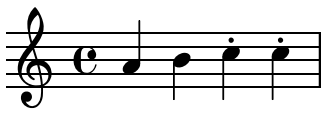
`+.ly`

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 2.2.6.

`add-staccato.ly`

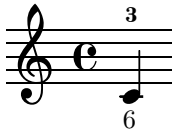
Using `make-music`, you can add various stuff to notes. In this example staccato dots are added to the notes. For this simple case, it is not necessary to use scm constructs (see `separate-staccato.ly`).



`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.

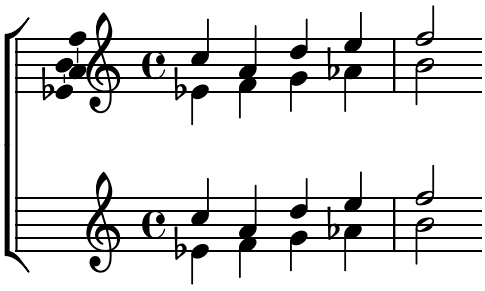
In general, first do a `display` of the music you want to create, then write a function that will structure the music for you.



`ambitus-mixed.ly`

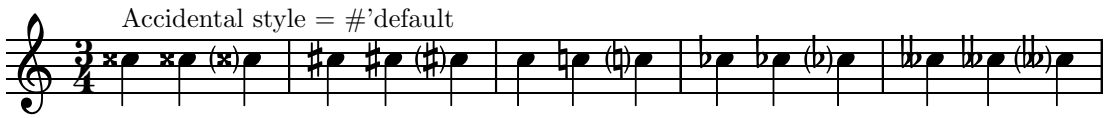
The showing of ambituses can be switched off or they can be shifted horizontally by using `applyoutput`.

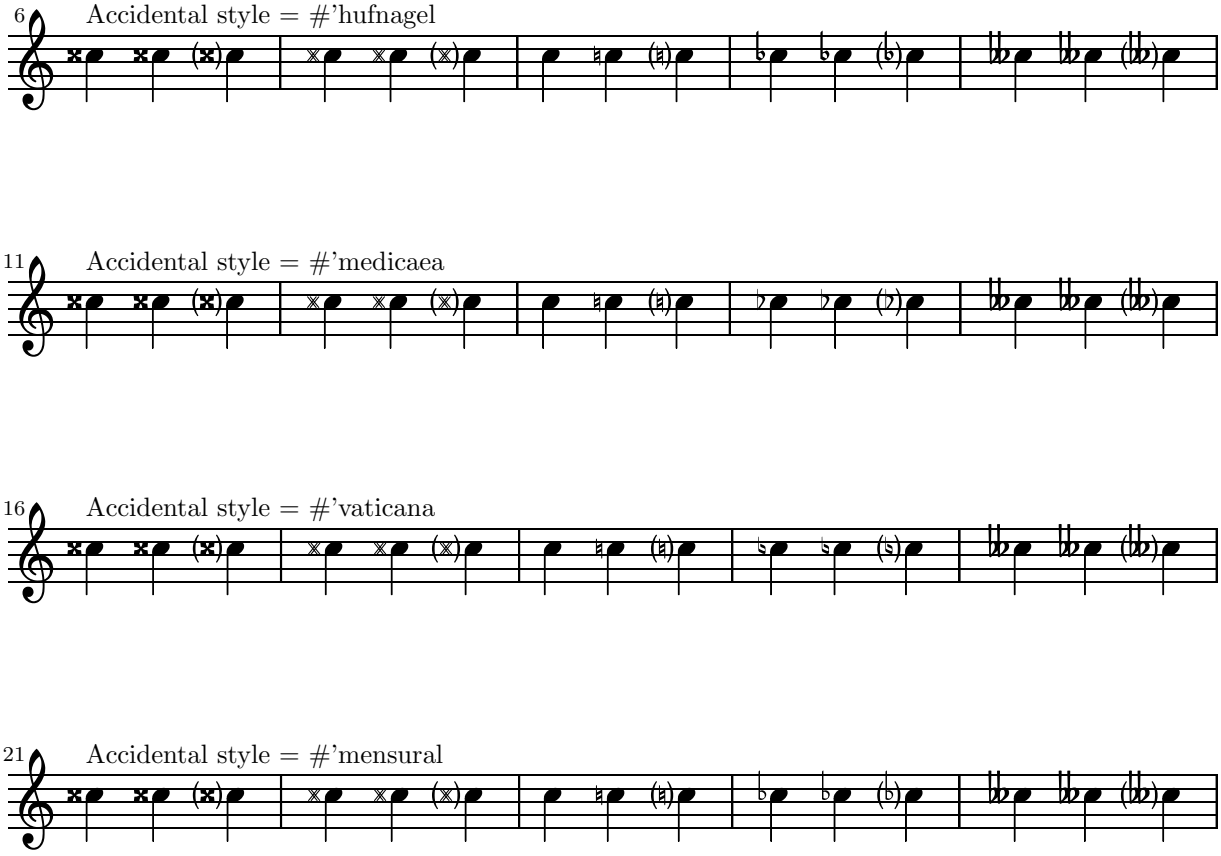
If you want to mix per-voice and per-staff ambituses, then you have to define new context type derived from the `Voice` or `Staff` context. The derived context must contain the `Ambitus_engraver` and it must be accepted by a proper parent context, which are respectively the `Staff` context or `Score` context in the example below. The original context and the derived context can then be used in parallel in the same score (not demonstrated in this file).



`ancient-accidentals.ly`

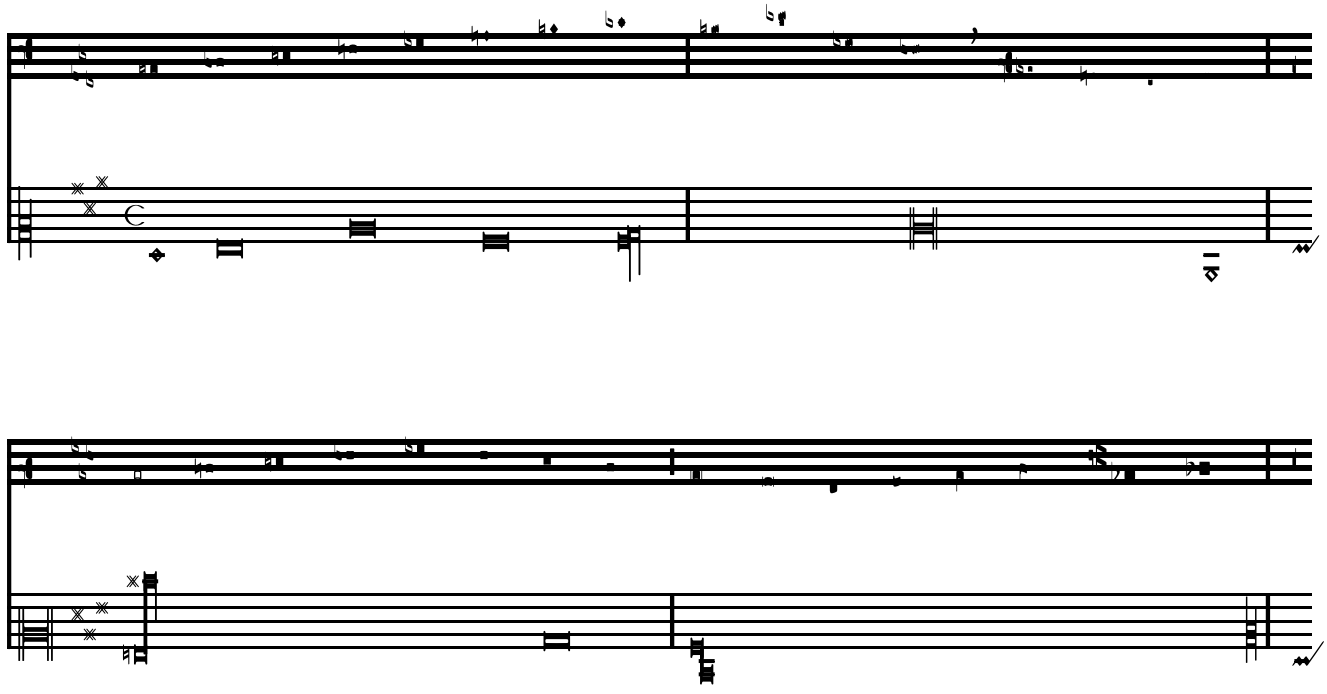
Accidentals are available in different ancient styles, which all are collected here.

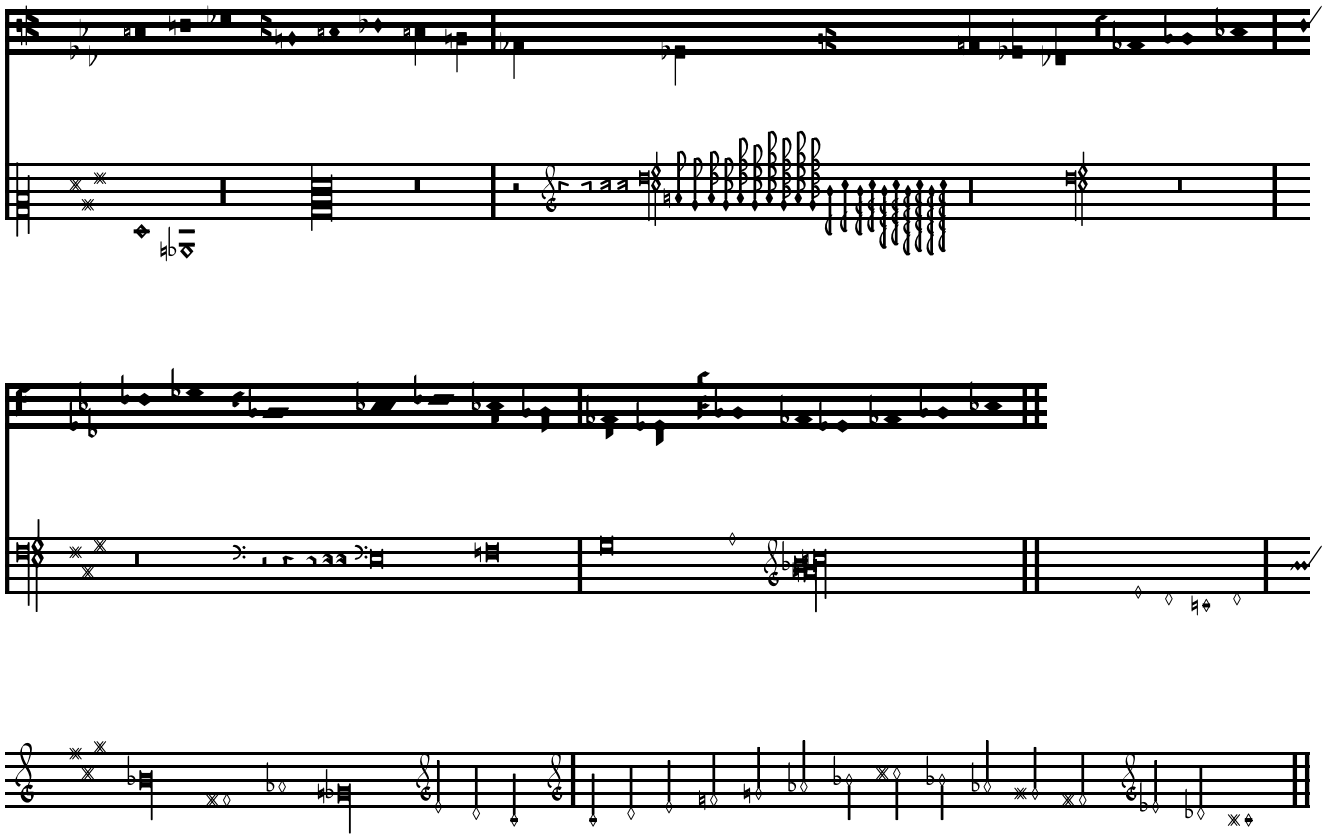




`ancient-font.ly`

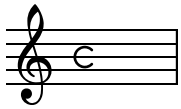
Here are shown many (all?) of the symbols that are included in LilyPond's support of ancient notation.





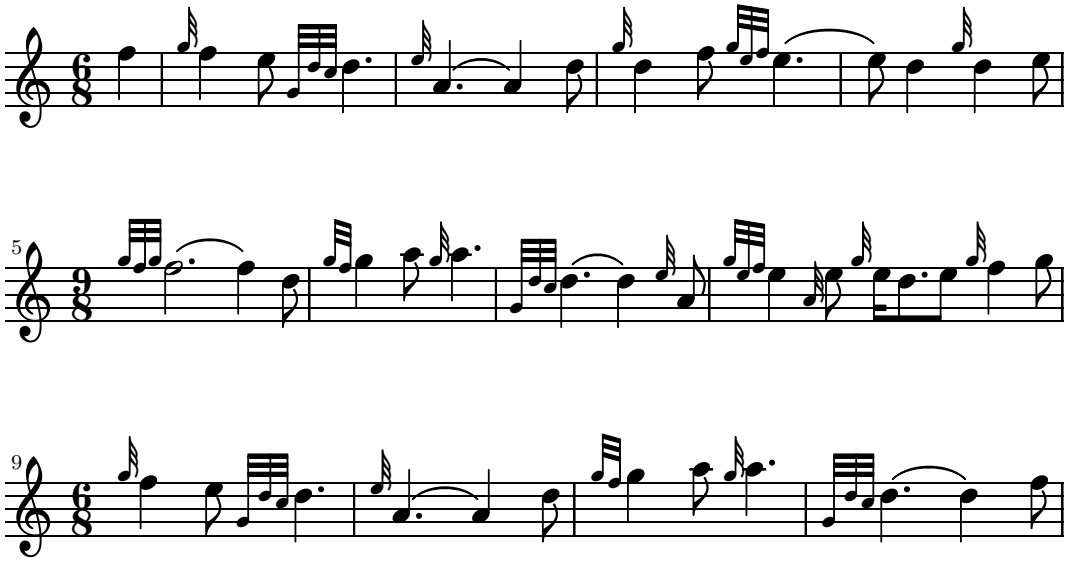
`ancient-time.ly`

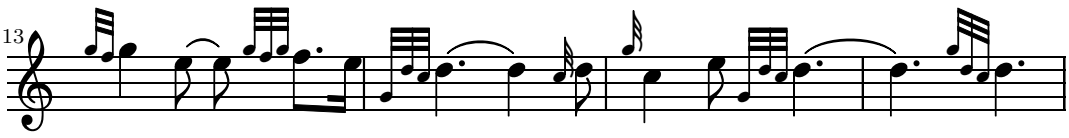
Time signatures may also be engraved in an old style.



`bagpipe.ly`

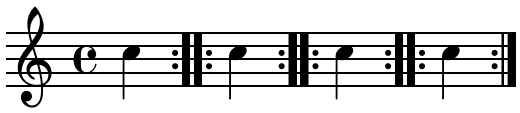
Here's an example of bagpipe music.





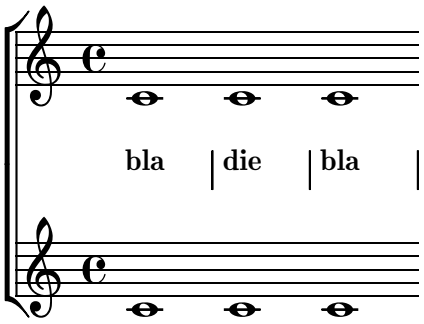
`bar-always.ly`

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



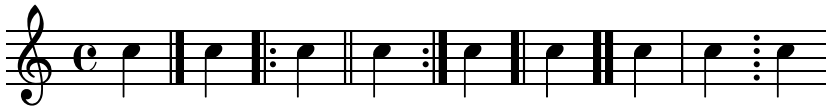
`bar-lines-lyric-only.ly`

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



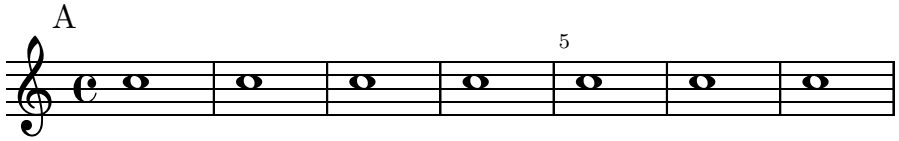
`bar-lines.ly`

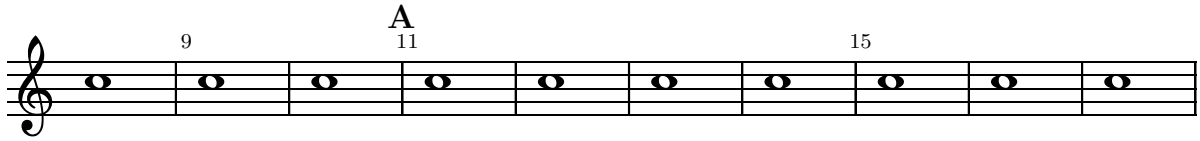
There a many types of bar lines available.

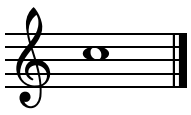


`bar-number-every-five-reset.ly`

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`.

