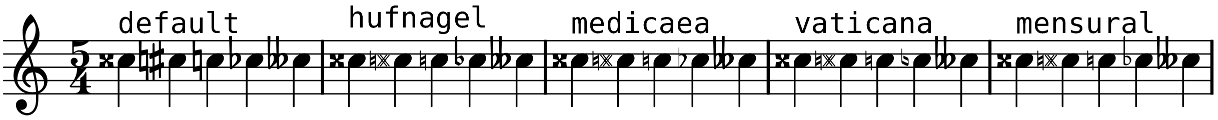
ambitus-mixed.ly

Ambits can be added per voice. In that case, the ambitus must be moved manually to prevent collisions.



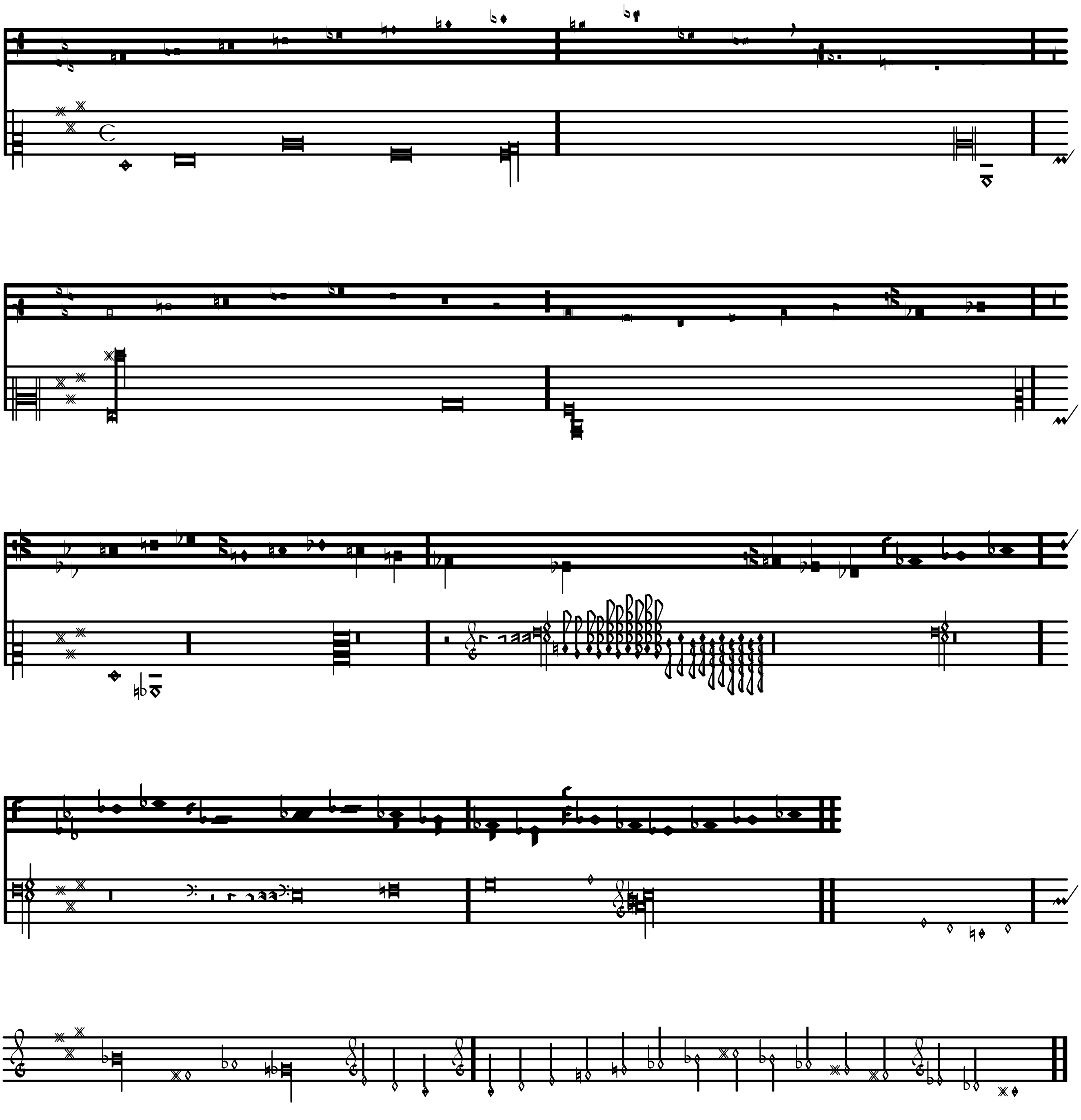
ancient-accidentals.ly

Accidentals are available in different ancient styles, which all are collected here.



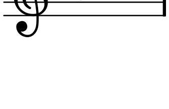
`ancient-font.ly`

Here are shown many (all?) of the symbols that are included in LilyPond's support of ancient notation.



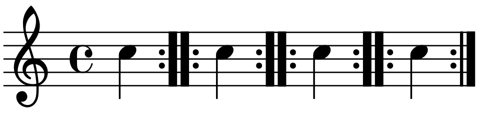
`ancient-time.ly`

Time signatures may also be engraved in an old style.



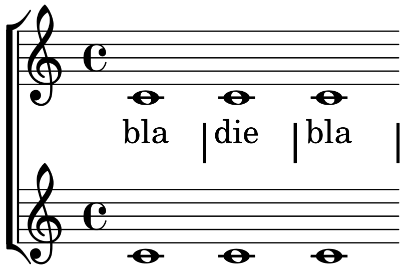
`bar-always.ly`

By setting `barAlways` and `defaultBarType`, barlines may be inserted automatically everywhere.



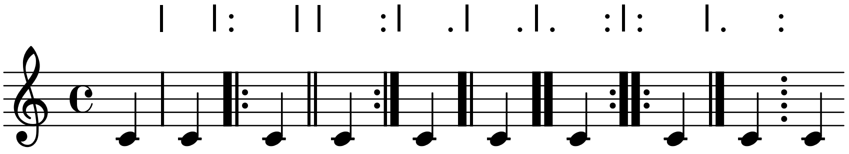
`bar-lines-lyric-only.ly`

You can move `Bar_engraver` and `Span_bar_engraver` to a different engraving context, if you want, for example, bar lines on lyrics.



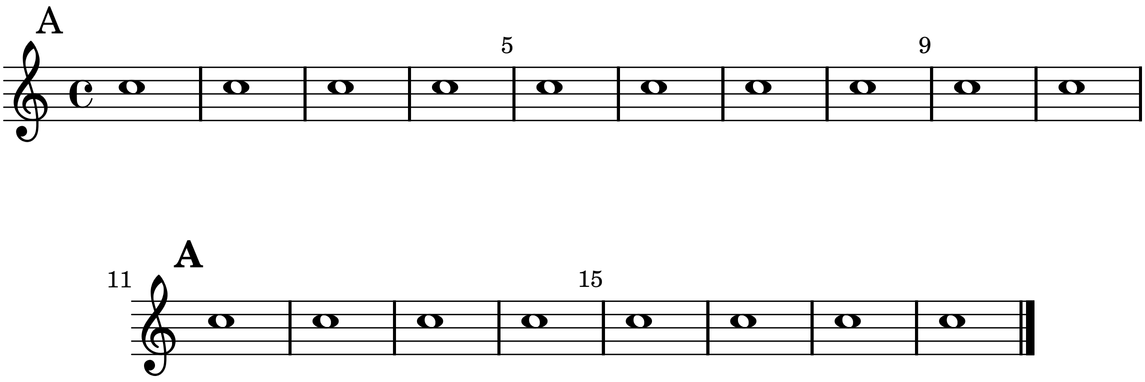
`bar-lines.ly`

There a many types of bar lines available.



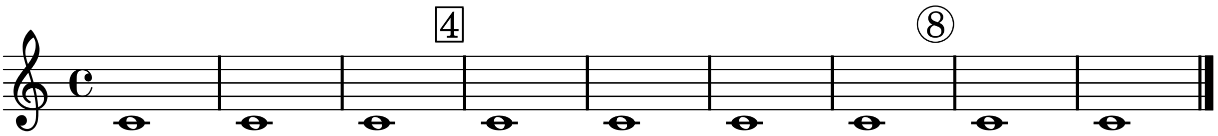
`bar-number-every-five-reset.ly`

If you would like the bar numbers to appear at regular intervals, but not starting from measure zero, you can use a context function, `set-bar-number-visibility`, to set automatically `barNumberVisibility`, so that the bar numbers appear at regular intervals, starting from the measure in which `set-bar-number-visibility` is set using `\applycontext`.



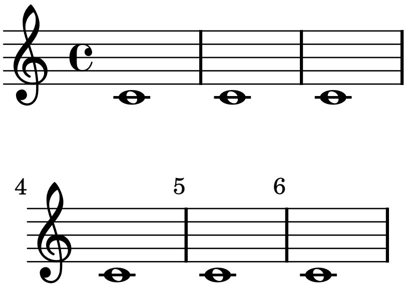
`bar-number-regular-interval.ly`

Bar numbers can be printed at regular intervals, inside a box or a circle.



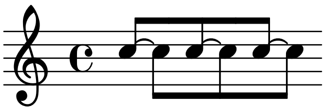
`bar-number-show-all.ly`

By default, bar numbers are printed only in the first measure. This setting can be overridden, so that bar numbers on start of every measure.



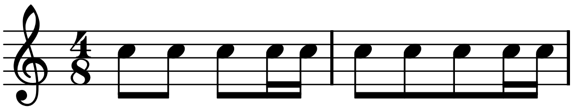
`beam-alternate.ly`

The eighth notes may be seemingly attached to different beams, and the corresponding notes connected by ties (see also ‘`tie-cross-voice.ly`’). Such a situation may occur, for example, in the cello suites.