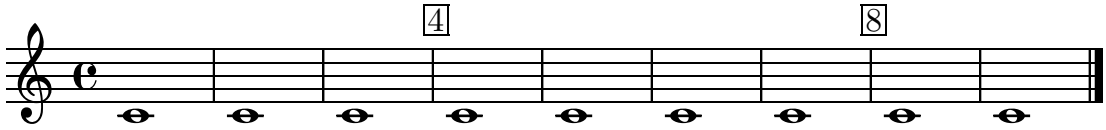






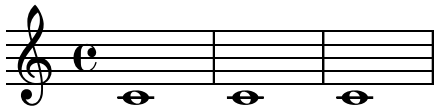
`bar-number-regular-interval.ly`

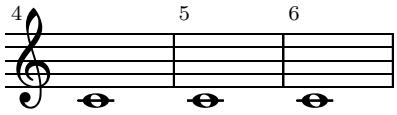
Bar numbers can also be printed inside boxes.



`bar-number-show-all.ly`

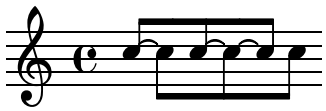
By default, bar numbers are printed only in the first measure. This setting can be overridden, so that bar numbers on start of every measure.





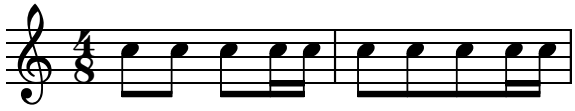
`beam-alternate.ly`

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.

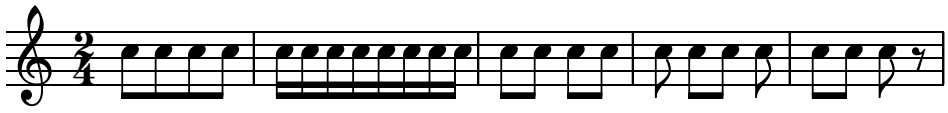


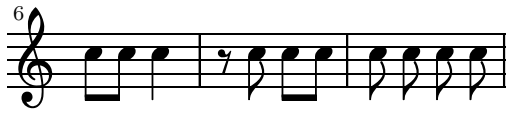
`beam-auto-override.ly`

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.

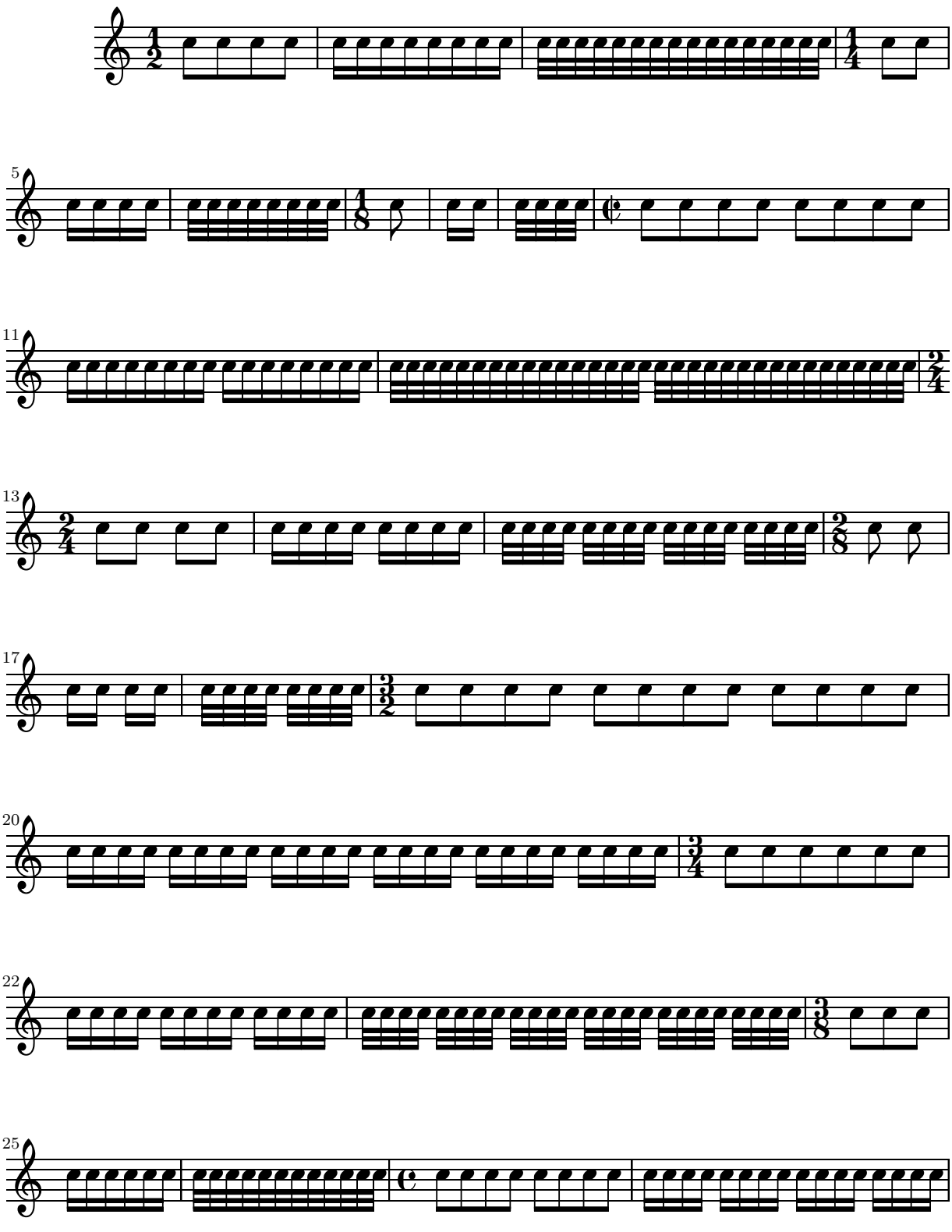
The `autoBeaming` can also be turned off.

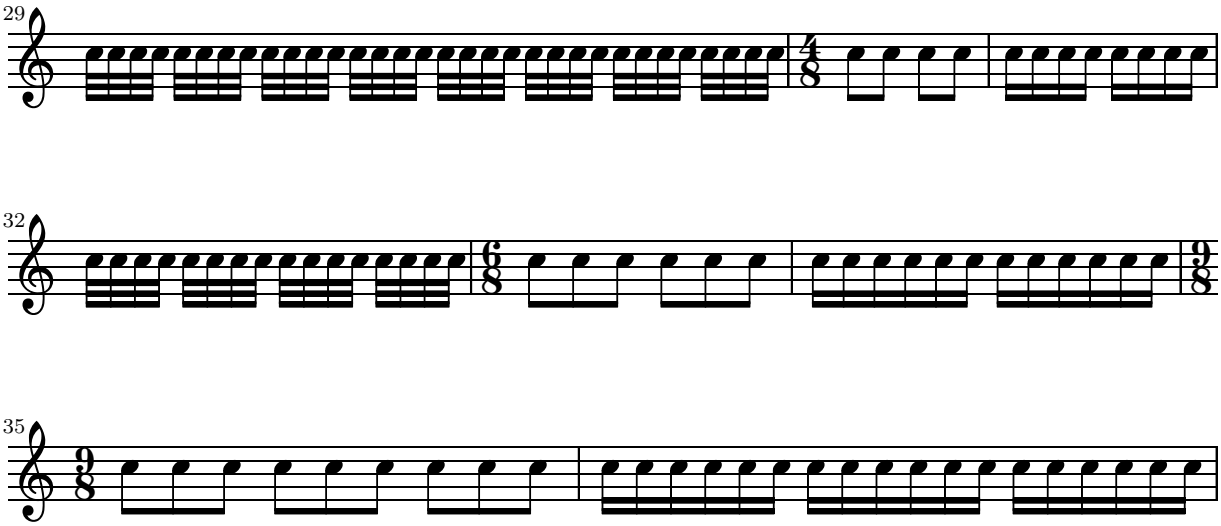




`beam-auto.ly`

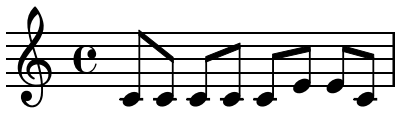
There are presets for the `auto-beam` engraver in the case of common time signatures.





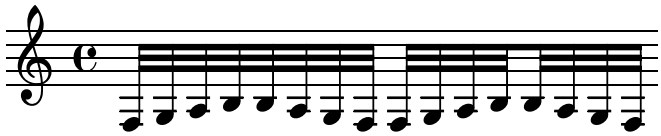
`beam-control.ly`

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.



`beam-dir-functions.ly`

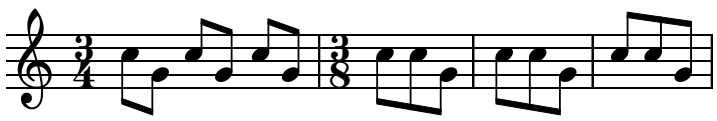
The direction of a beam may be calculated in several ways. As shown in the example, the beam are be below the notes if:

`majority` of (individual) notes would have down stems,

`mean` of note pitches is on the center line or below it, or

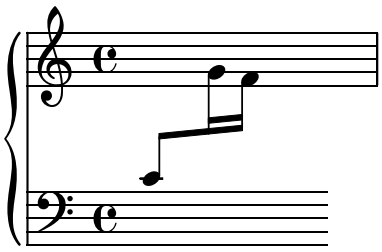
`median` of note pithes (i.e. the centermost element of ordered pitches) is on the center line or below it.

If your favourite algorithm is not one of these, you can hook up your own one. (These beam direction functions are defined in '`scm/beam.scm`'.)



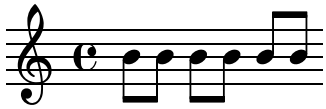
`beam-isknee.ly`

Beams can be placed across a `PianoStaff`.



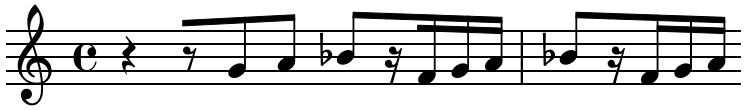
`beam-neutral-direction.ly`

When a beam falls in the middle of the staff, the beams point normally down. However, this behaviour can be altered, if desired.



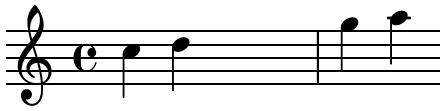
`beam-rest.ly`

Beams may be forced to be over rests.



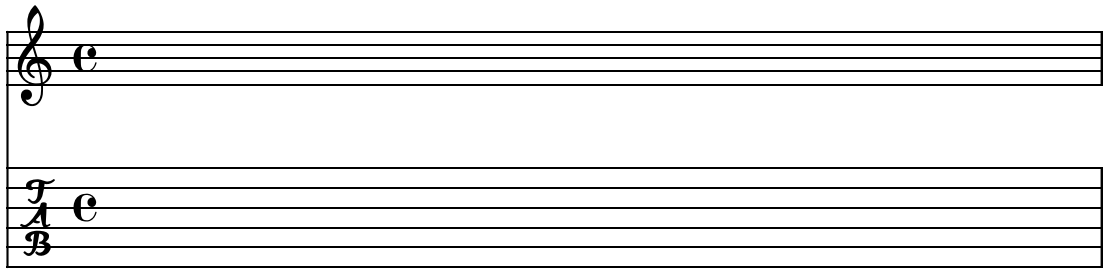
`blank-notes.ly`

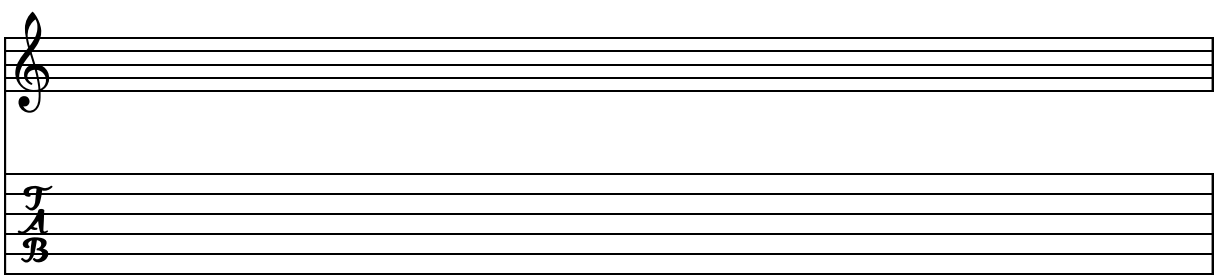
Invisible (or transparent) can be useful, when wierd tricks are needed; especially, a slur cannot be attach to a rest or spacer rest.

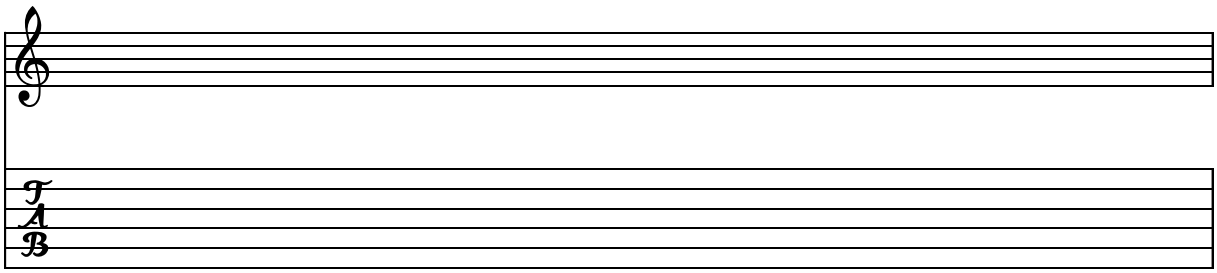


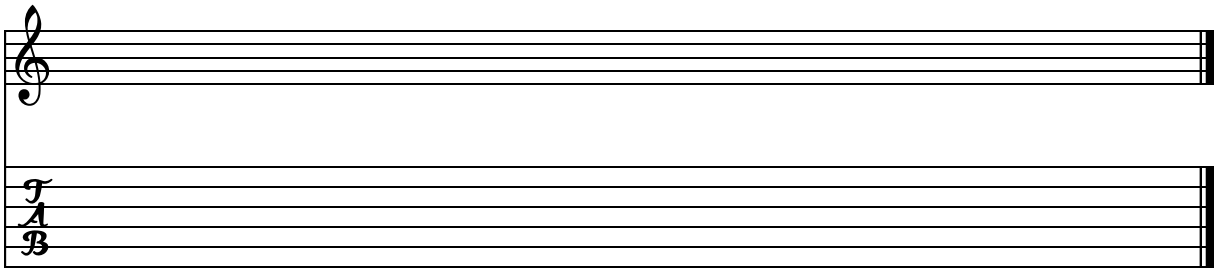
`blank-paper-tab.ly`

A blank music paper can be produced by using spacer rests, and removing `Bar_number_engraver`. Here is an empty staff with a tablature staff.



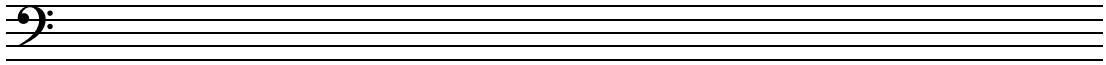


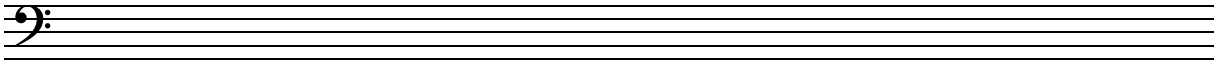


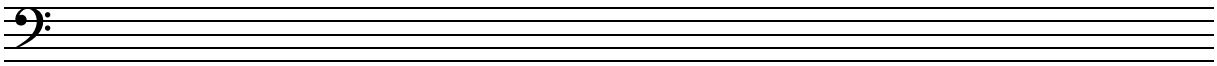


`blank-paper.ly`

A blank music paper can be produced also by using invisible notes, and removing `Bar_number_engraver`.

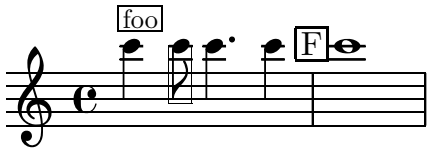






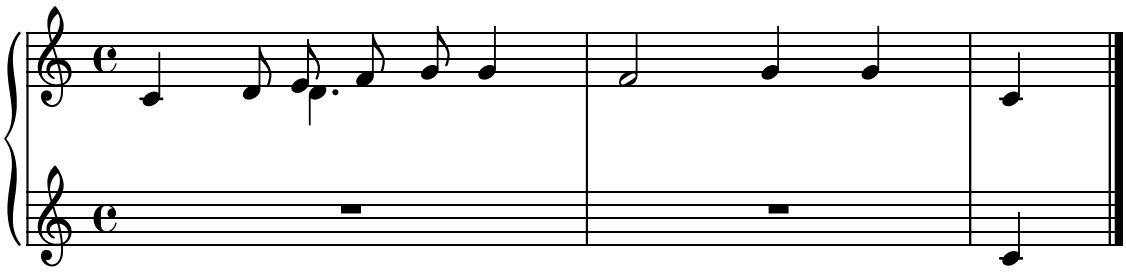
`boxed-molecule.ly`

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



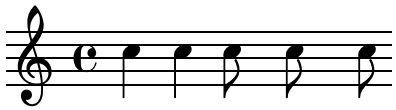
`cadenza-skip.ly`

A second staff can be aligned to a fragment (say, a cadenza) from the first staff, using a Scheme function: the function creates a skip of the same length as the cadenza.



`caps.ly`

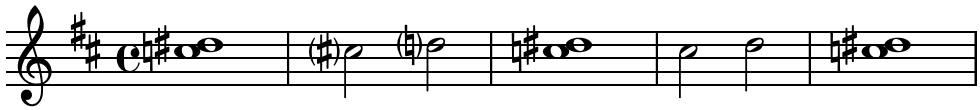
The font can be changed to small caps.

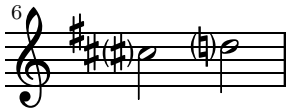


what is THE MA-TRIX?

`cautionaries.ly`

Cautinary accidentals are displayed in slurs by default. They can be shown also with accidentals of smaller size.

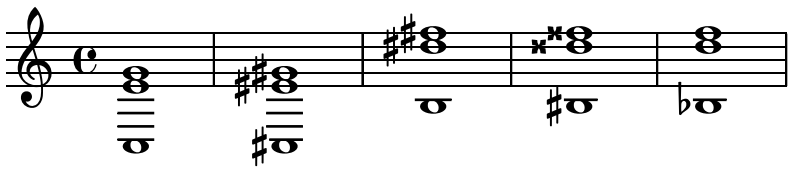




`chord-names-german.ly`

The english naming of chords (default) can be changed to german (`\germanChords` replaces B and Bes to H and B) or semi-german (`\semiGermanChords` replaces B and Bes to H and Bb).

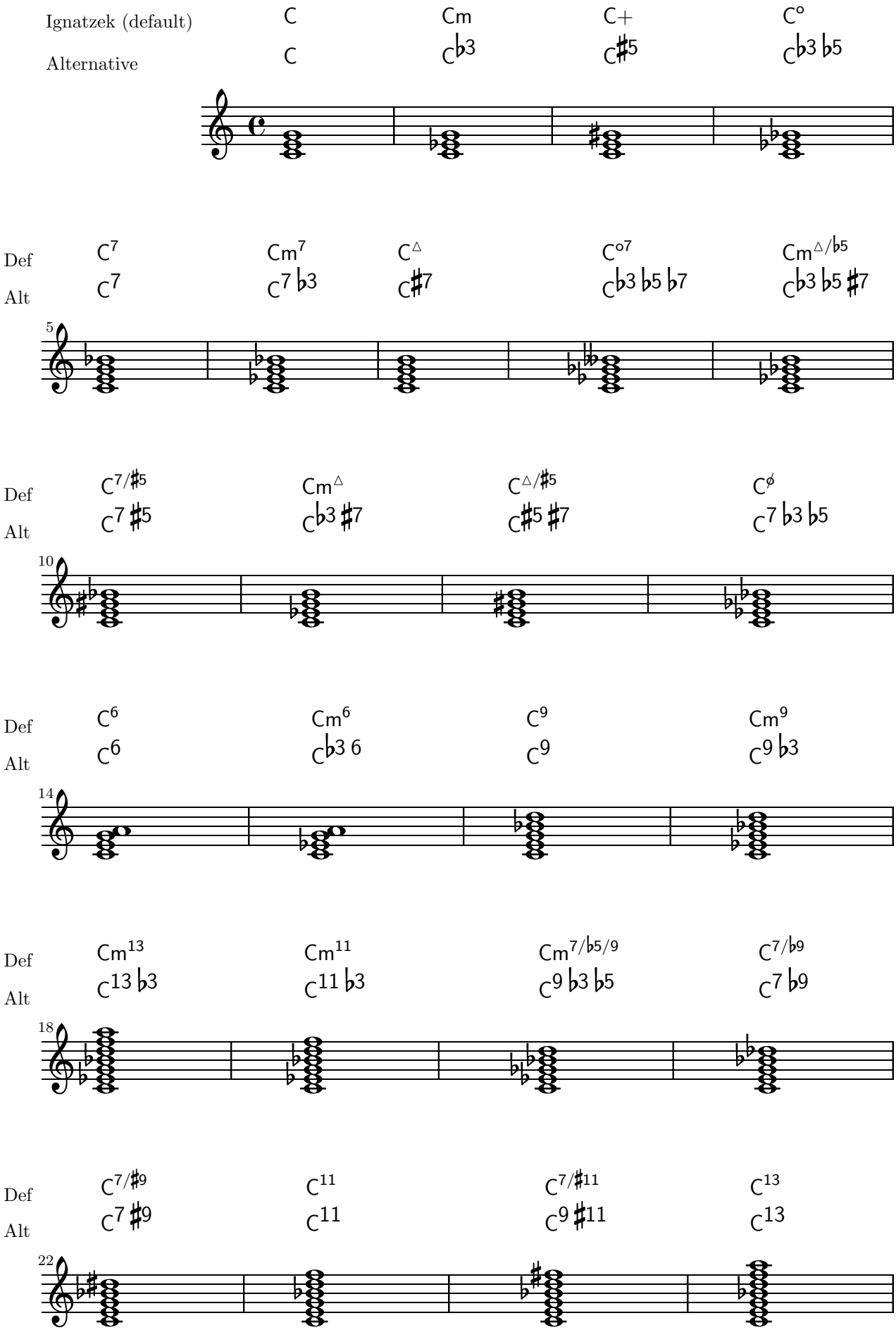
	C/C	C♯/C♯	B/B	B♯/B♯	B♭/B♭
german	C/c	C♯/cis	H/h	H♯/his	B/b
semi-german	C/c	C♯/cis	H/h	H♯/his	B♭/b

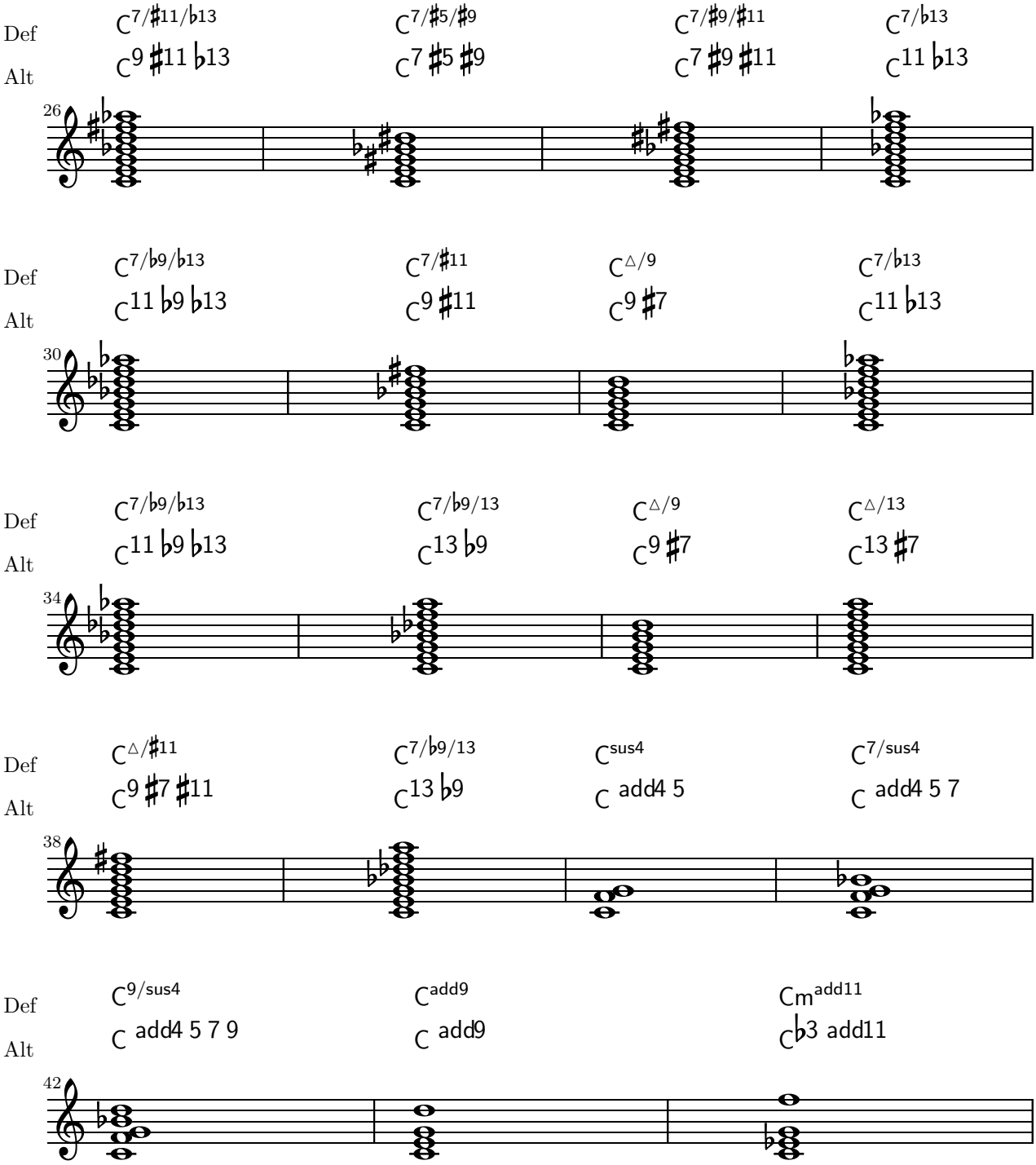


`chord-names-jazz.ly`

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.

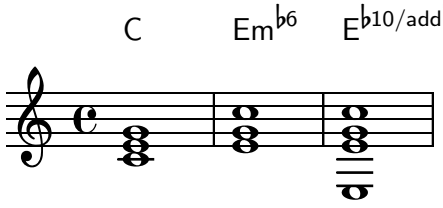
Chords following Banter (1987) can also be printed from this file, but are turned off for brevity.


Ignatzek (default)
C
Cm
C+
C°
Alternative
C
C♭3
C♯5
C♭3 ♭5
Def
Alt
C7
C7
Cm7
C7 ♭3
C△
C♯7
C°7
C♭3 ♭5 ♭7
Cm△/♭5
C♭3 ♭5 ♯7
5
Def
Alt
C7/♯5
C7 ♯5
Cm△
C♭3 ♯7
C△/♯5
C♯5 ♯7
Cø
C7 ♭3 ♭5
10
Def
Alt
C6
C6
Cm6
C♭3 6
C9
C9
Cm9
C9 ♭3
14
Def
Alt
Cm13
C13 ♭3
Cm11
C11 ♭3
Cm7/♭5/9
C9 ♭3 ♭5
C7/♭9
C7 ♭9
18
Def
Alt
C7/♯9
C7 ♯9
C11
C11
C7/♯11
C9 ♯11
C13
C13
22



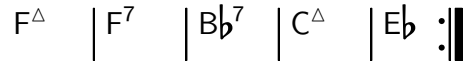
`chord-names-no-inversions.ly`

Since there are several interpretations for recognizing chord names, the lowest note is the bass note of a chord and the inversion of the chord is found accordingly.



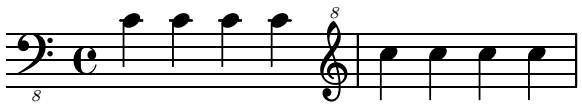
`chords-without-melody.ly`

Jazz chord names can also be printed without notes.



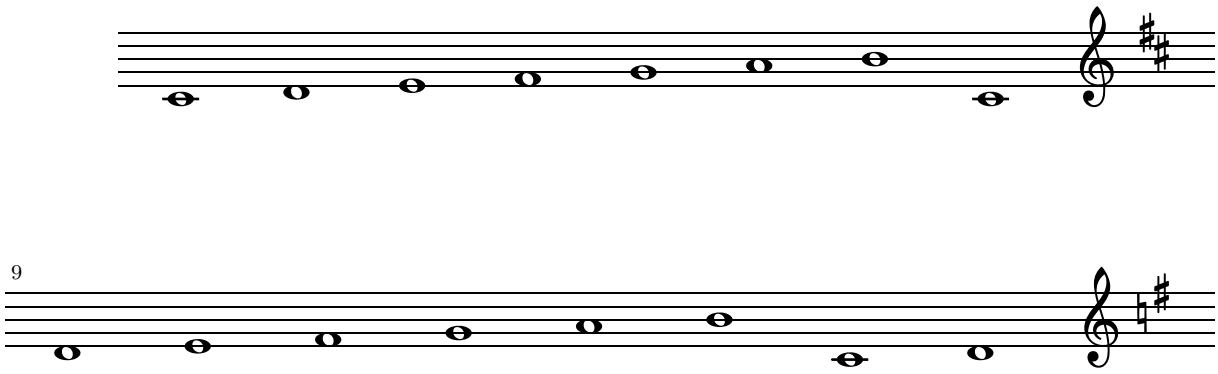
`clef-8-syntax.ly`

Appending `_8` or `^8` to a clef name will add an octavation sign to the clef; then the clef name is given in quotes (such as `"treble^8"`).



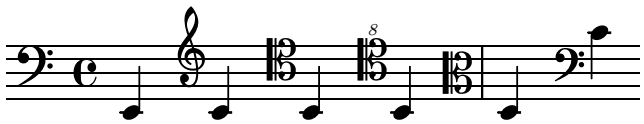
`clef-end-of-line.ly`

In these scales, the clef and key signature are shown at the end of the line.



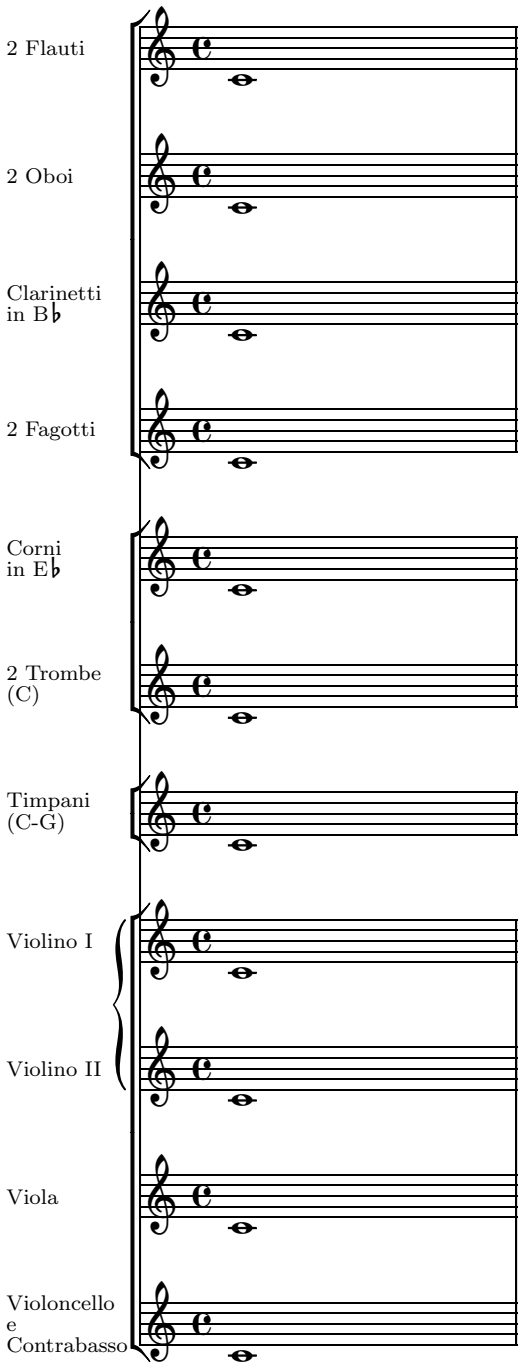
`clef-manual-control.ly`

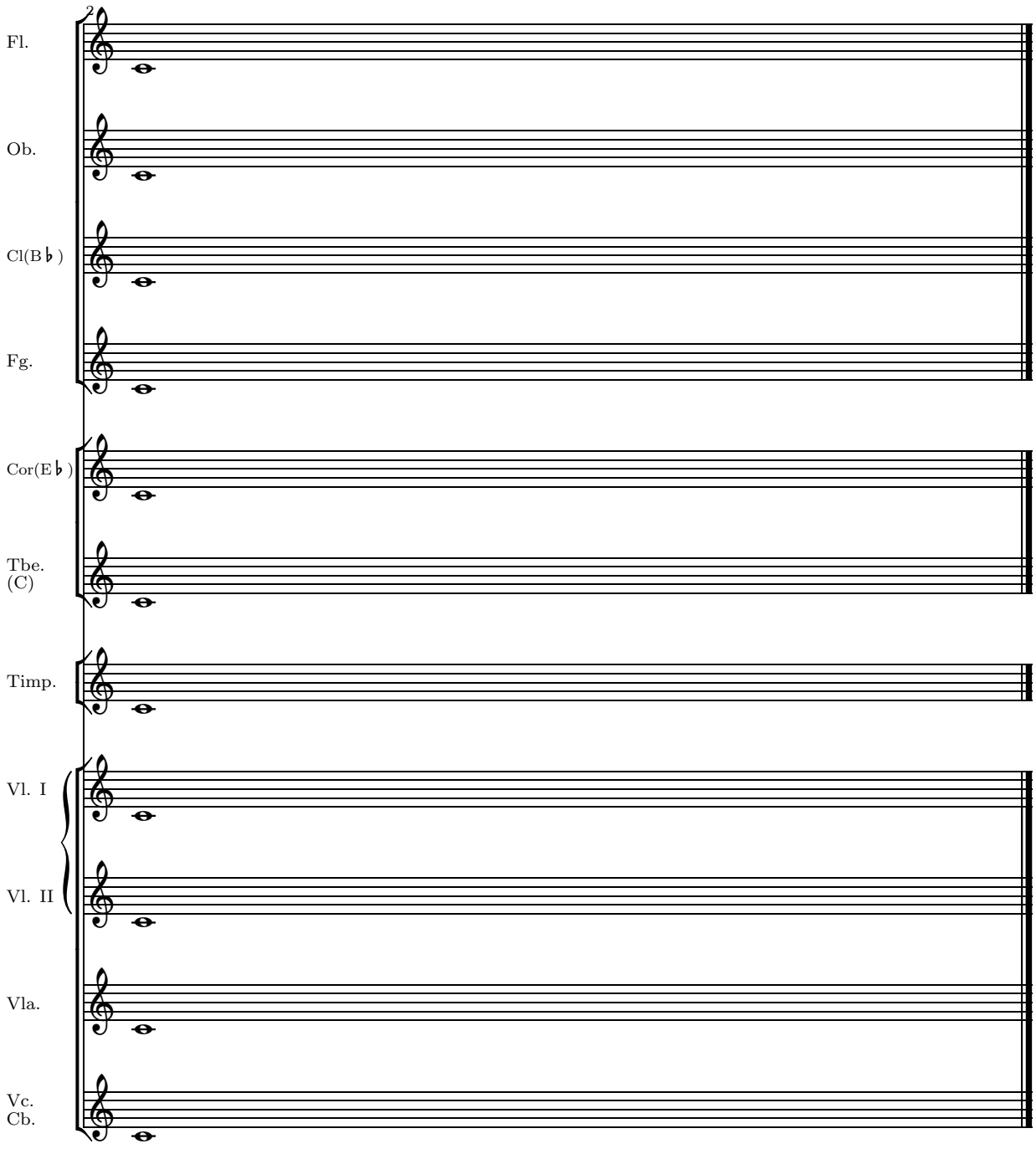
The positioning of glyph and note can be separated. `\clef` is a front-end, which keeps them together. All the notes in this example are central C.