`beam-auto-4-8.ly`

You can override the automatic beaming settings.

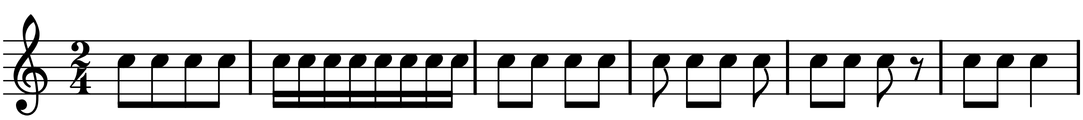


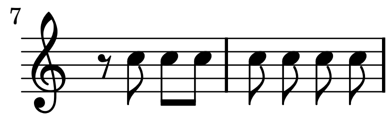
`beam-auto-override.ly`

The auto-beamer, which can be overridden, will only engrave beams that end before encountering of

- a rest,
- an other, manually entered beam, or
- a bar line.

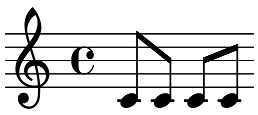
The `autoBeaming` can also be turned off.





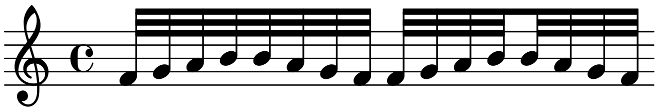
beam-control.ly

Beam positions may be controlled manually, by overriding the `positions` setting of the `Beam` grob.



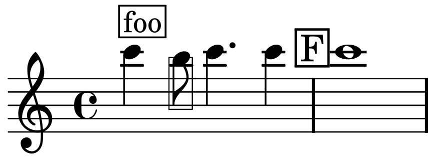
beam-count.ly

You can alter the number of stems in a beam. In this example, two sets of four 32nds are joined, as if they were 8th notes.



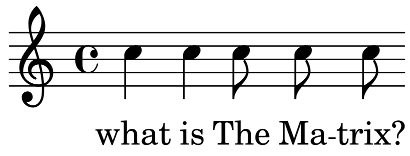
boxed-stencil.ly

The `print-function` can be overridden to draw a box around an arbitrary grob.



caps.ly

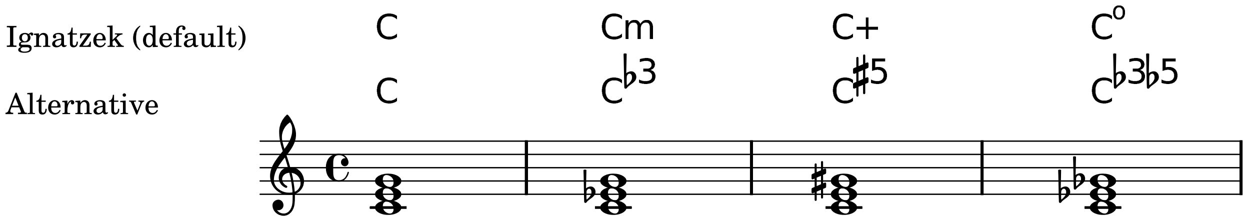
The font can be changed to small caps.

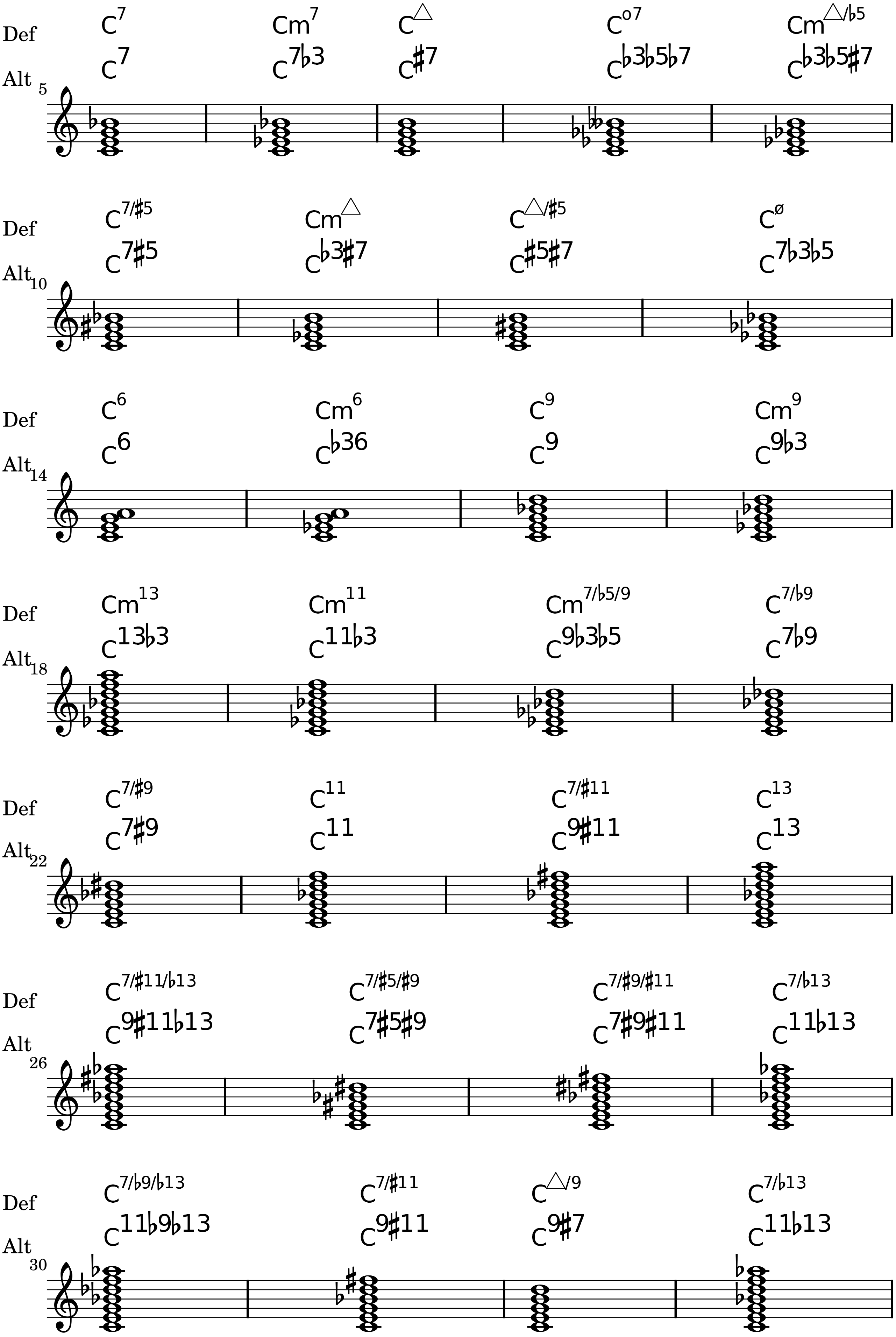


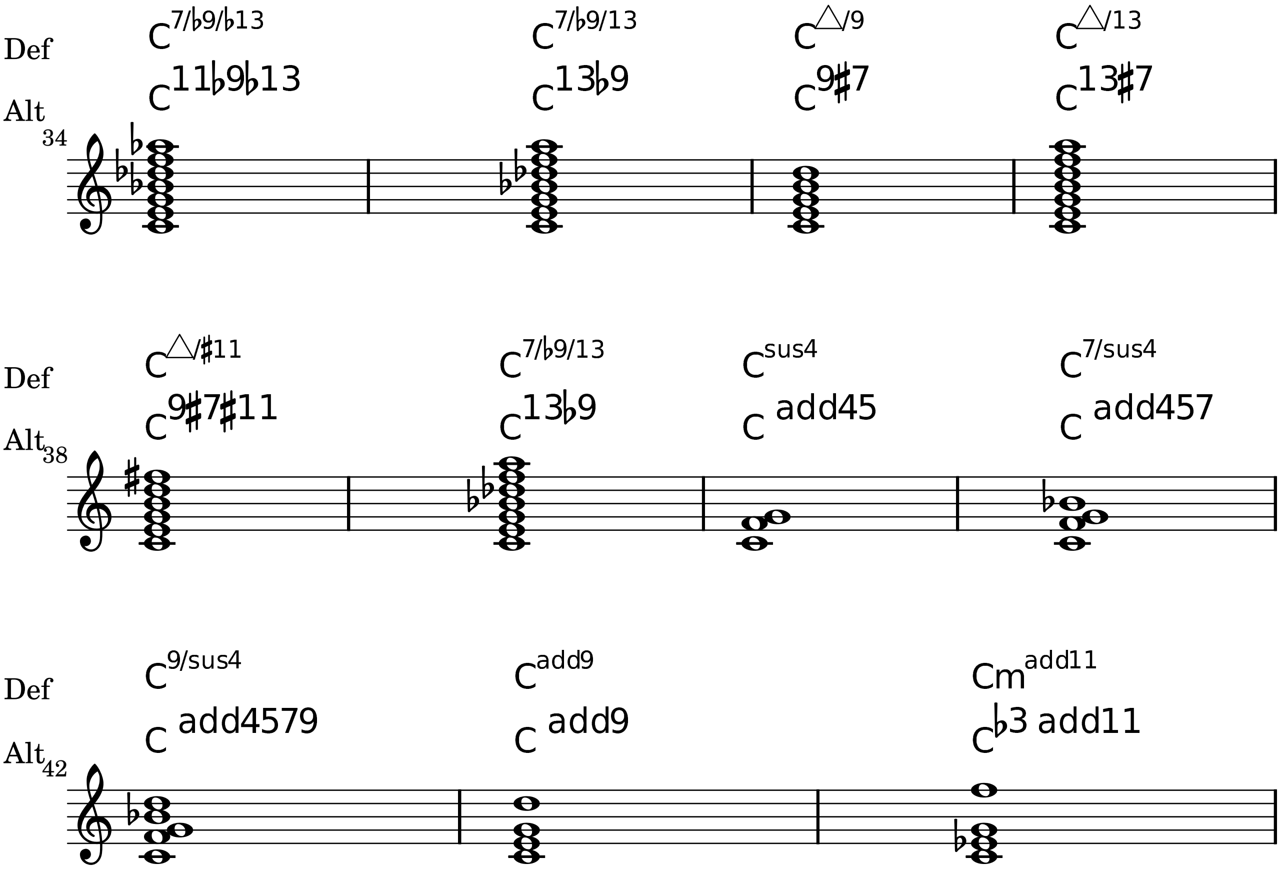
chord-names-jazz.ly

Chord names are generated from a list pitches. The functions which construct these names can be customised. Here are shown Jazz chords, following Ignatzek (pp. 17-18, 1995) and an alternative Jazz chord notation.

Chords following Banter (1987) can also be printed from this file, but are turned off for brevity.

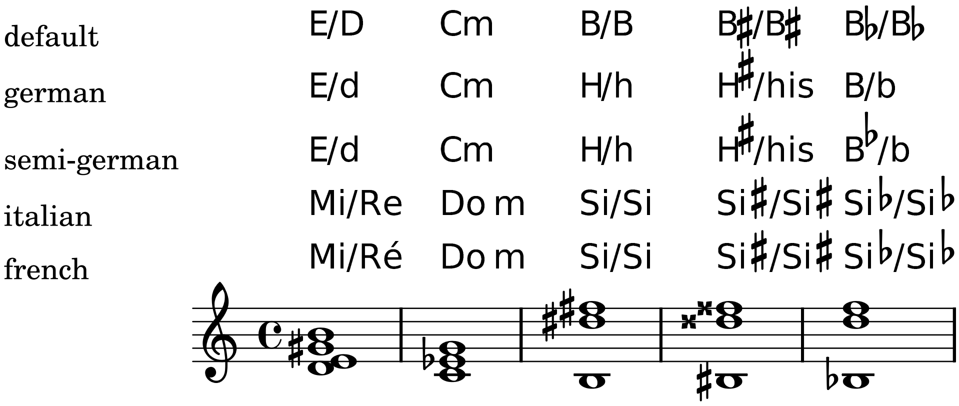



Def
Alt
5
C7
C7
Cm7
C7♭3
C△
C♯7
Co7
C♭3♭5♭7
Cm△/♭5
C♭3♭5♯7
Def
Alt
10
C7/♯5
C7♯5
Cm△
C♭3♯7
C△/♯5
C♯5♯7
Cø
C7♭3♭5
Def
Alt
14
C6
C6
Cm6
C♭36
C9
C9
Cm9
C9♭3
Def
Alt
18
Cm13
C13♭3
Cm11
C11♭3
Cm7/♭5/9
C9♭3♭5
C7/♭9
C7♭9
Def
Alt
22
C7/♯9
C7♯9
C11
C11
C7/♯11
C9♯11
C13
C13
Def
Alt
26
C7/♯11/♭13
C9♯11♭13
C7/♯5/♯9
C7♯5♯9
C7/♯9/♯11
C7♯9♯11
C7/♭13
C11♭13
Def
Alt
30
C7/♭9/♭13
C11♭9♭13
C7/♯11
C9♯11
C△/9
C9♯7
C7/♭13
C11♭13



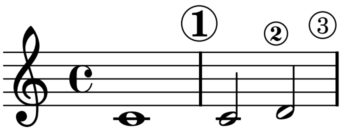
`chord-names-languages.ly`

The english naming of chords (default) can be changed to german (`\germanChords` replaces B and Bes to H and B), semi-german (`\semiGermanChords` replaces B and Bes to H and Bb), italian (`\italianChords` uses Do Re Mi Fa Sol La Si), or french (`\frenchChords` replaces Re to R).



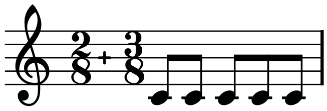
`circle.ly`

Circles can be drawn around various objects.



`compound-time.ly`

Compound time signatures can be printed. Automatic beaming works in compound time.



`coriolan-margin.ly`

In an orchestral score (Beethoven's Coriolan overture), there are different instrument groups, and some of the instruments may be transposed. Instruments are indicated either with a long or short name.

Ouvertüre

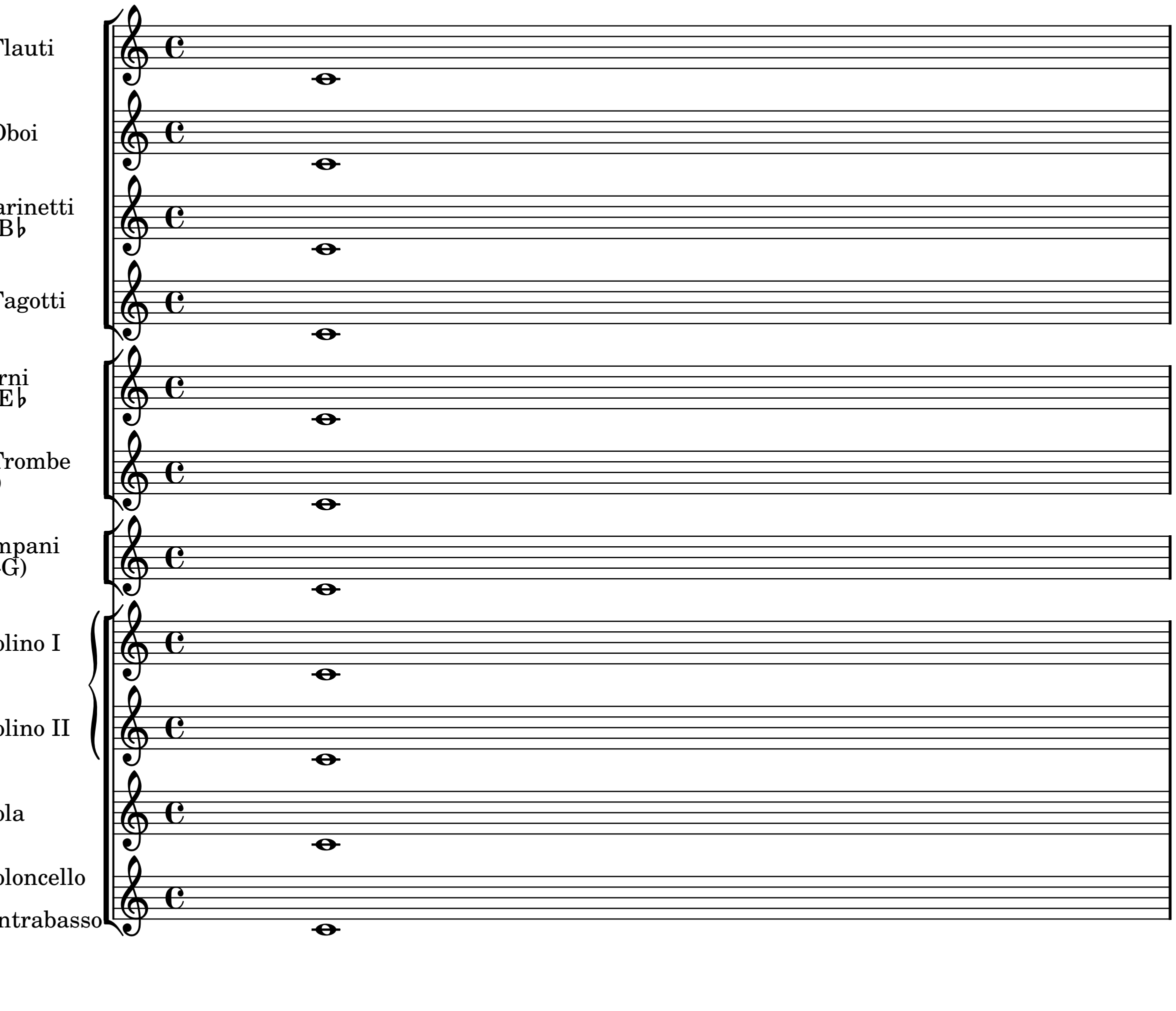
Zu Heinrich Joseph v. Collins Trauerspiel