`coriolan-margin.ly`

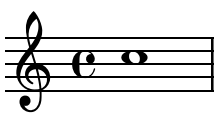
In an orchestral score (Beethoven's Coriolan overture), there are different instrument groups, and some of the instruments may be transposed. Instruments are indicated either with a long or short name.

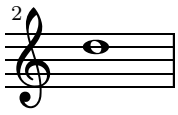

2 Flauti
2 Oboi
Clarinetti
in B♭
2 Fagotti
Corni
in E♭
2 Trombe
(C)
Timpani
(C-G)
Violino I
Violino II
Viola
Violoncello
e
Contrabasso



`count-systems.ly`

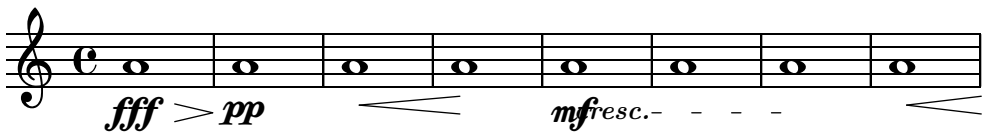
After a line break, some function may called by overriding `after-line-breaking-callback`. This can be most useful to assertain that a piece uses a specified number of lines; typically the number of lines (or systems) is not engraved, but it can be printed to console when generating the output. The number of lines may be associated either to the number of systems or the system number of a grob.

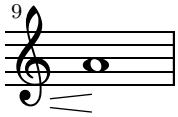




`crescendi.ly`

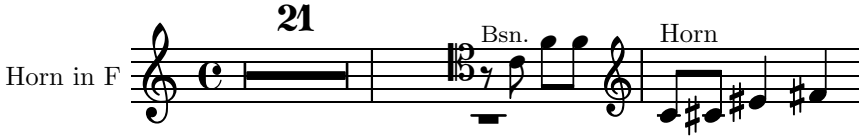
Crescendi can be printed in a number of different ways.





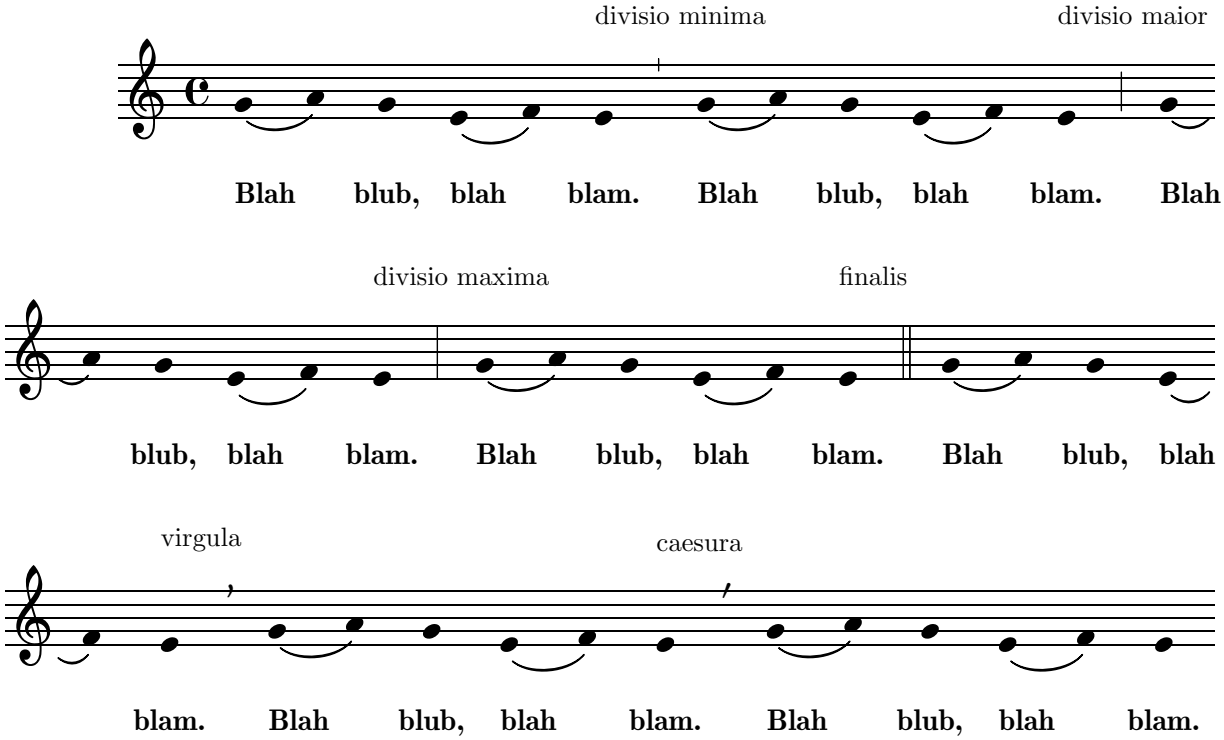
`cue-notes.ly`

Cue notes are typeset in a smaller font.



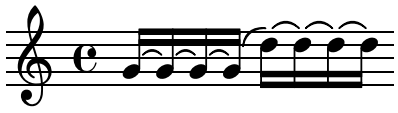
`divisiones.ly`

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



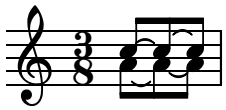
`drarn-slurs.ly`

Slurs can be forced to always attach to note heads.



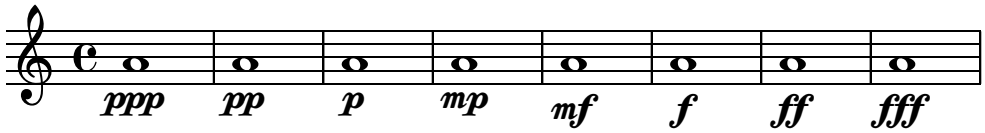
`drarn.ly`

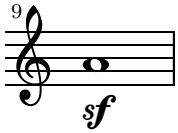
You can attach slurs and ties to noteheads.



`dynamic-absolute-volume.ly`

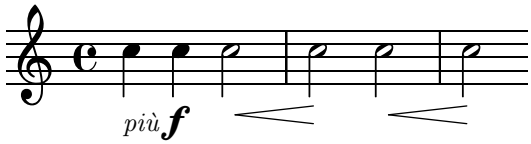
Absolute dynamics have an effect on MIDI files.





`dynamic-extra.ly`

Pi forte dynamics is produced using `\markup`.

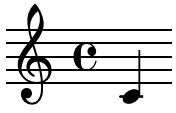


`embedded-postscript.ly`

By inserting the TEX command \embeddedps, you can insert postscript directly into the output.

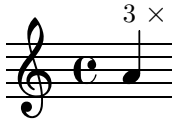
`embedded-scm.ly`

You can embed scheme functions in your scores. While generating the output, “hello world” is printed to the console.



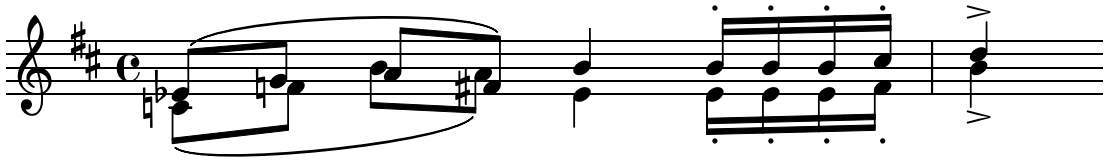
`embedded-tex.ly`

You can embed Tex commands in your score.



`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`

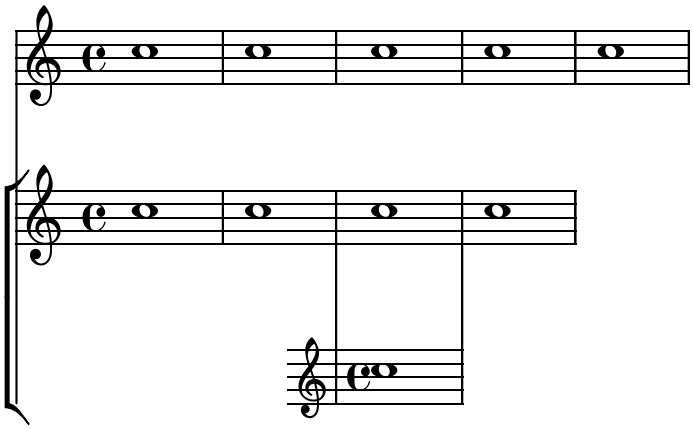
The notation problem, creating a certain symbol, is handled by plugins. Each plugin is called Engraver. In this example, engravers is switched on one by one, in the following order:

- 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.

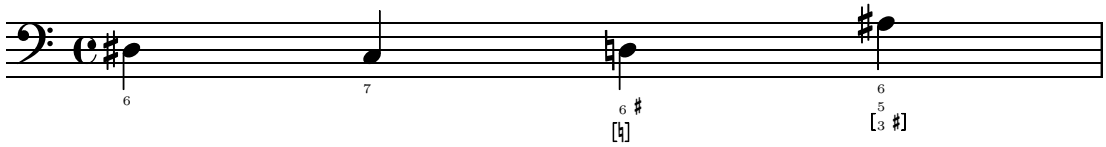
`extra-staff.ly`

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



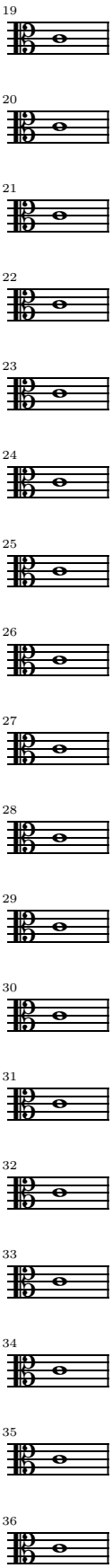
`figured-bass-alternate.ly`

An alternate method to make bass figures is to use `markup` texts.



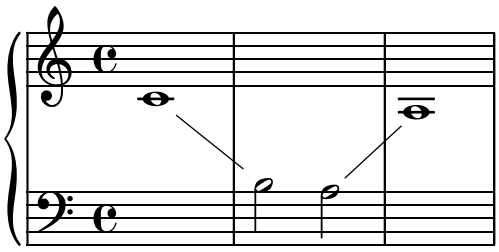
`fill-a4.ly`

This should fill a4 paper.



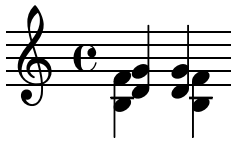
`follow-voice.ly`

Voices can be traced automatically when they switch staves by setting `followVoice`.



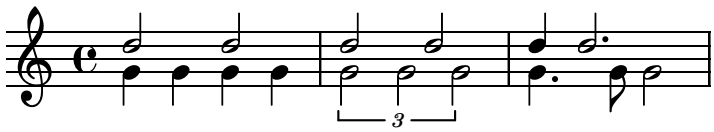
`force-hshift.ly`

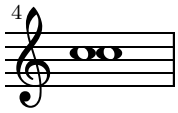
Horizontal shift (`hshift`) can be forced in order to avoid collisions.



`gourlay.ly`

The breaking of line works also with polyphony. This is taken from Gourlay's paper on breaking lines.

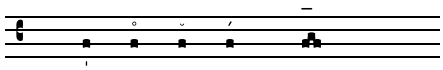




`gregorian-scripts.ly`

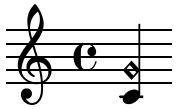
Here is demonstrated a preliminary support of Gregorian Scripts:

ictus, circulus, semicirculus, accentus, episem.



`harmonic.ly`

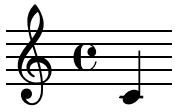
Artificial harmonics are notated with a different notehead style, by marking the harmonic pitch with `\harmonic`.



`header-ifelse.ly`

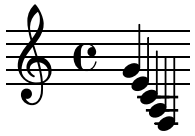
High level functionality (eg. conditional defines), can be accomplished with GUILE.

This example puts the current version in the tagline via Scheme, however, the tagline is not printed to the collated webpage snippets.



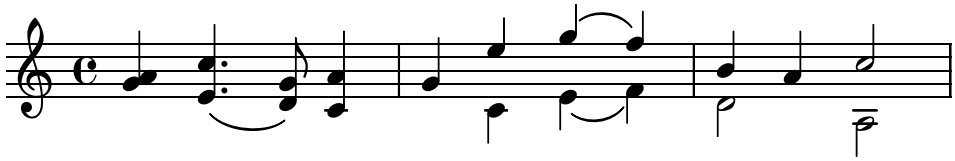
`hshift.ly`

Notes may be manually horizontally shifted.



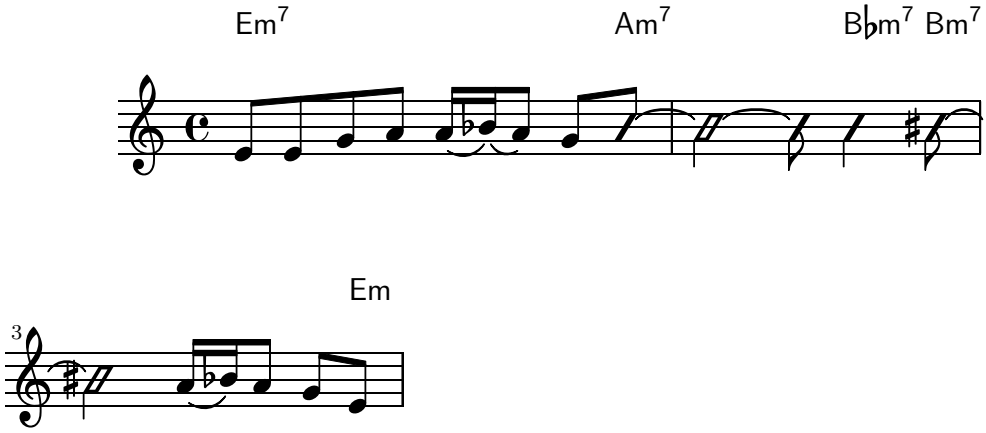
`hymn.ly`

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.



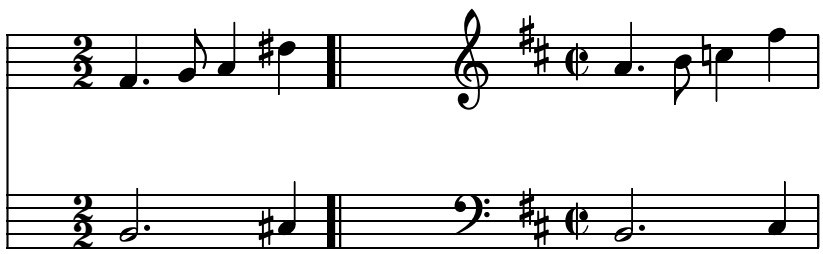
`improv.ly`

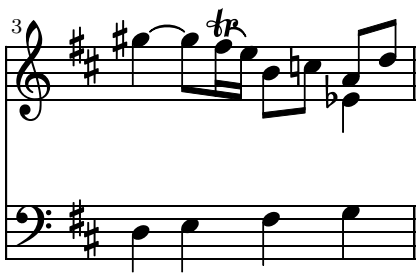
In improvisation, noteheads do not have a pitch, and have different shapes. In this example, this is achieved by adding `Pitch_squash_engraver` and setting `squashedPosition` when the improvisation is active.



`incipit.ly`

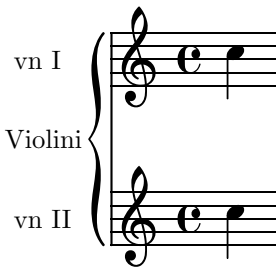
This example shows how to make an "incipit" to indicate scordatora tuning of a violin part, by overriding the `style` of a `TimeSignature`. Here are the two first bars of Biber's Rosary sonata III.





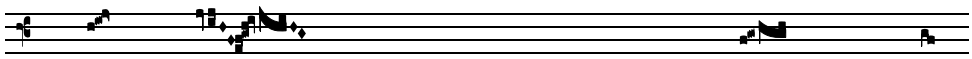
`instrument-name-grandstaff.ly`

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



`ligature-vaticana.ly`

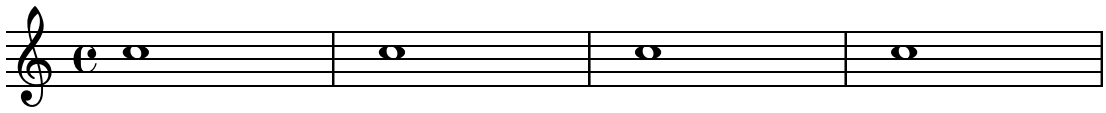
Vaticana ligature uses four staff lines, special clef, an notes calligraphic notes.



Al- le- lu- ia.

`lilypond-testpage.ly`

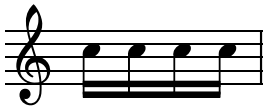
In the generated output for printing, there are several titles which do not appear in the web pages.



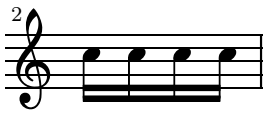
`lyric-hyphen-retain.ly`

In tightly engraved music, hyphens are removed, except at the end of the line. Normally, lyrics are not typeset so tightly, but by tuning down `padding` of in `SeparationItem`, syllables are put closer together, and as a result hyphens may disappear.

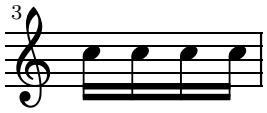
In some languages (e.g. German and Hungarian), hyphens should not disappear, since spelling depends on hyphenation. For that purpose, hyphens can be forced to remain by overriding `minimum-length` of the `LyricHyphen` grob.



bla bla bla bla-



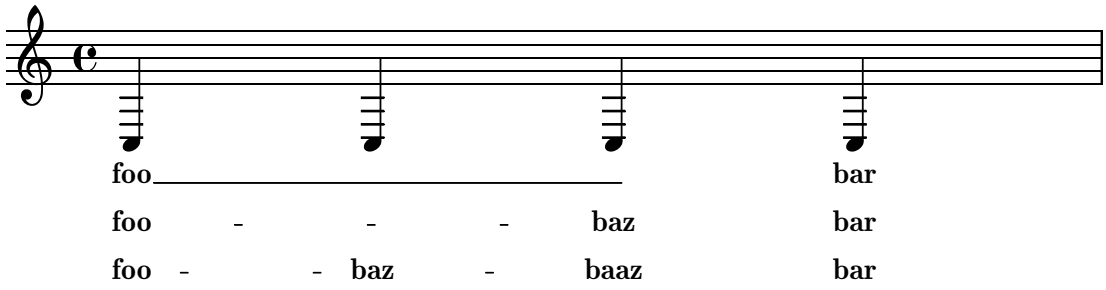
bla bla bla bla-



bla-bla-bla-bla

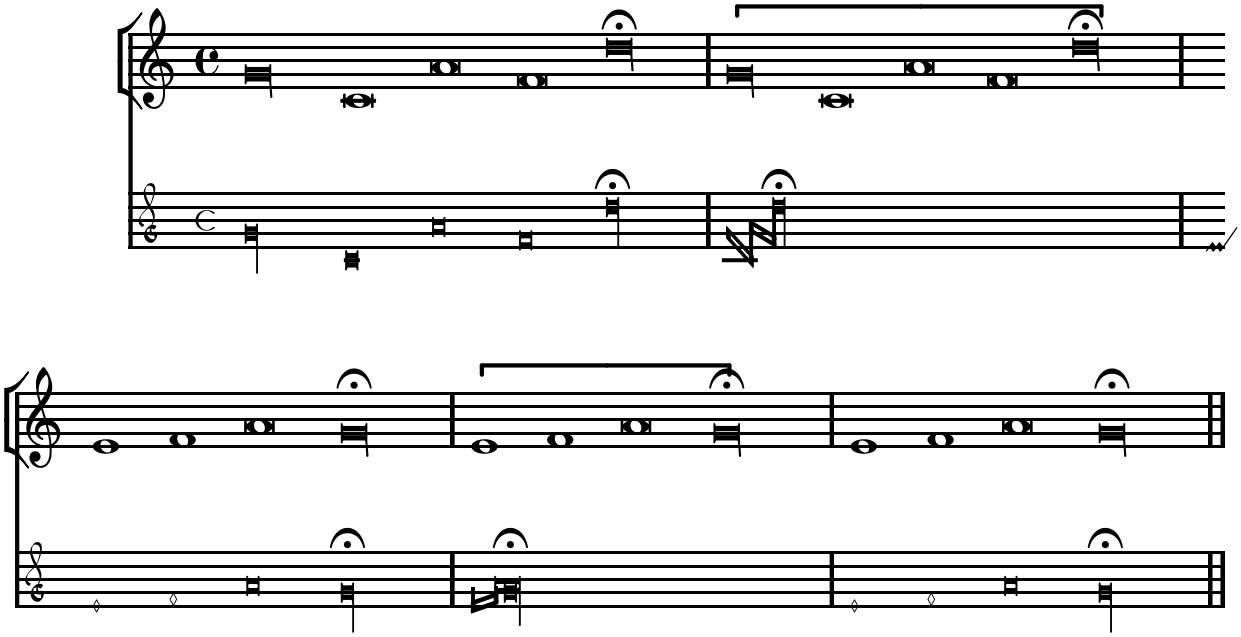
`lyrics-skip-notes.ly`

By inserting `\skip` statements into lyric lines, one can attach less lyric syllables to a melody.



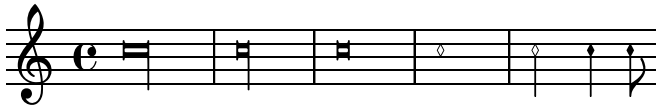
`mensural-ligatures.ly`

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



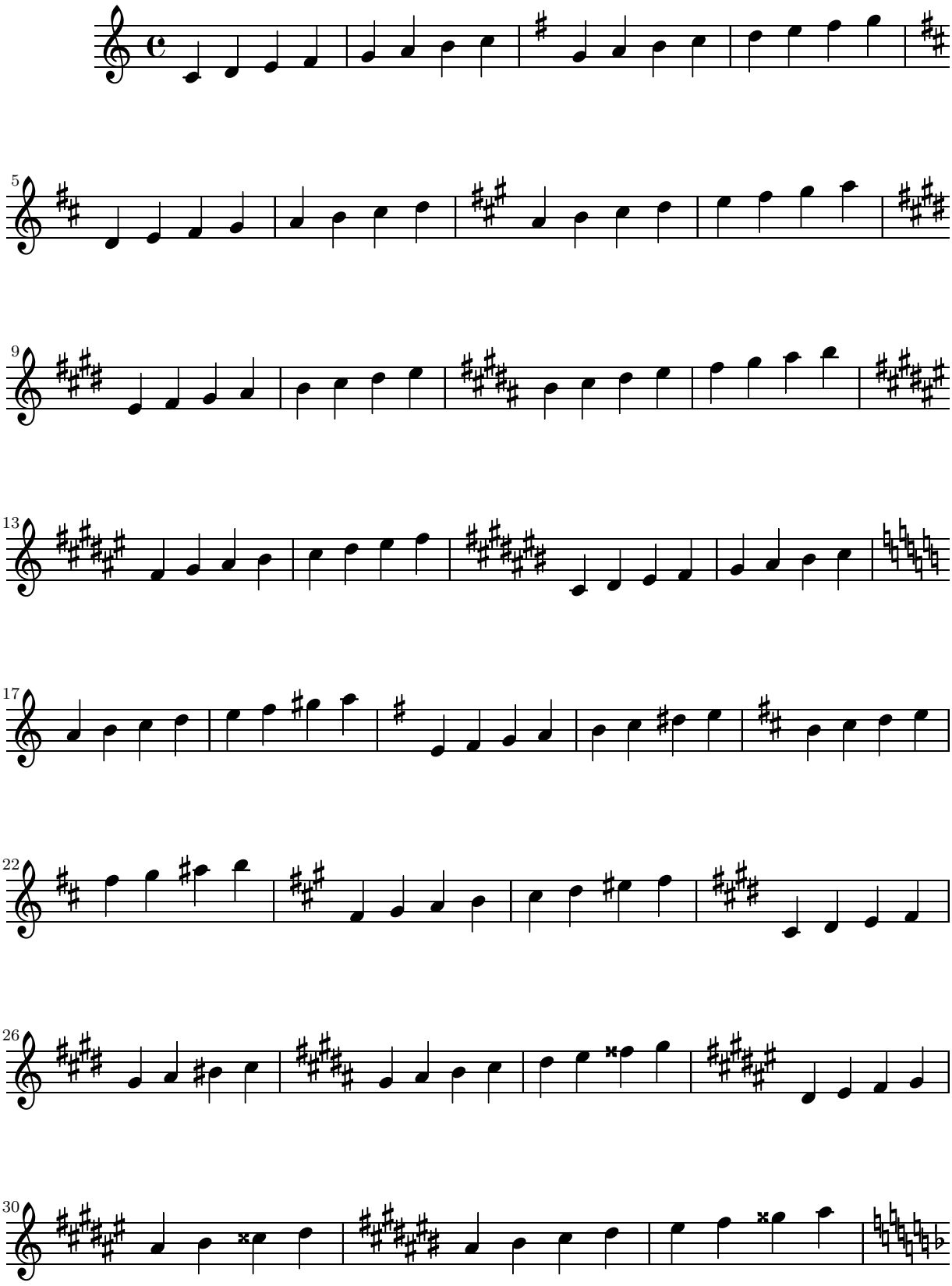
`mensural-note-heads.ly`

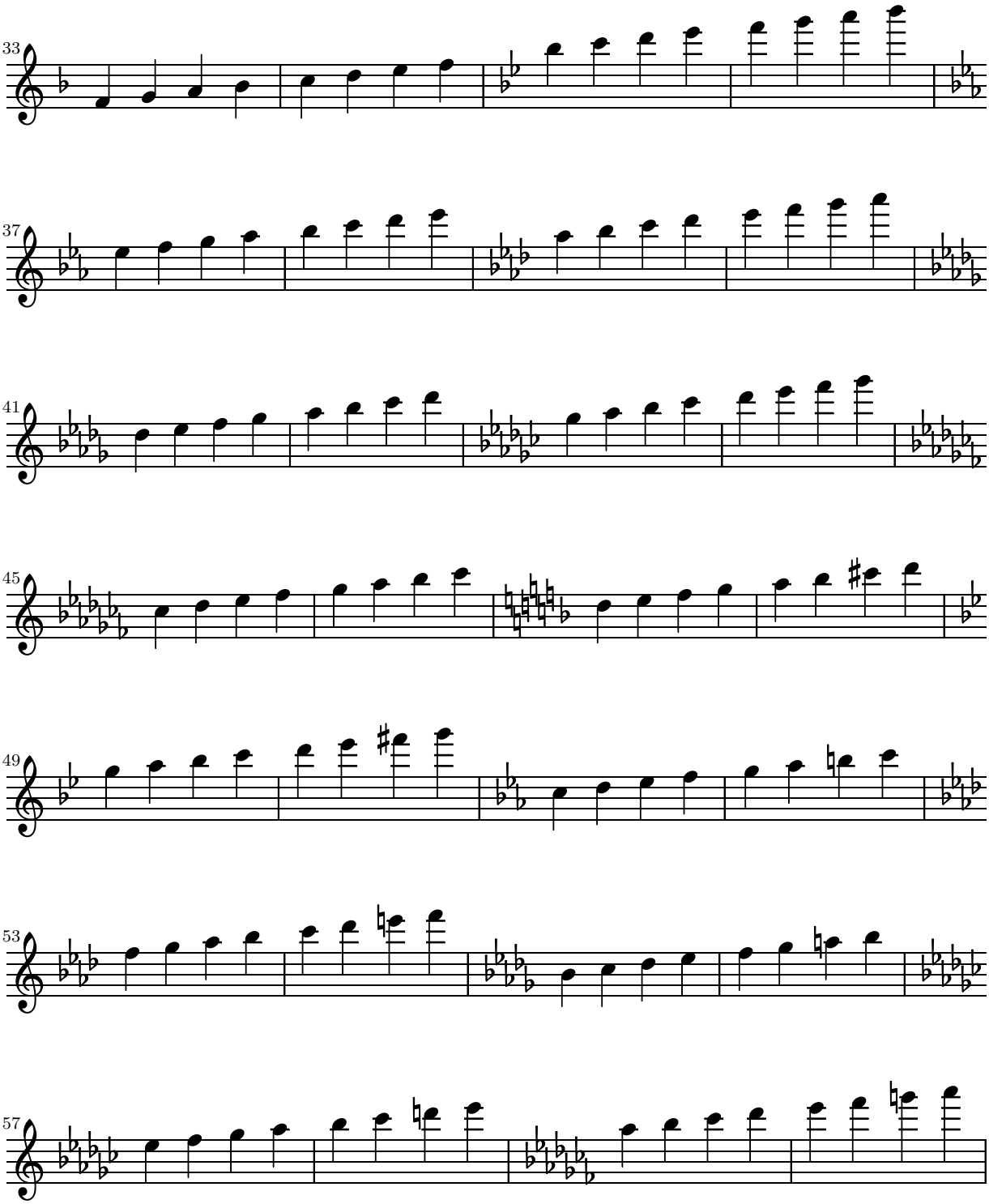
Mensural notes may also have note heads.



`midi-scales.ly`

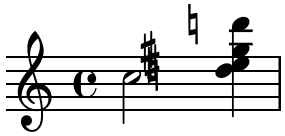
Converting LilyPond input to MIDI and then again back with `midi2ly.py` is a reversible procedure in some simple cases, which mean that the original `.ly` -file and the one converted back from the generated `.midi` -file do not differ. Here are produced some scales.





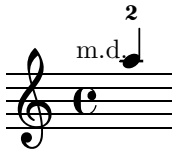
`move-accidentals.ly`

The positions of accidentals may be manually set by incorporating some Scheme code.



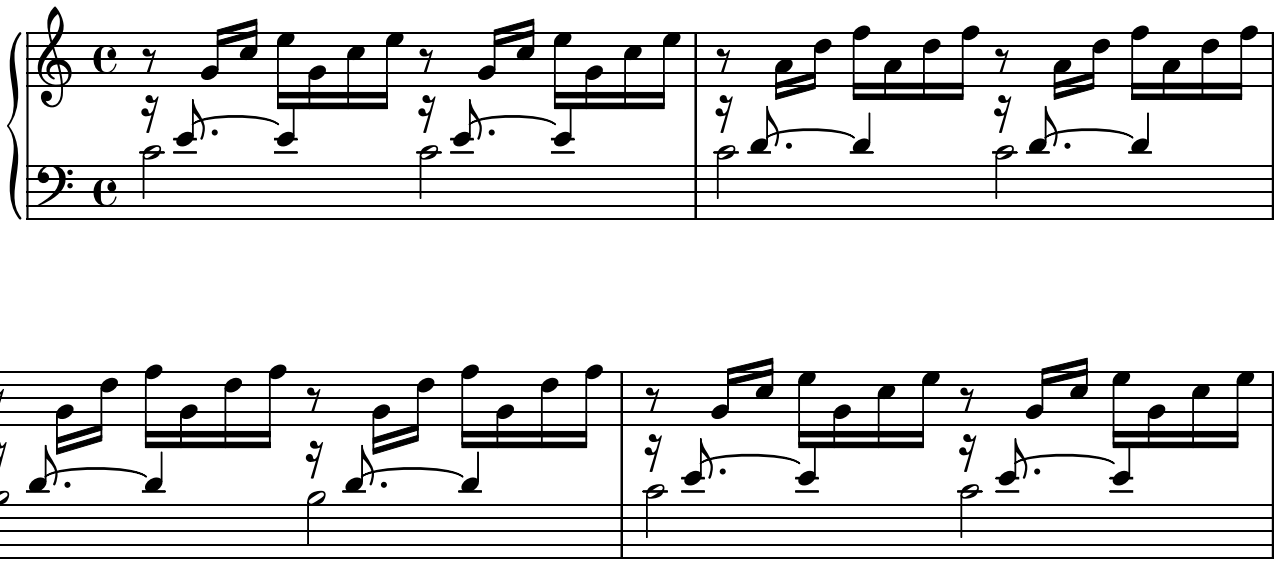
`move-specific-text.ly`

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



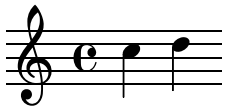
`music-box.ly`

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



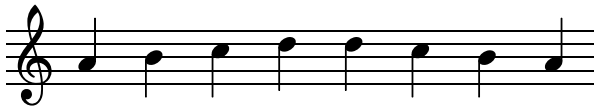
`music-creation.ly`

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.