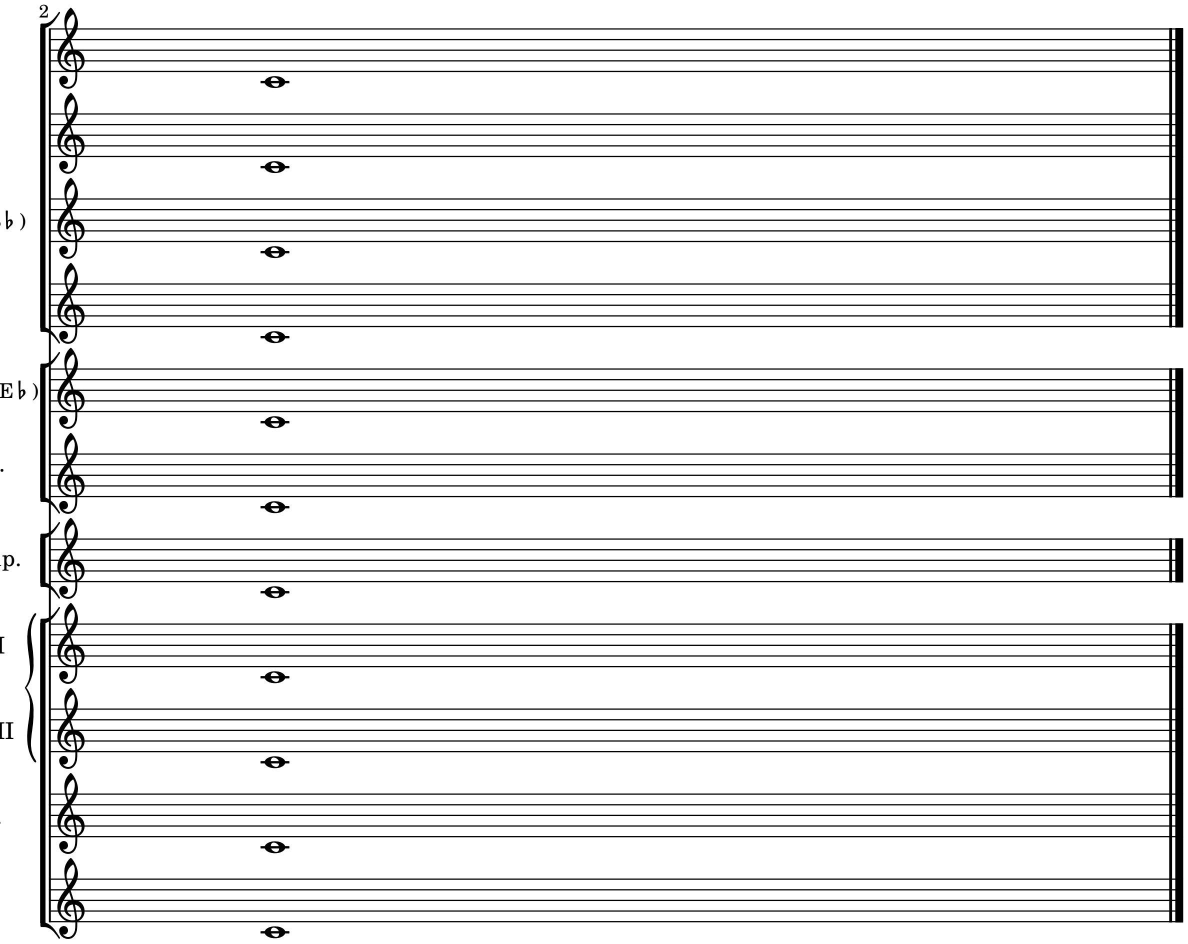
Coriolan

Ludwig van Beethoven (1770-1827)

Allegro con brio

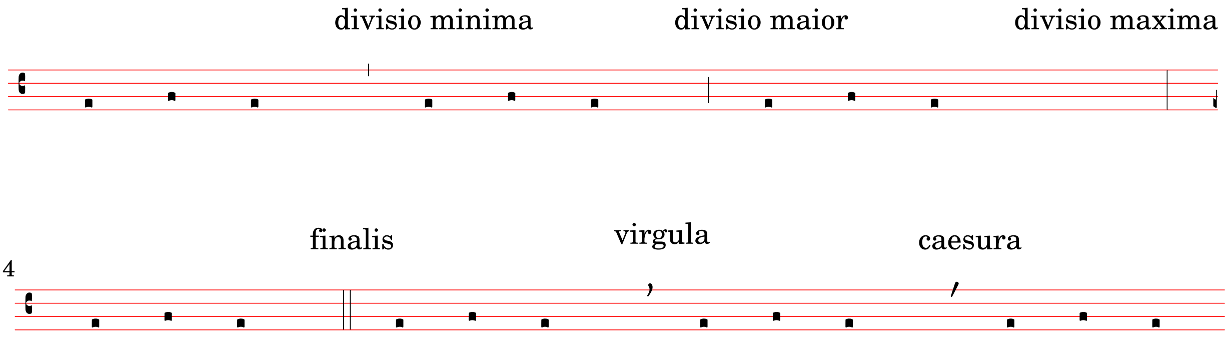
Op. 62




2
♭)
E♭)
.
p.
I
II
.

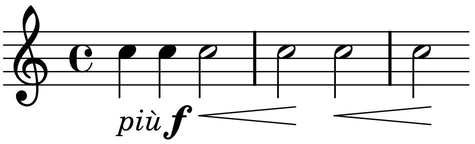
divisiones.ly

Divisiones are ancient variants of breathing signs. Choices are divisioMinima, divisioMaior, divisioMaxima and finalis, virgula and caesura.



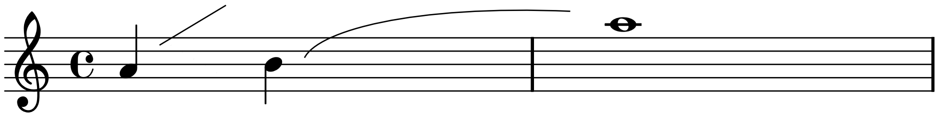
dynamic-extra.ly

Pi forte dynamics is produced using \markup.



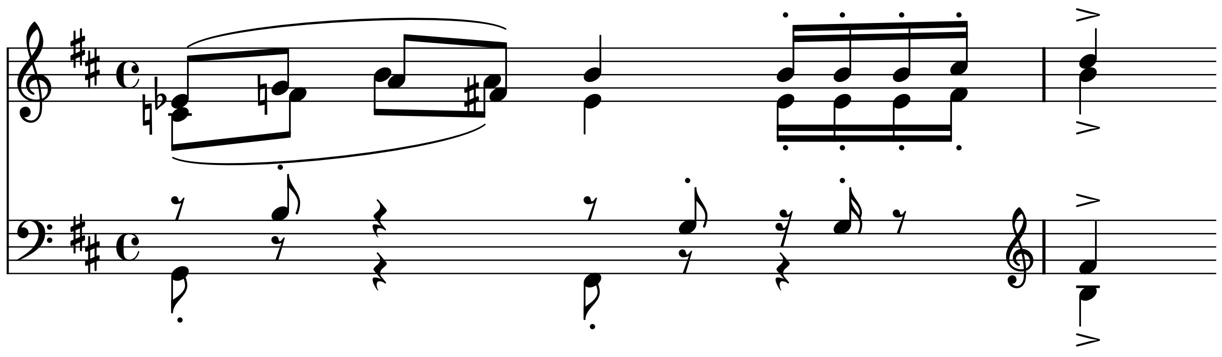
embedded-postscript.ly

The markup command \postscript inserts postscript directly into the output.



engraver-contexts.ly

In polyphonic notation, many voices can share a staff: In this situation, the accidentals and staff are shared, but the stems, slurs, beams, etc. are private to each voice. Hence, engravers should be grouped. The engravers for note head, stems, slurs, etc. go into a group called “Voice context”, while the engravers for key, accidental, bar, etc. go into a group called “Staff context”. In the case of polyphony, a single Staff context contains more than one Voice context. Similarly, more Staff contexts can be put into a single Score context.

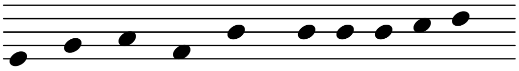


`engraver-one-by-one.ly`

The notation problem, creating a certain symbol, is handled by plugins. Each plugin is called Engraver. In this example, engravers are switched on one by one, in the following order:

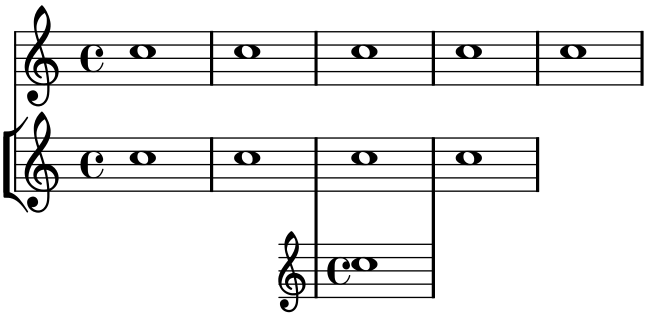
- note heads,
- staff symbol,
- clef,
- stem,
- beams, slurs, accents,
- accidentals, bar lines, time signature, and key signature.

Engravers are grouped. For example, note heads, slurs, beams etc. form a Voice context. Engravers for key, accidental, bar, etc. form a Staff context.