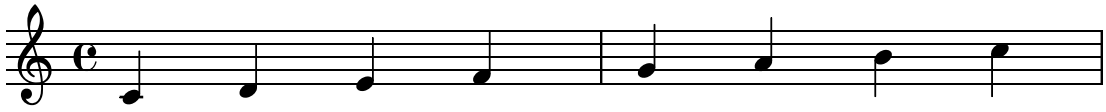
`no-bar-lines.ly`

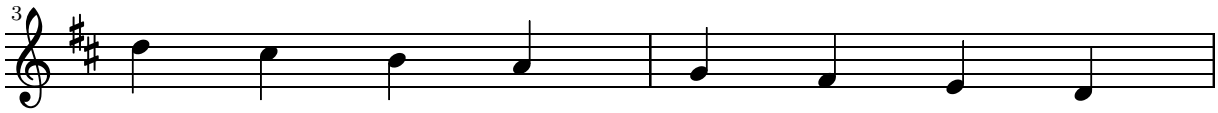
Engravers can be removed one by one. Here, the time signature and bar lines have been removed.



`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.



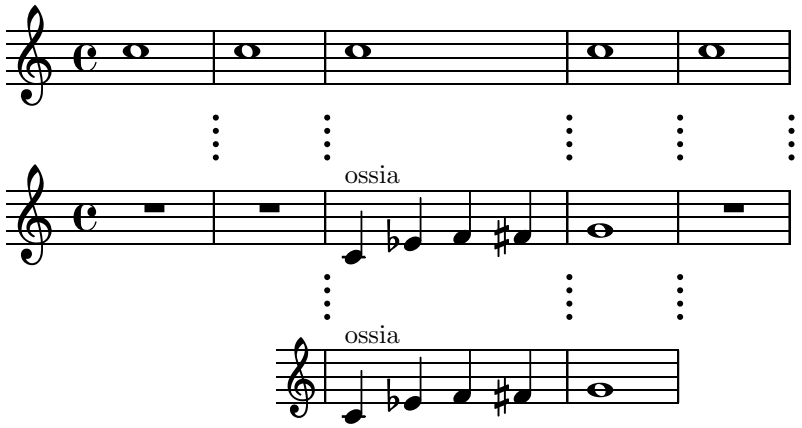


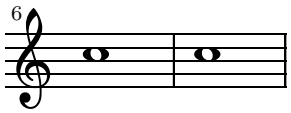
`no-staff.ly`

The printing of the staff lines may be suppressed by removing the corresponding engraver.

`ossia.ly`

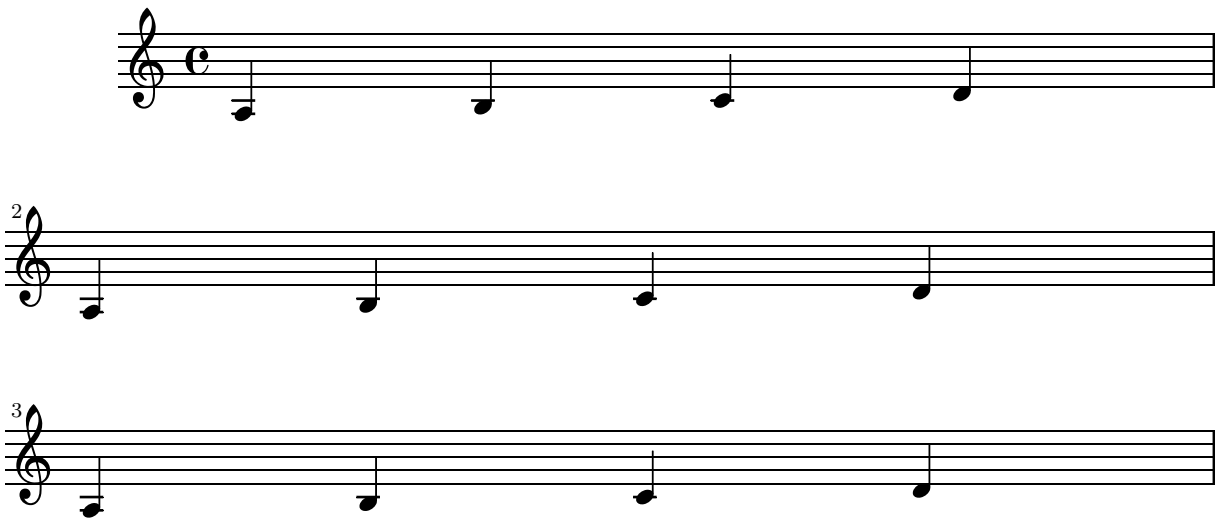
A temporary ossia in an instrumental part may be printed using a separate, short staff. A simpler solution is also given: instantiate a full staff, and let `RemoveEmptyStaffContext` take out the unused parts.

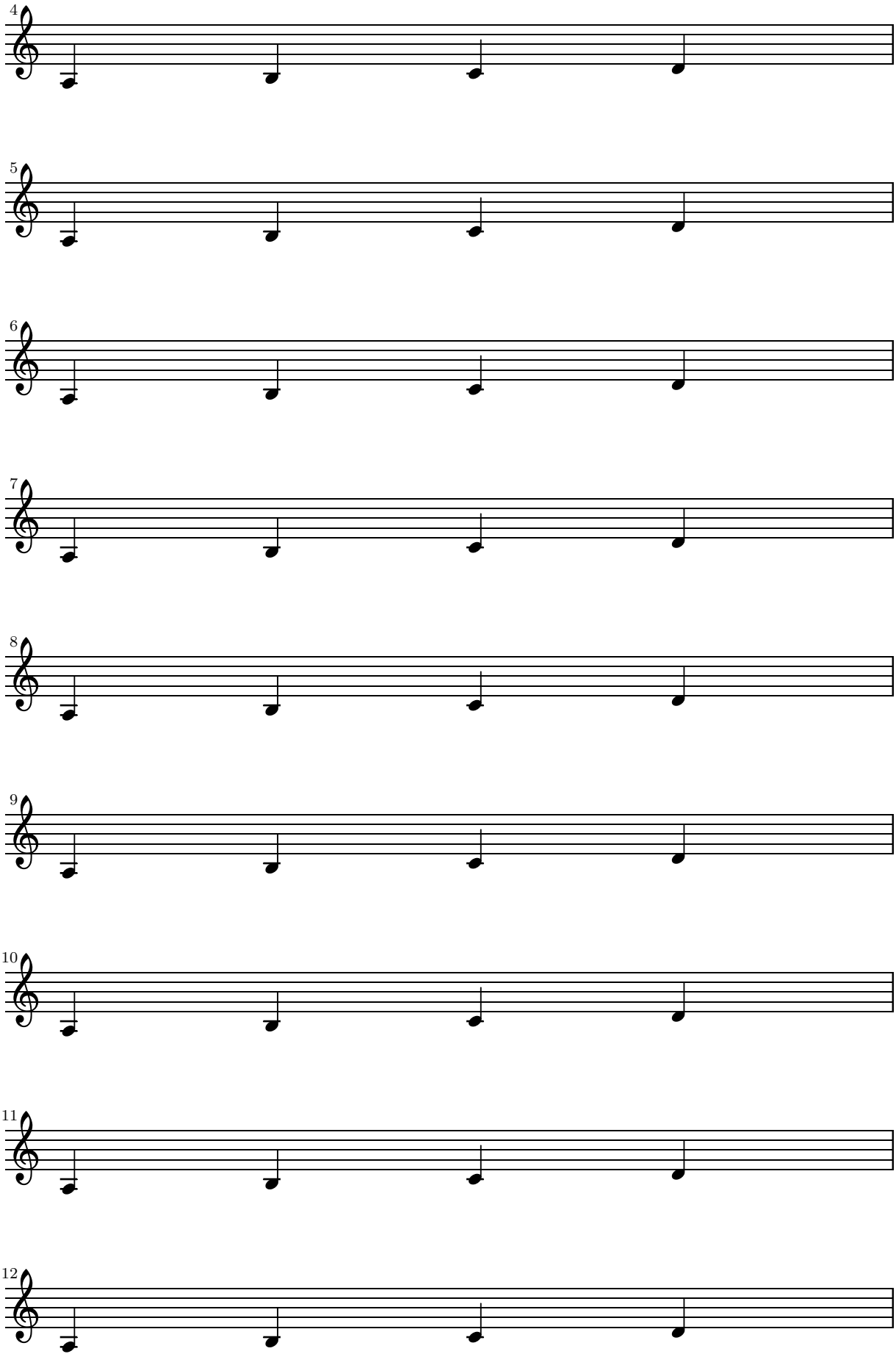


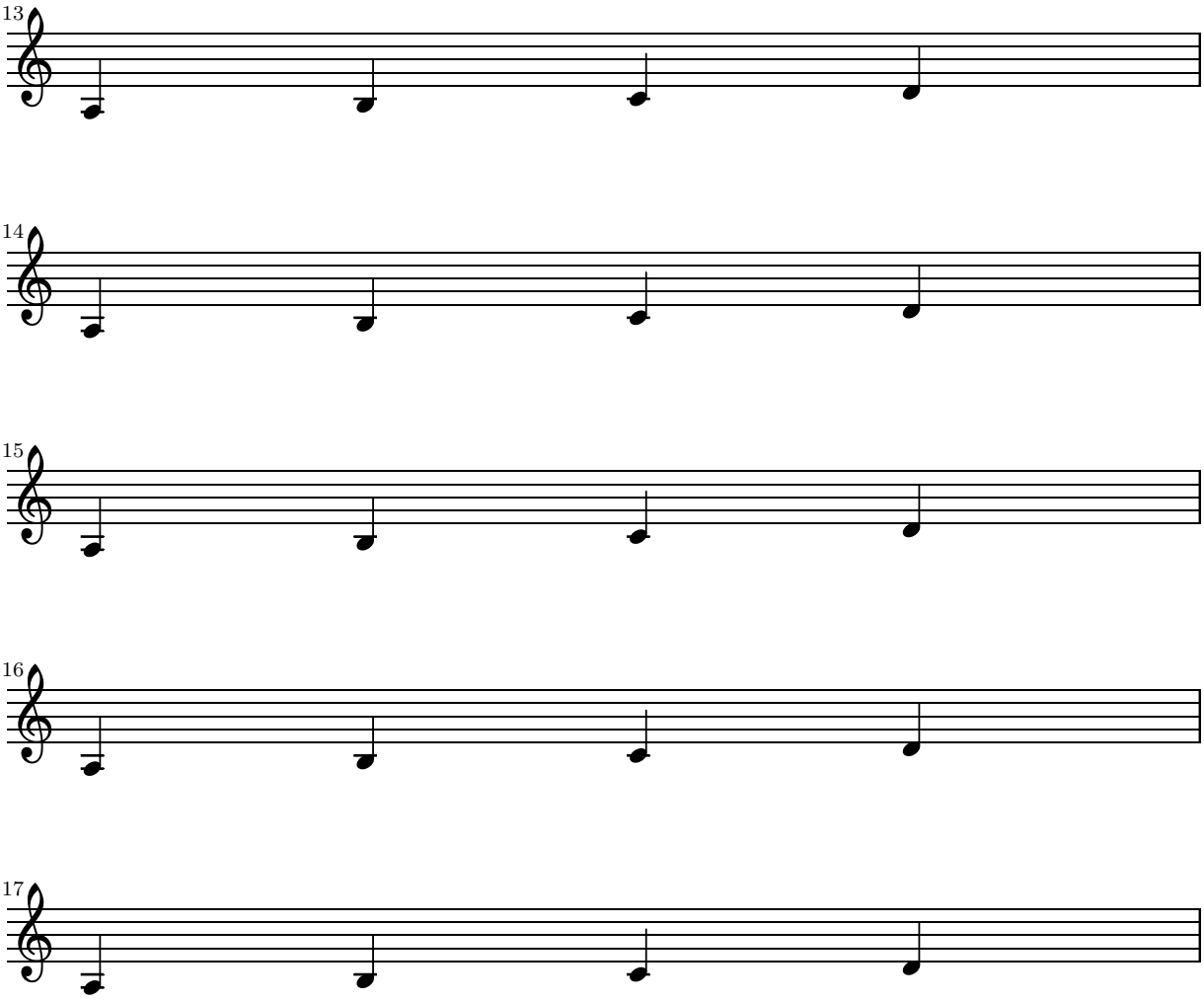


`page-breaks.ly`

Stress optimal page breaking. This should look nice on 4 a6 pages.

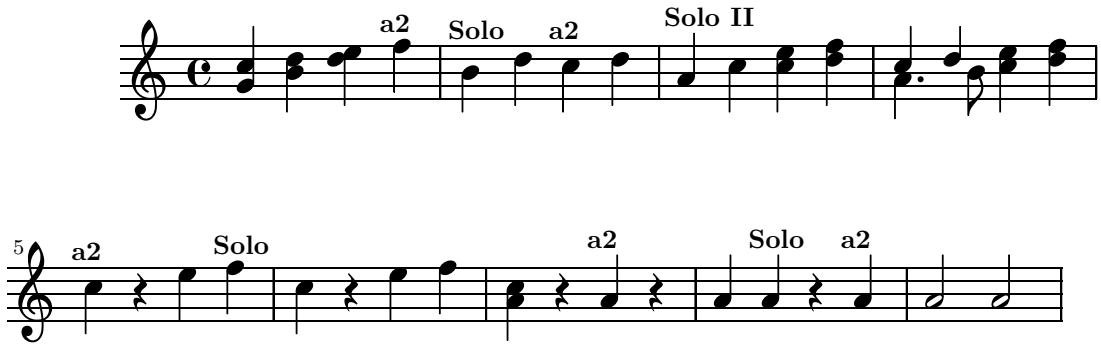



4
5
6
7
8
9
10
11
12



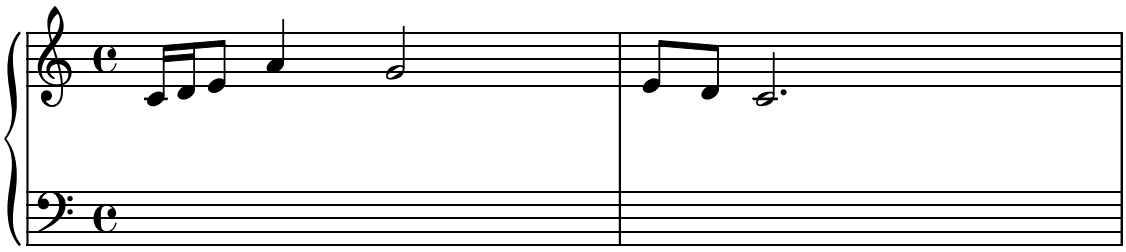
`part-combine.ly`

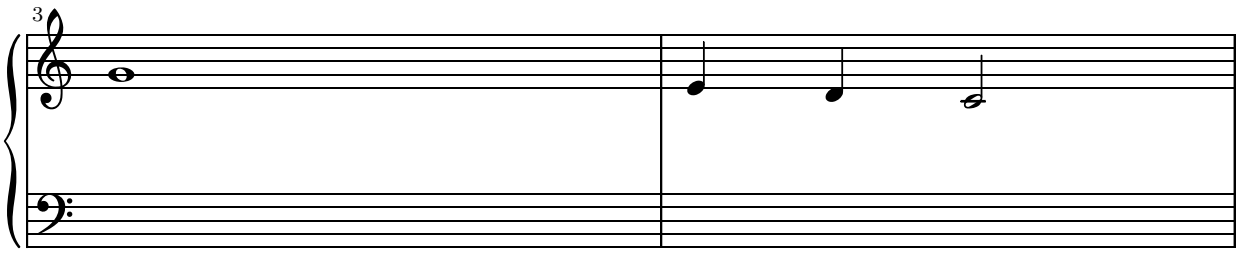
In orchestral scores and hymns, voices are traditionally combined into one staff. LilyPond has a part combiner that combines, or separates, two voices according to the actual rhythm and pitch. Configurable texts, such as “solo” and “à2”, are typeset automatically in appropriate places.



`partial-blank.ly`

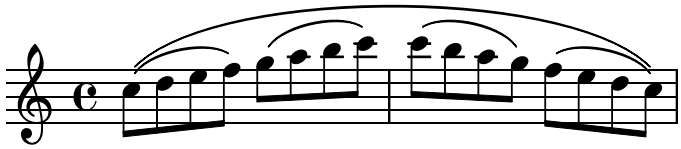
When entering partially typeset music (i.e. for students to be completed by hand), you may need the spacing that correspond to the timing of notes: all measures have same length, etc. It can be implemented by adding an invisible staff with a lot of fast notes.





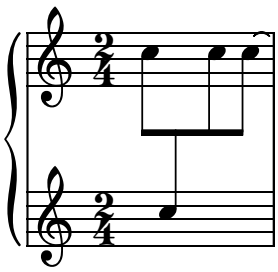
`phrasing-slur-height.ly`

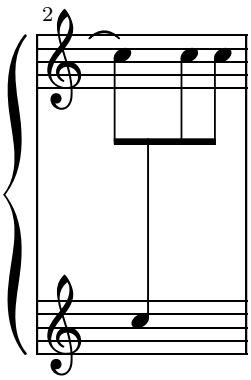
The `PhrasingSlur` can be made higher in order to avoid collision with other slurs.



`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.

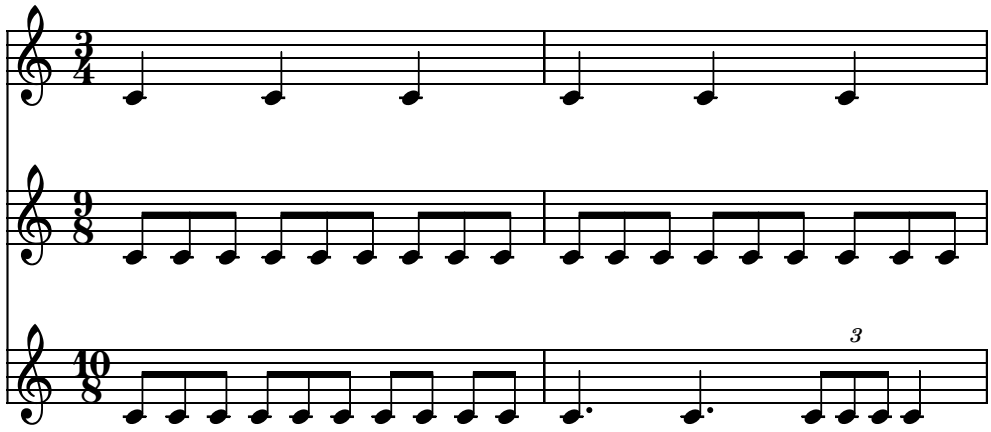




`polymetric-differing-notes.ly`

It is possible to have multiple time signatures, each one in an own staffs, at the same time, and have even a proper vertical alignment of the different durations. This is done, firstly, by setting a common time signature for each staff but replacing it manually using `timeSignatureFraction` to the desired fraction, and secondly, by scaling the printed durations to the actual, polymetric durations.

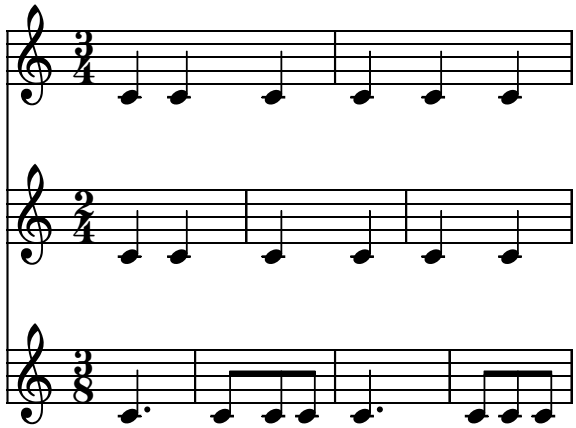
In this example, music with the time signatures of 3/4, 9/8 and 10/8 are used in parallel. In the second staff, shown durations are multiplied by 2/3, so that 2/3 * 9/8 = 3/4, and in the third staff, shown durations are multiplied by 3/5, so that 3/5 * 10/8 = 3/4.



`polymetric.ly`

You can have multiple time signatures occuring at the same time.

This is done by moving the timing engraver to staff context. Also, `Staff` should be given the alias `Timing` to make `\time` command work correctly. The spacing is aligned vertically, although the bar lines seem to distort the regular spacing.



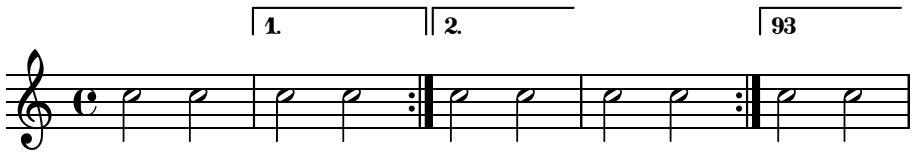
`preset-extent.ly`

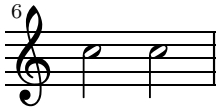
The object may be extended to larger sized by overriding their properties. The lyrics in this example have an extent of `(-10,10)`, which is why they are spaced so widely.

foo- bar baz

`repeat-manual.ly`

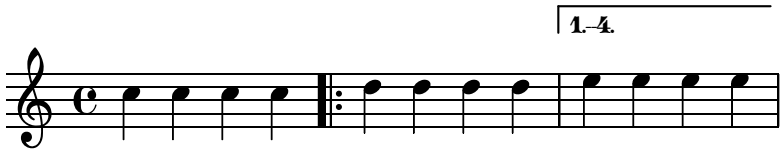
By controlling manually the signs and numbers in repeats, an unusual output can be produced.



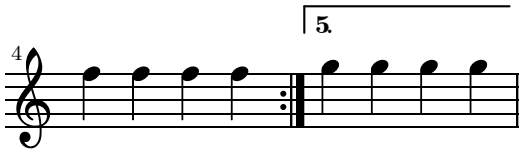


`repeat-shorter-bracket.ly`

By setting `voltaSpannerDuration`, the horizontal length of a volta bracket can be shortened.



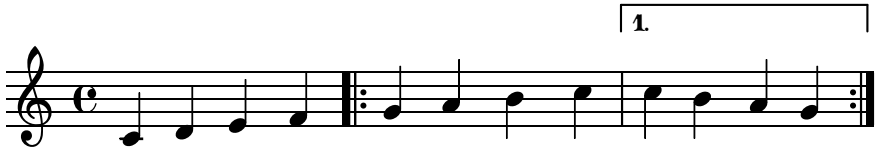
intro **chorus** **one**
chorus **two**
chorus **three**
chorus **four**



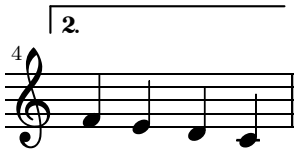
verse **five**
verse
verse
verse

`repeat.ly`

Alternate lyrics can be used, as well as alternate notes for repeats.



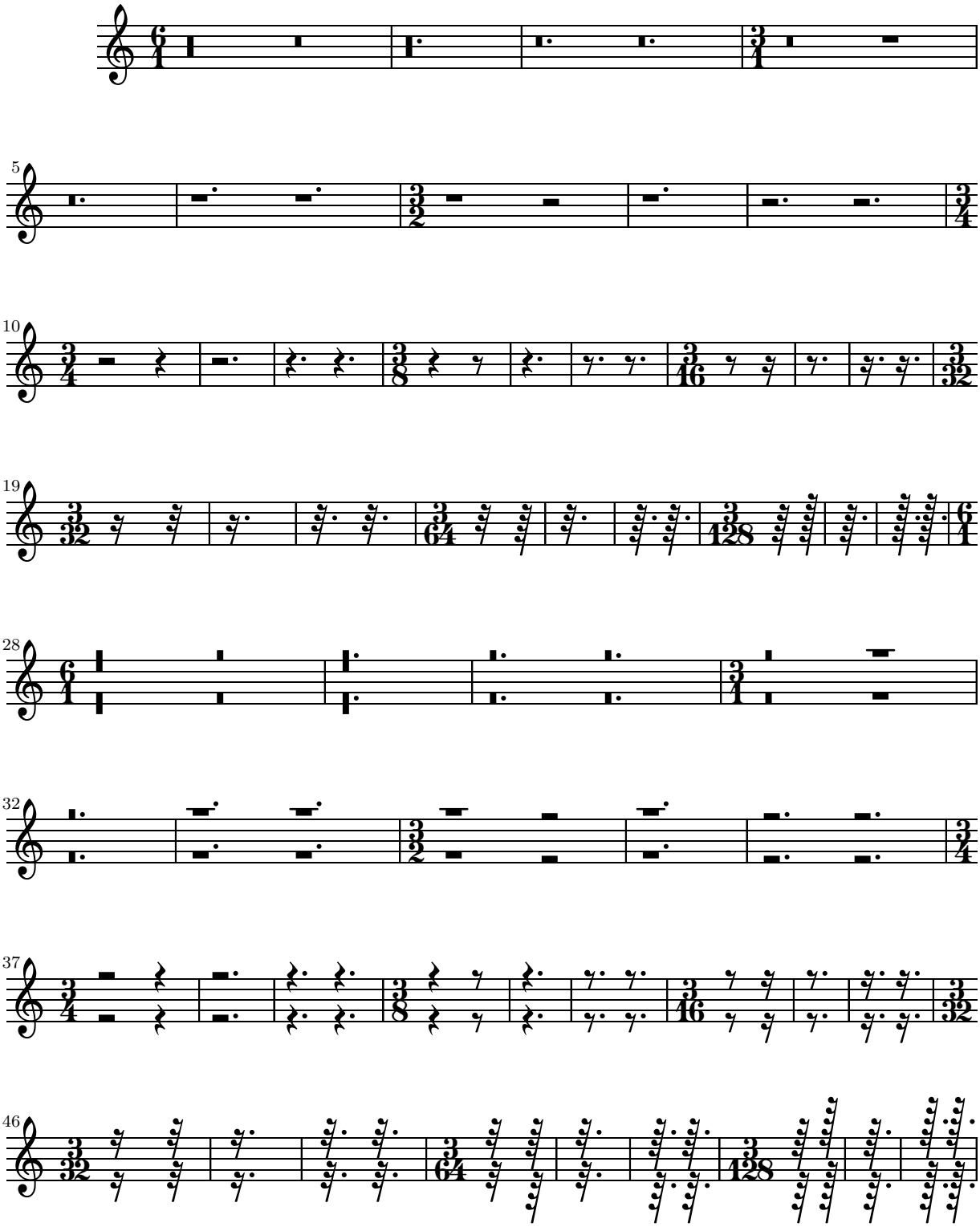
De eer- ste maat en dan twee keer en dan nog dit
een koe- plet



er ach- ter aan

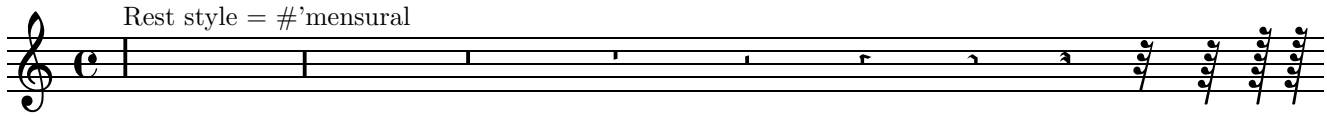
`rest-dot-positions.ly`

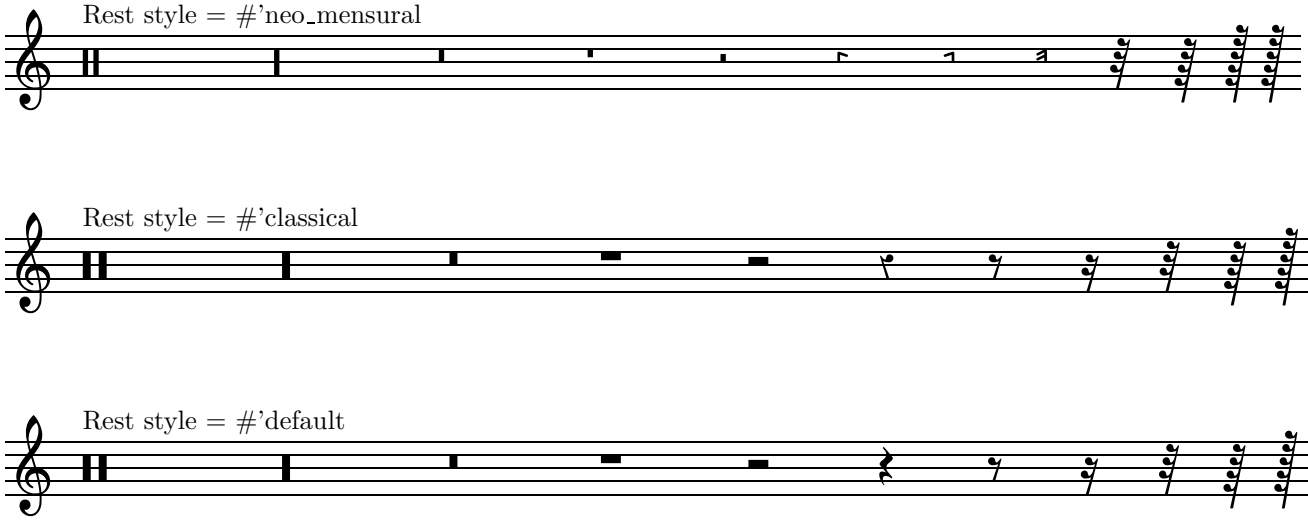
Dots of rests should follow the rest positions.



`rests.ly`

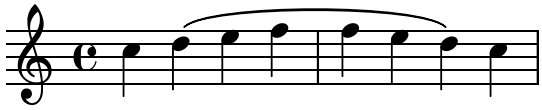
Rests may be used in various styles.





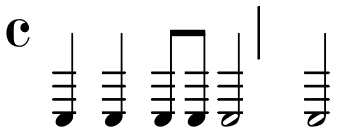
`reverse-music.ly`

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.



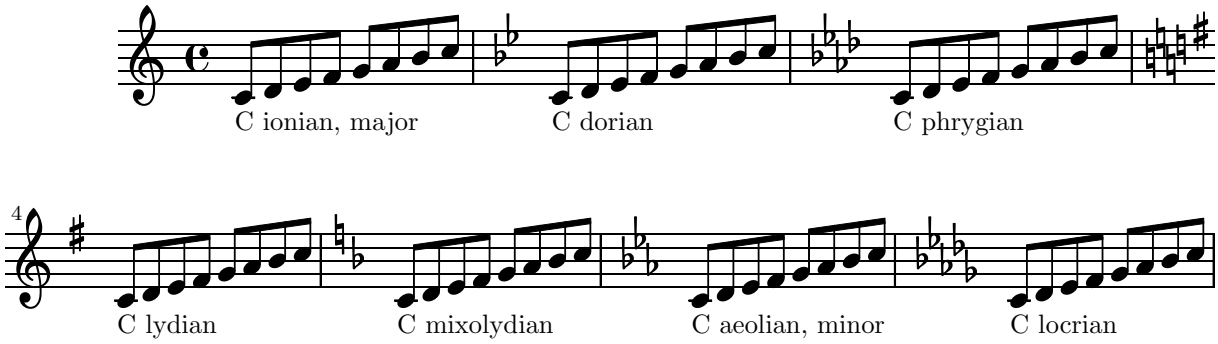
`rhythm-excercise.ly`

Rythmic exercises may be produced by removing the `Clef` engraver, putting all notes to the same pitch and using transparent staff lines.



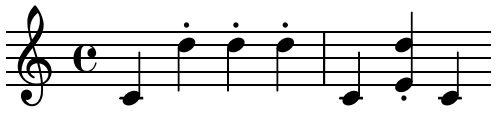
`scales-greek.ly`

In addition to major and minor keys, the key can be given also in terms of greek, modal scales: ionian (= major), dorian, phrygian, lydian, mixolydian, aeolian (= minor), and locrian. All these scales are in the key of C.



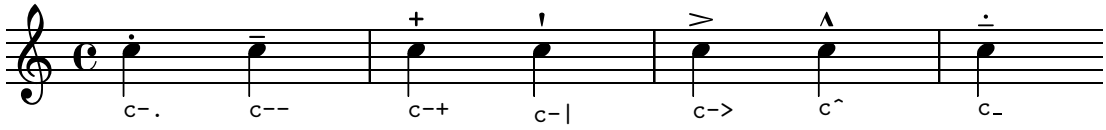
`scheme-interactions.ly`

Using `ly:export`, the result of Scheme expressions can be passed as LilyPond input. Within a Scheme expression, you can use, define or change the corresponding variables. In this example, the D-s and E-s are generated using scheme functions, and between there are manually (without Scheme) entered C-s.



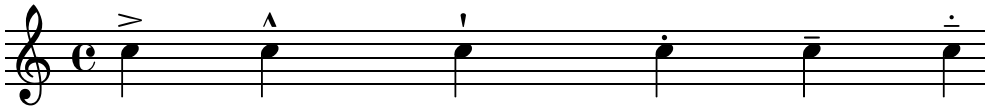
`script-abbreviations.ly`

Some articulations may be entered using an abbreviation.