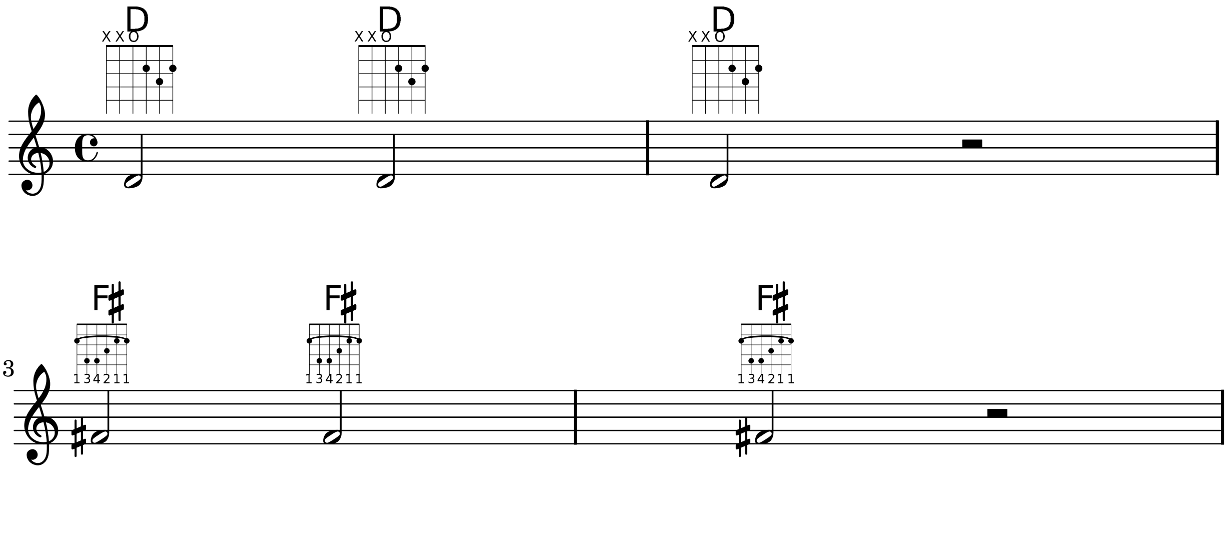
`extra-staff.ly`

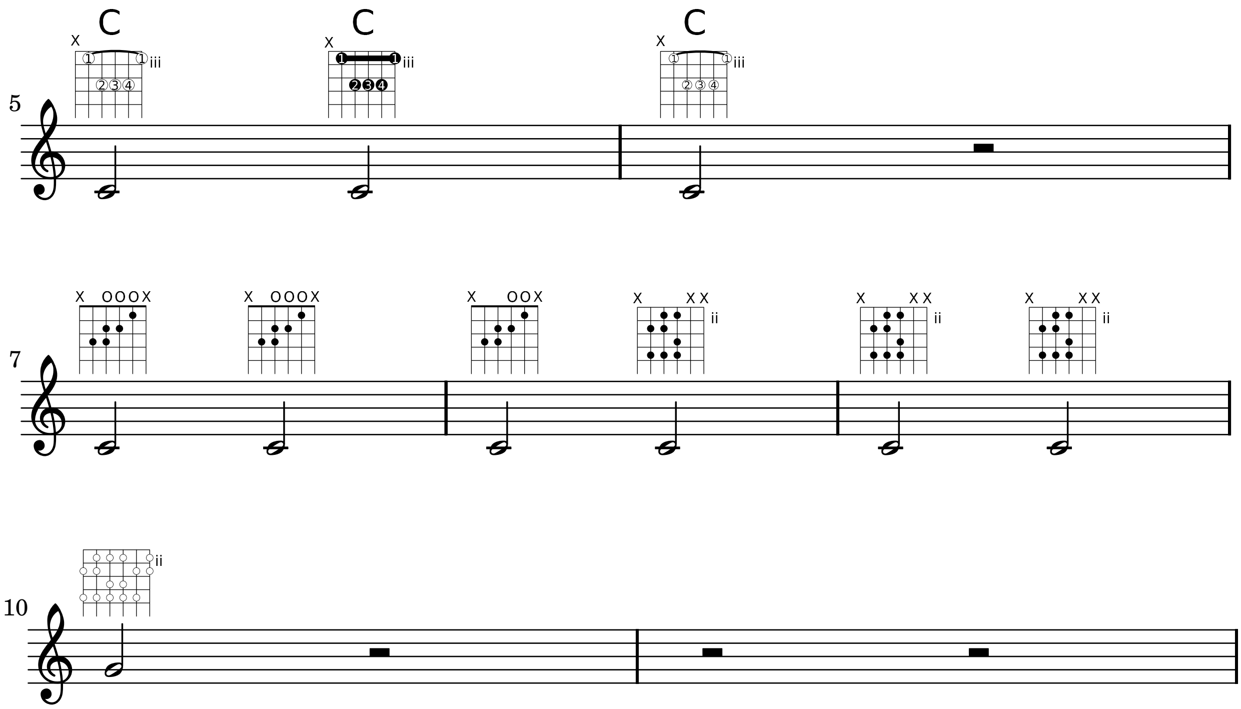
You can add (possibly temporarily) an extra staff after the beginning of a piece.



`fret-diagram.ly`

Frets are supported as markup commands.

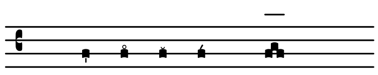




`gregorian-scripts.ly`

Here is demonstrated a preliminary support of Gregorian Scripts:

ictus, circulus, semicirculus, accentus, episem.

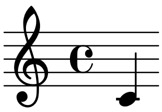


`header-ifelse.ly`

High level functionality (eg. conditional defines), can be accomplished with GUILE.

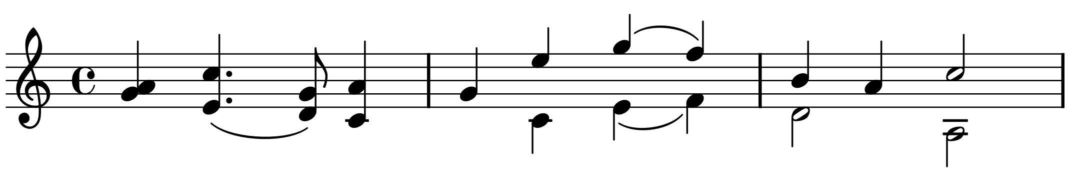
This example puts the current version in the title via Scheme.

Title has version 2.6.3



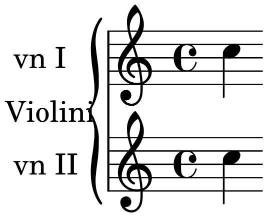
`hymn.ly`

You can combine two parts on the same staff using the part combiner. For vocal scores (hymns), there is no need to add solo/a2 texts, so they should be switched off.



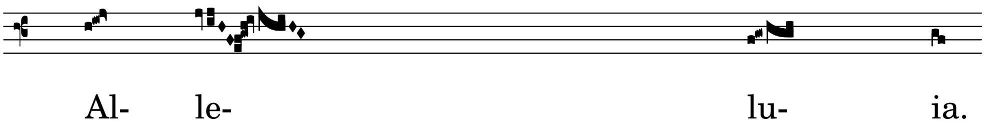
`instrument-name-grandstaff.ly`

You can have a name for the whole `GrandStaff` in addition to individual `Staffs`.



`ligature-vaticana.ly`

Vaticana ligature uses four staff lines, special clef, and calligraphic notes.



`lilypond-testpage.ly`

All header fields with special meanings.

localtitle

localsubtitle

localpoet

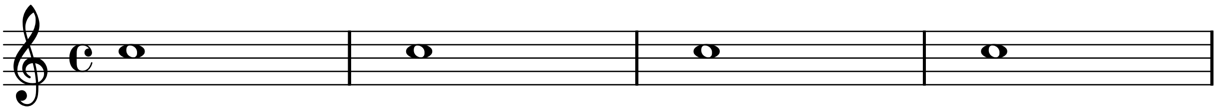
localinstrument

localcomposer

localarranger

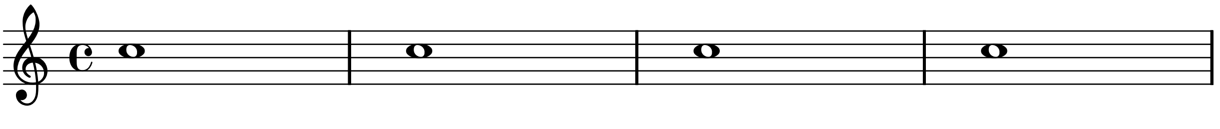
localpiece

localopus



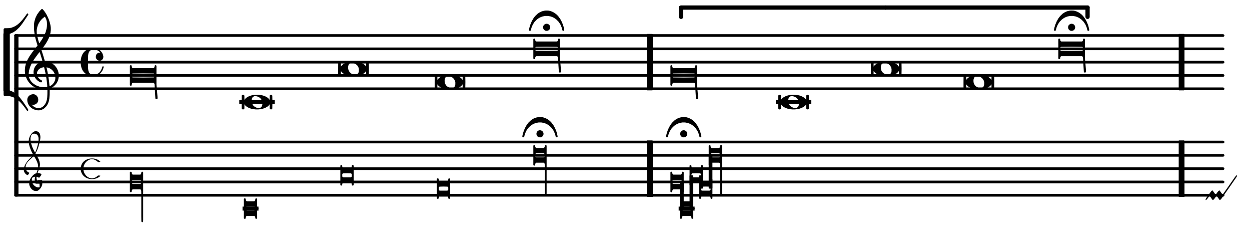
piece

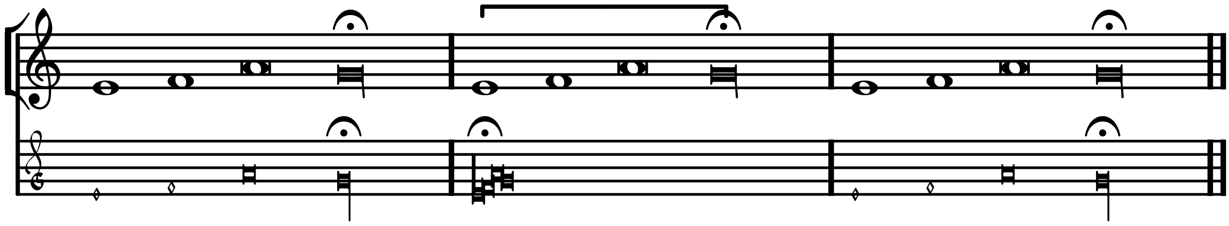
opus



`mensural-ligatures.ly`

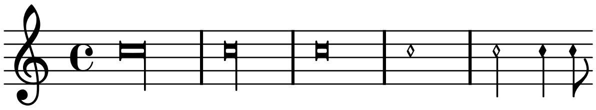
In mensural ligatures, notes with ancient durations are printed in a tight manner.





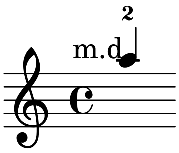
`mensural-note-heads.ly`

Mensural notes may also have note heads.



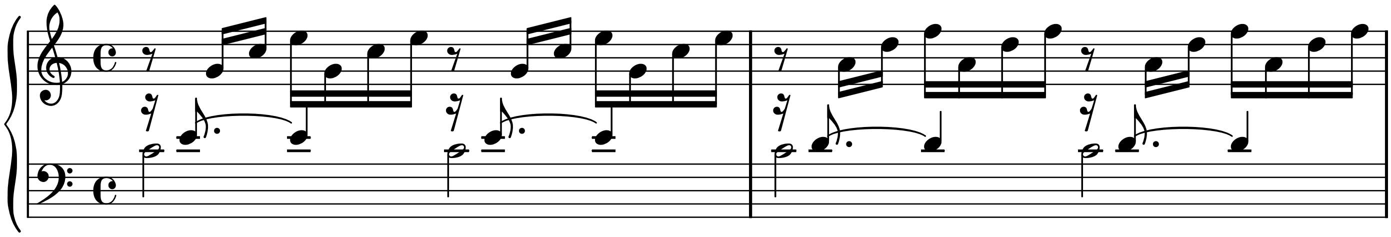
`move-specific-text.ly`

Objects, like text, can be moved around by using some Scheme code.



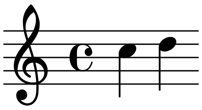
`music-box.ly`

This example shows prelude in C major of WTK1, but coded using Scheme functions to avoid typing work.



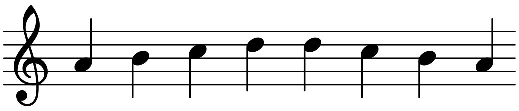
`music-creation.ly`

You can engrave music using just Scheme expressions. Although those expressions reflect the inner mechanism of LilyPond, they are rather clumsy to use, so avoid them, if possible.



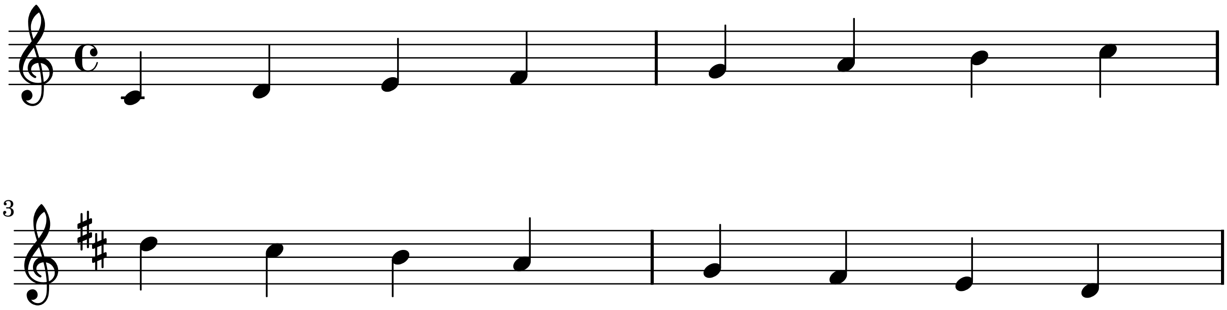
`no-bar-lines.ly`

Engravers can be removed one by one. Here, the time signature and bar lines have been removed.



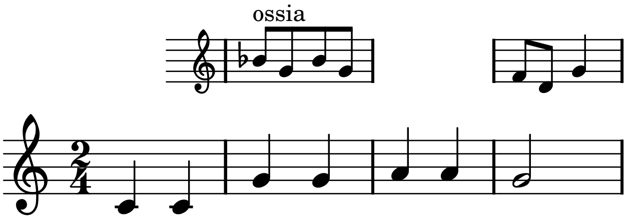
`no-key-at-end-of-line.ly`

According to normal typesetting conventions, LilyPond typesets key changes at the end of the line, when the change appears at a line break. This example shows how to change this default to only print the new key signature at the beginning of the next line.



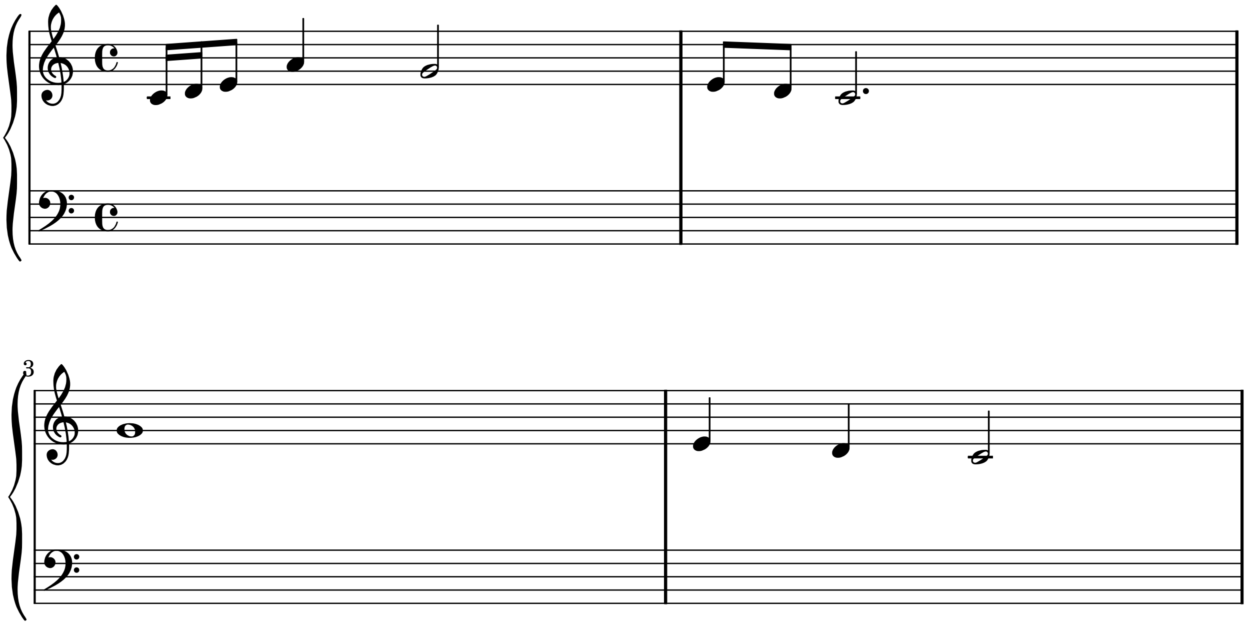
`ossia.ly`

Ossia fragments can be done with starting and stopping staves.



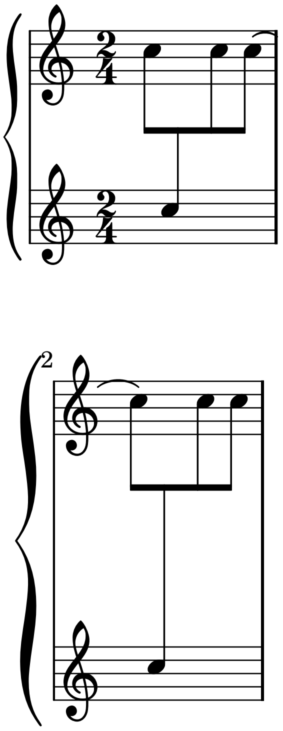
`partial-blank.ly`

When entering partially typeset music (i.e. for students to be completed by hand), you may need the spacing that correspond to the timing of notes: all measures have same length, etc. It can be implemented by adding an invisible staff with a lot of fast notes.



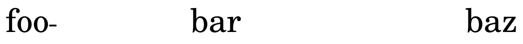
`piano-staff-distance.ly`

It is possible to have different staff distances between the staffs of a piano system, but it requires some advanced Scheme code. Currently, this is for testing purposes.



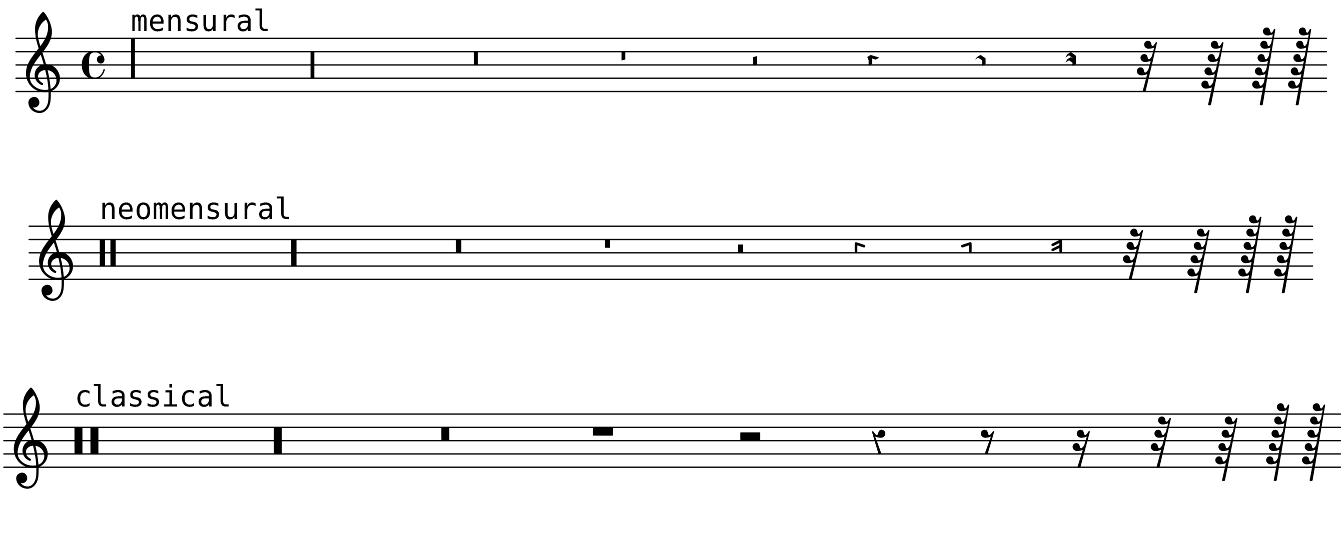
`preset-extent.ly`

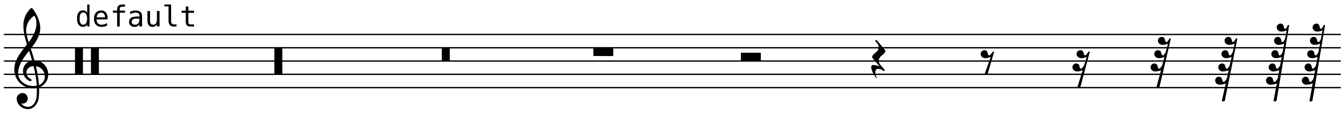
The object may be extended to larger sized by overriding their properties. The lyrics in this example have an extent of `(-10,10)`, which is why they are spaced so widely.



`rests.ly`

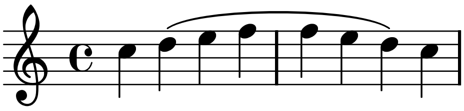
Rests may be used in various styles.





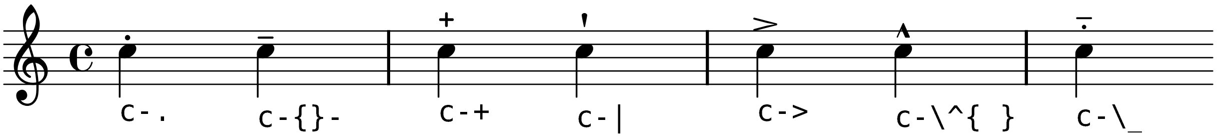
`reverse-music.ly`

Symmetric, or palindromical music can be produced, first, by printing some music, and second, by printing the same music applying a Scheme function to reverse the syntax.



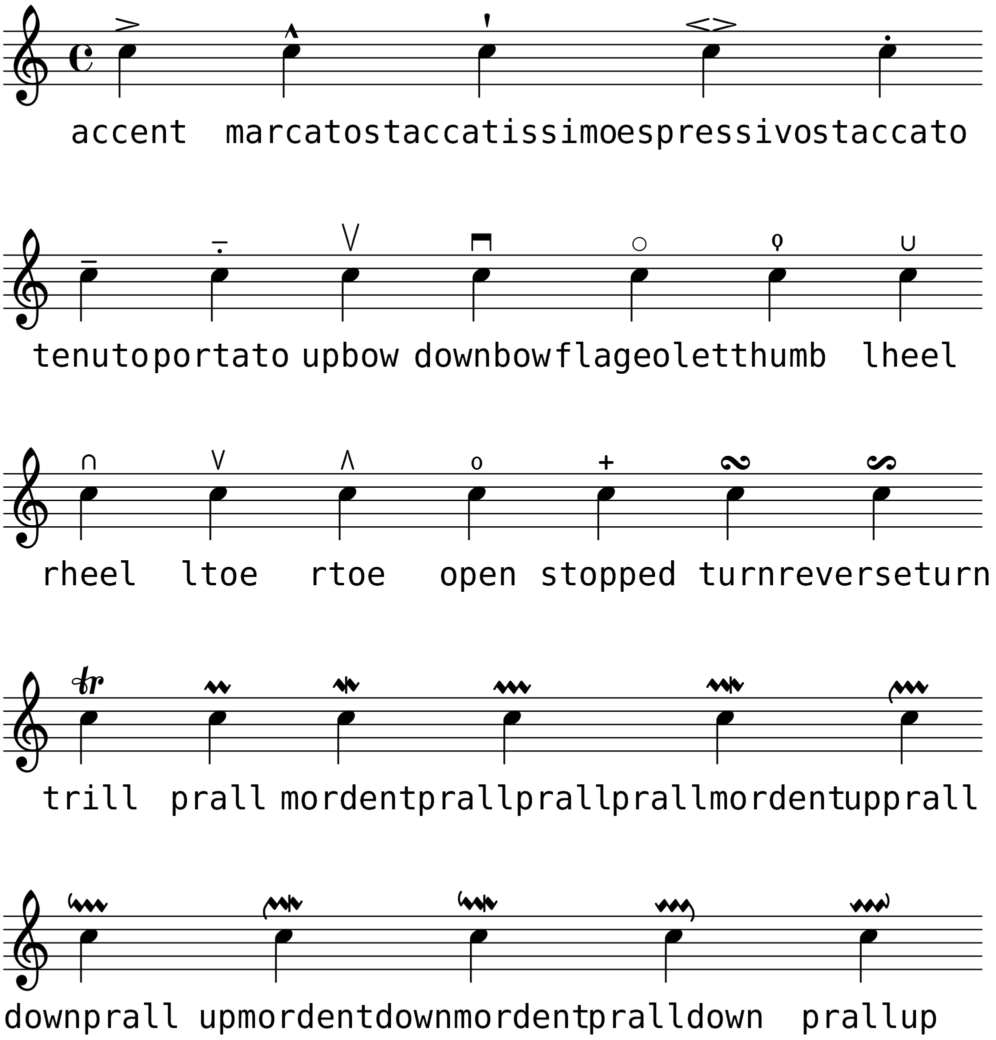
`script-abbreviations.ly`

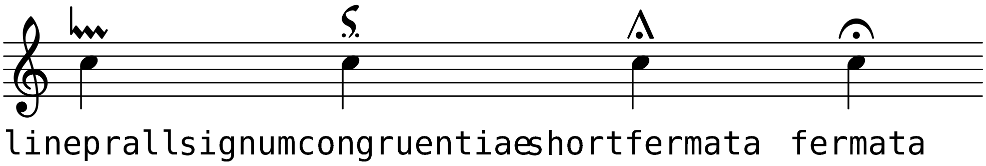
Some articulations may be entered using an abbreviation.

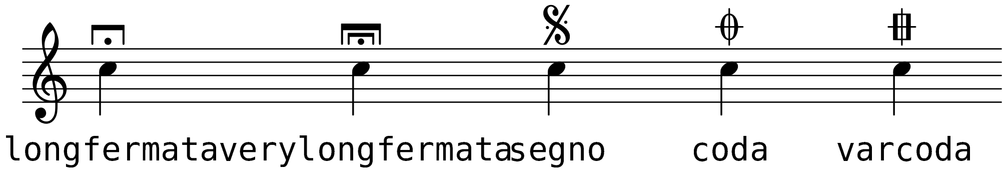


`script-chart.ly`

This chart shows all articulations, or scripts, that feta font contains.

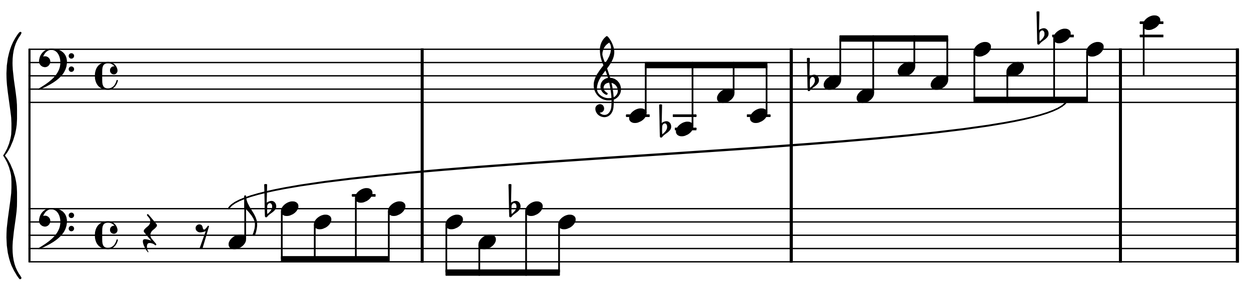






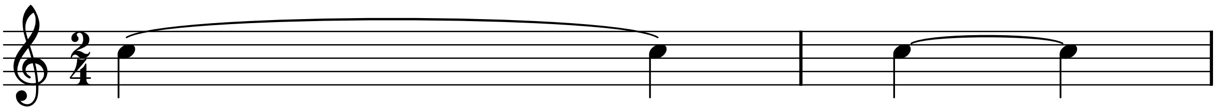
`slur-manual.ly`

In extreme cases, you can resort to setting the `control-points` of a slur manually, althout it involves a lot of trial and error. Be sure to force line breaks at both sides, since different horizontal spacing will require rearrangement of the slur.



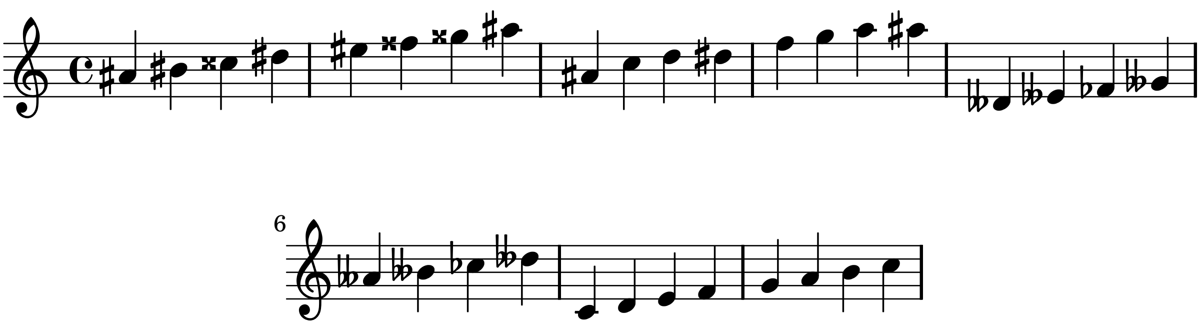
`slur-minimum-length.ly`

By setting the minimum length of a slur, notes are more separated.



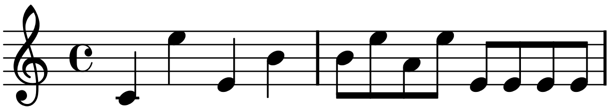
`smart-transpose.ly`

There is a way to enforce enharmonic modifications for notes in order to have the minimum number of accidentals. In that case, “Double accidentals should be removed, as well as E-sharp (-> F), bC (-> B), bF (-> E), B-sharp (-> C).”, as proposed by a request for a new feature. In this manner, the most natural enharmonic notes are chosen in this example.



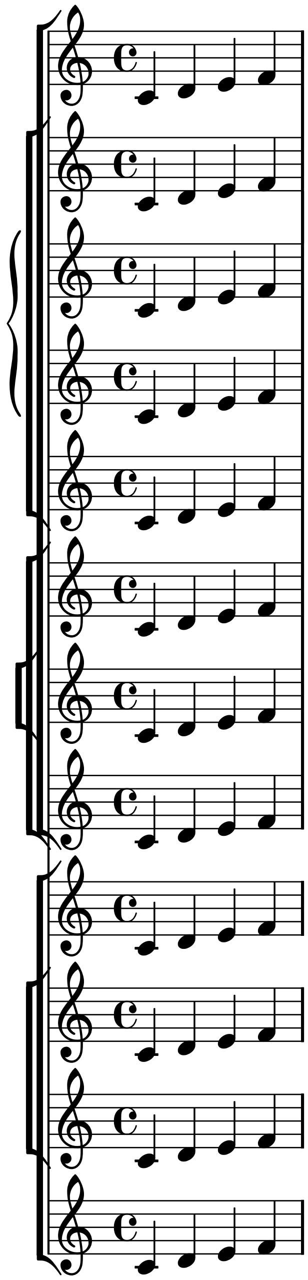
`spacing-optical.ly`

Stem directions and head positions are taken into account for spacing



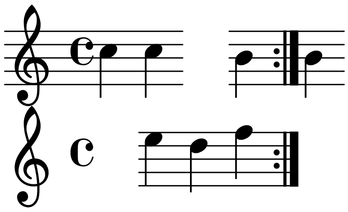
`staff-bracket.ly`

Staffs can be nested in various combinations. Here, `StaffGroup` and `ChoirStaff` produce similar straight brackets, whereas `GrandStaff` produces curly brackets. In `InnerStaffGroup` and `InnerChoirStaff`, the brackets are shifted leftwards.



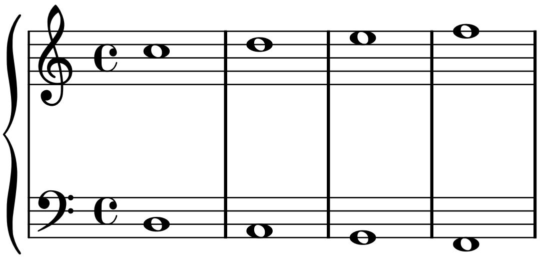
staff-container.ly

In this preliminary test of a modern score, the staff lines are washed out temporarily. This is done by making a tuned `StaffContainer`, which `\skip`s some notes without printing lines either and creates a `\new Staff` then in order to create the lines again. (Be careful if you use this; it has been done by splitting the grouping `Axis_group_engraver` and creating functionality into separate contexts, but the clefs and time signatures may not do what you would expect.)



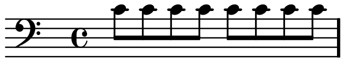
staff-lines.ly

The number of lines in a staff may changed by overriding `line-count` in the properties of `StaffSymbol`.



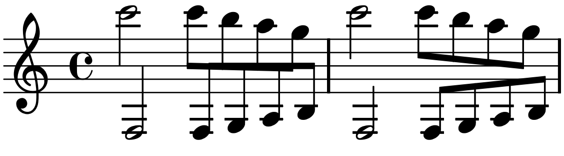
staff-size.ly

In order to change staff sizes, both `staff-space` and `fontSize` must be scaled.



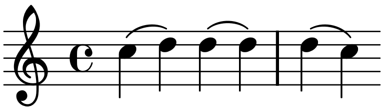
stem-extend.ly

Extending stems to the center line may be prevented using `no-stem-extend`.



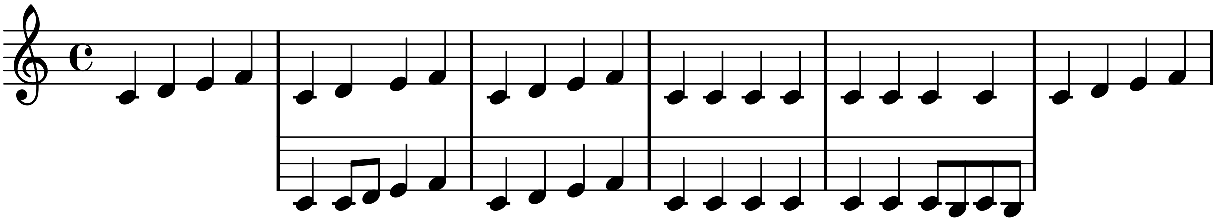
tablature-hammer.ly

A hammer in tablature can be faked with slurs.



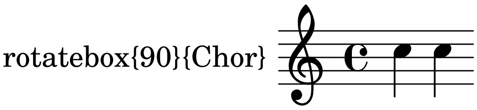
`temporary-stave.ly`

An additional stave can be typeset in the middle of a score line. A new context type is created for the temporary staff to avoid printing time and key signatures and clef at the beginning of the extra stave.



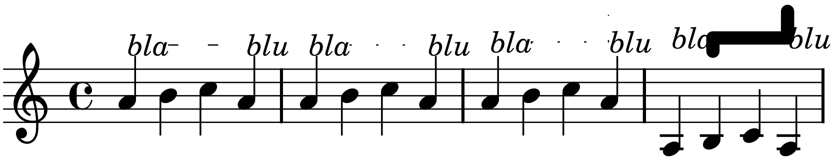
`text-rotate.ly`

Inline TeX (or PostScript) may be used, for example, to rotate text. To see the result, use the `lilypond.py` script to generate the output for printing of the source of this example (commenting one line).



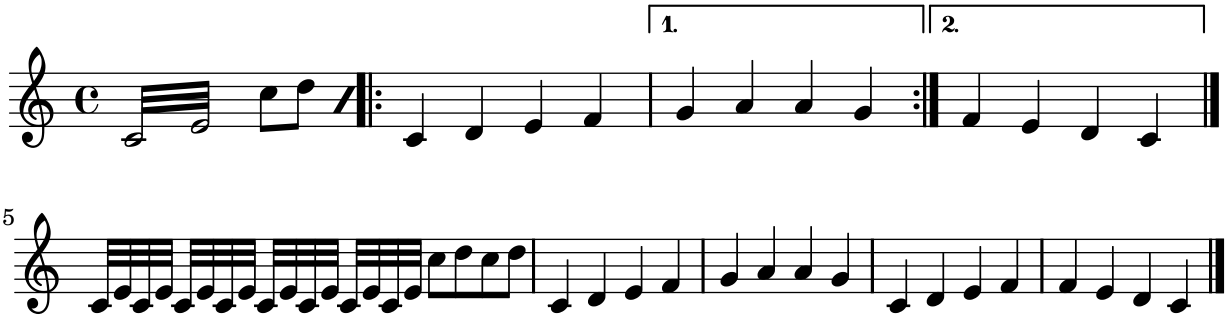
`text-spanner.ly`

Text spanners can be used in the similar manner than markings for pedals or octavation.



`unfold-all-repeats.ly`

Applying the standard function `unfold-repeats` unfolds recursively all repeats for a correct MIDI output.



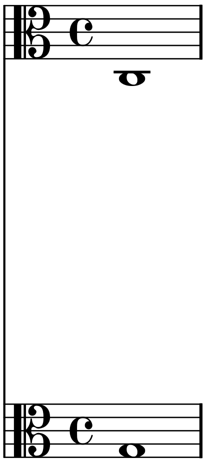
`version-output.ly`

By putting the output of `lilypond-version` into a lyric, it is possible to print the version number of LilyPond in a score, or in a document generated with `lilypond-book`. Another possibility is to append the version number to the doc-string, in this manner: 2.6.3

Processed with LilyPond version 2.6.3

`vertical-extent.ly`

Vertical extents may increased by setting `minimumVerticalExtent`, `extraVerticalExtent`, and `verticalExtent`. In this example, `verticalExtent` is increased.



`volta-chord-names.ly`

Volta brackets can be placed over chord names. Just set the `voltaOnThisStaff` property to true for the `ChordNames` context and to false for the topmost ordinary `Staff` context.