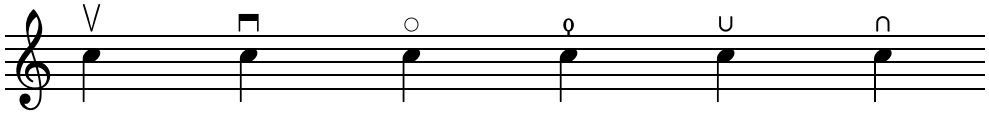


`script-chart.ly`

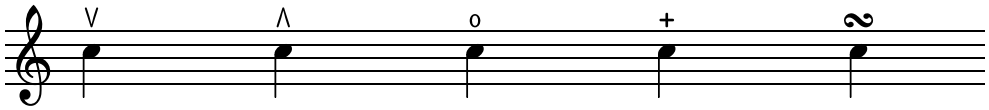
This chart shows all articulations, or scripts, that feta font contains.



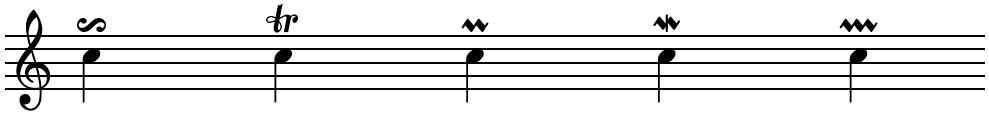
accent marcato staccatissimo staccato tenuto portato



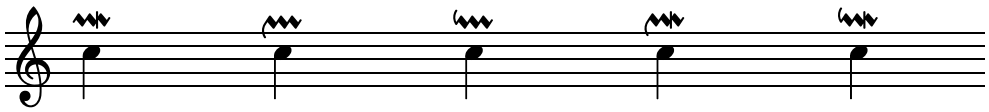
upbow downbow flageolet thumb lheel rheel



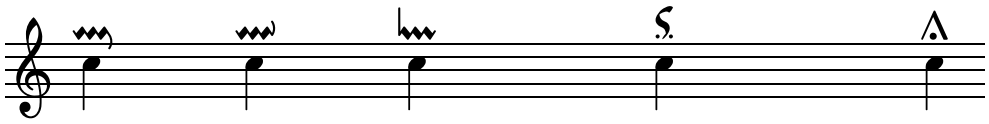
ltoe rtoe open stopped turn



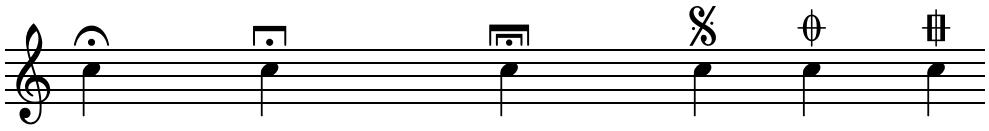
reverseturn trill prall mordent prallprall



prallmordent upprall downprall upmordent downmordent



`pralldown` `prallup` `lineprall` `signumcongruentiae` `shortfermata`

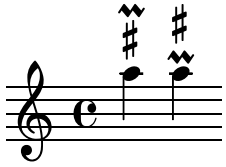


`fermata` `longfermata` `verylongfermata` `segno` `coda` `varcoda`

`script-priority.ly`

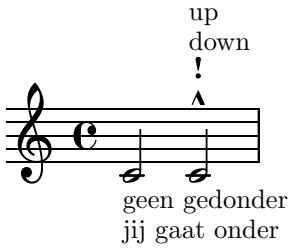
Relative placements of different script types can be controlled by overriding `script-priority`.

In this example, accidentals are put either below or above other script symbols.



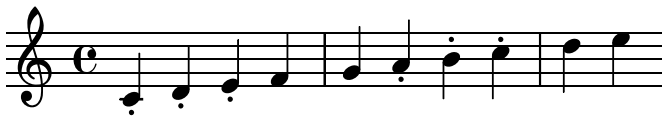
`script-stack.ly`

Text and articulations may be stacked on top of each other.



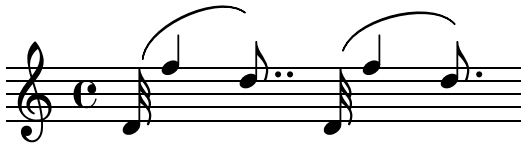
`separate-staccato.ly`

You can enter notes and articulations separately, and merge them into one thread. In this example, a repead series of staccato dots is attached to the notes.



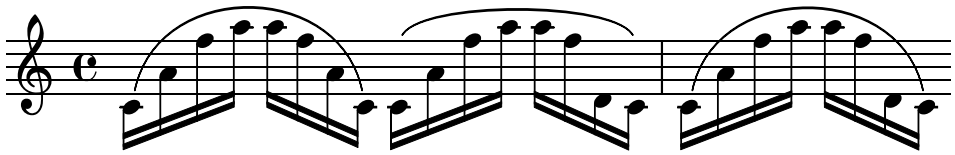
`slur-attachment-override.ly`

In some cases, you may want to control the attachment points of a slur by hand.



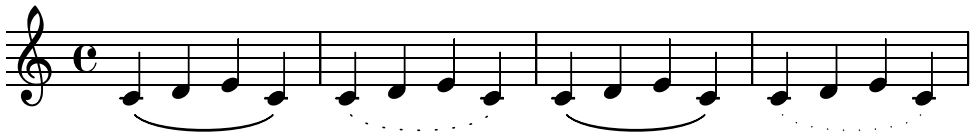
`slur-beautiful.ly`

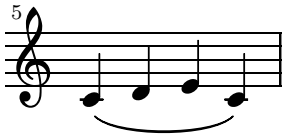
The curvature of a slur is adjusted to stay away from note heads and stems. When the curvature would increase much, the slur is reverted to its default shape. The Slur's property `beautiful` (which is loosely related to the enclosed area between the slur and the notes) controls the transition point, and by increasing that value you may keep slurs more curved.



`slur-dash.ly`

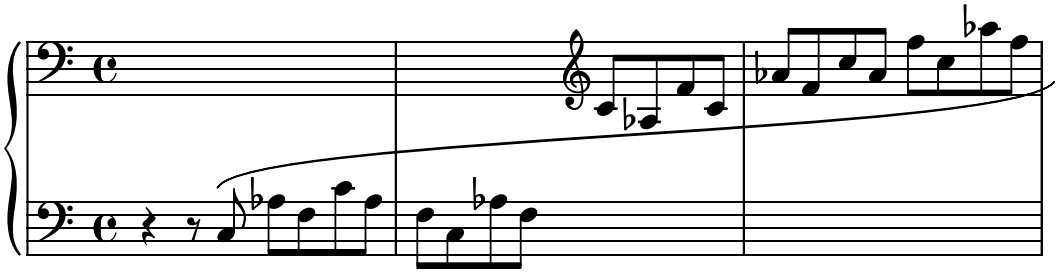
The appearance of slurs may be changed from solid to dotted or dashed.

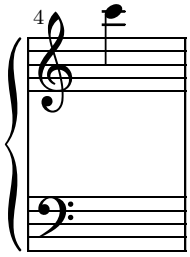




`slur-manual.ly`

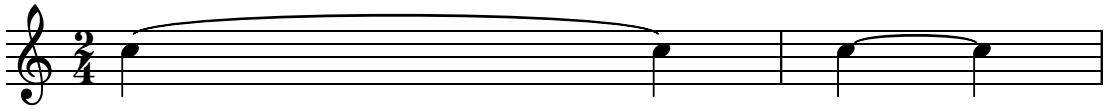
In extreme cases, you can resort to setting the `control-points` of a slur manually, althout it involves a lot of trial and error. Be sure to force line breaks at both sides, since different horizontal spacing will require rearrangement of the slur.





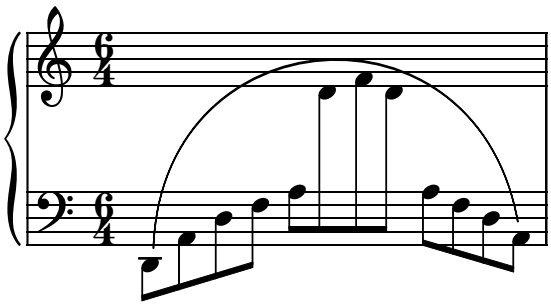
`slur-minimum-length.ly`

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



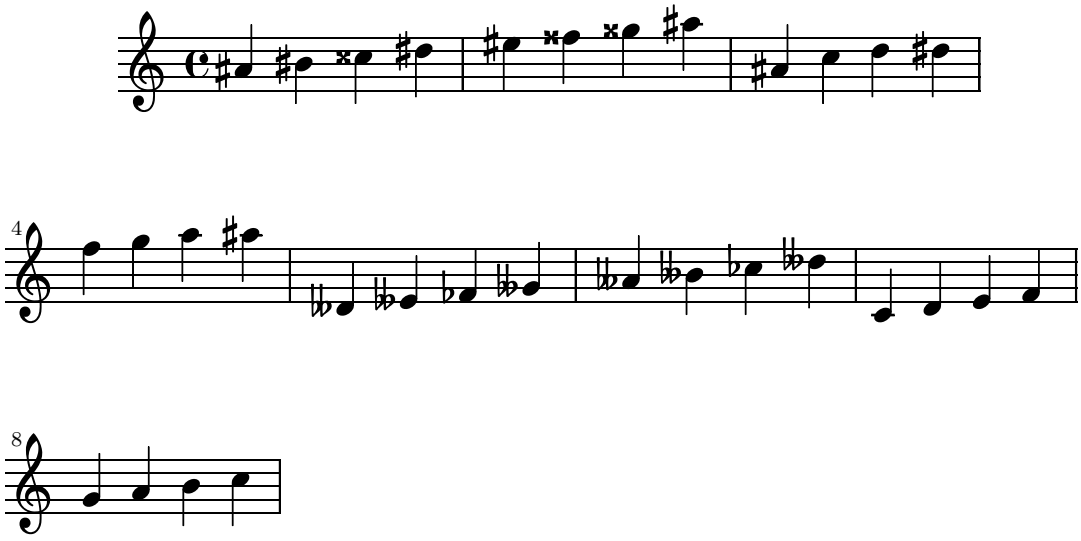
`slur-ugly.ly`

Strange slurs can be produced by setting properties by hand.



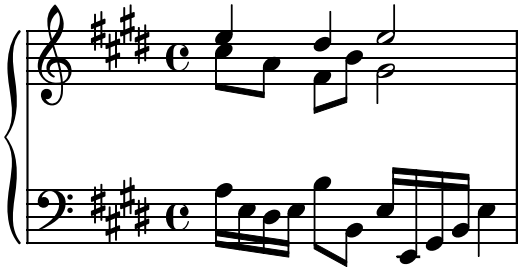
`smart-transpose.ly`

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 that manner, the most natural enharmonic notes are chosen in this example.



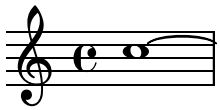
`spacing-2.ly`

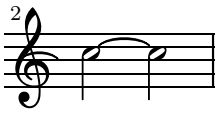
When stretching notes, every note should stretch according to its duration. Eighth notes should be spaced equidistantly.



`spanner-after-break-tweak.ly`

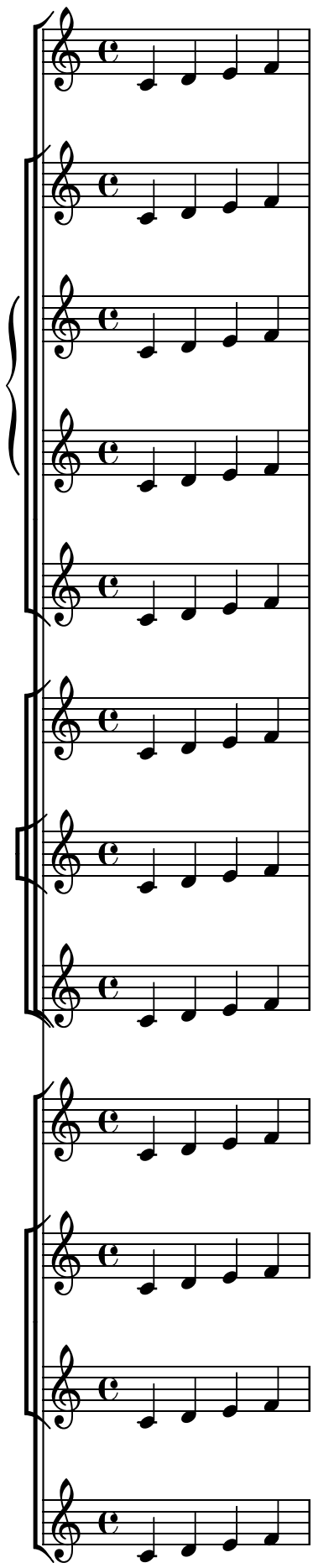
In order to selectively change the properties of spanners after a line break, Scheme code must be used. In thas manner, the tie after the line break in this example is moved around.





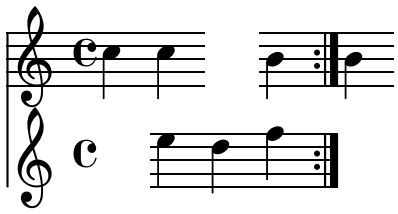
staff-bracket.ly

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.



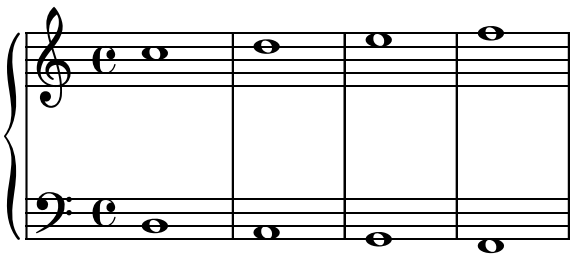
`staff-container.ly`

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 `ew 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.)



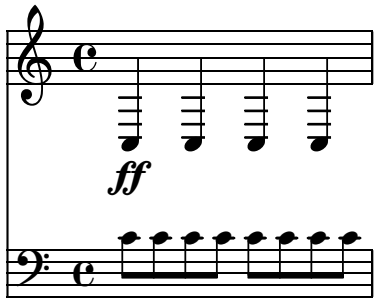
`staff-lines.ly`

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



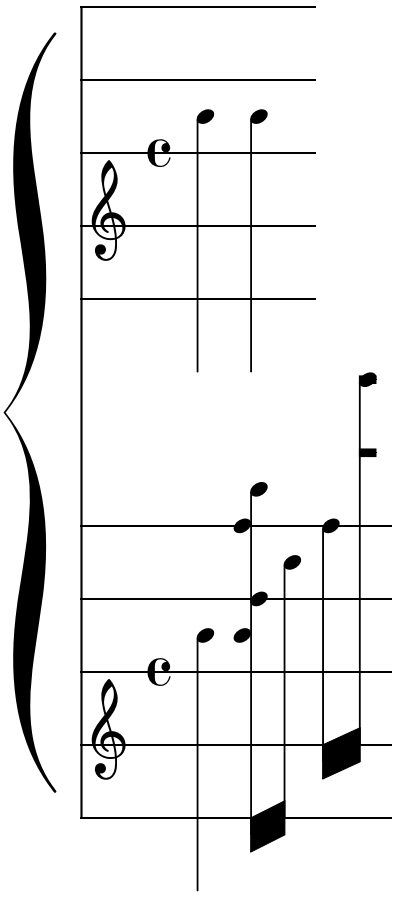
`staff-size.ly`

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



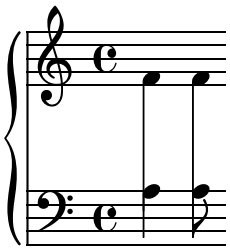
`staff-space.ly`

By just increasing `staff-space` on a staff, you may produce strange results.



`stem-cross-staff.ly`

The chords which exceptionally cross staves may be produced by increasing the length of the stem in the lower stave, so it reaches the stem in the upper stave, or vice versa.

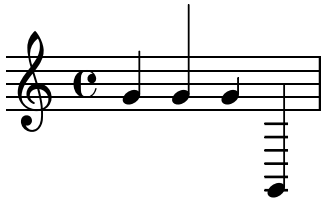


`stem-extend.ly`

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

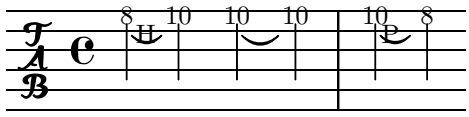
`stem-length.ly`

The length of stems can be altered.



`tablature-hammer.ly`

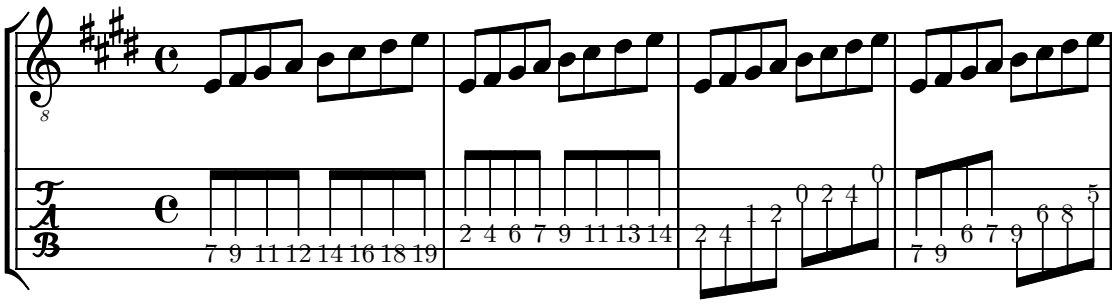
A hammer in tablature can be faked with slurs.



`tablature.ly`

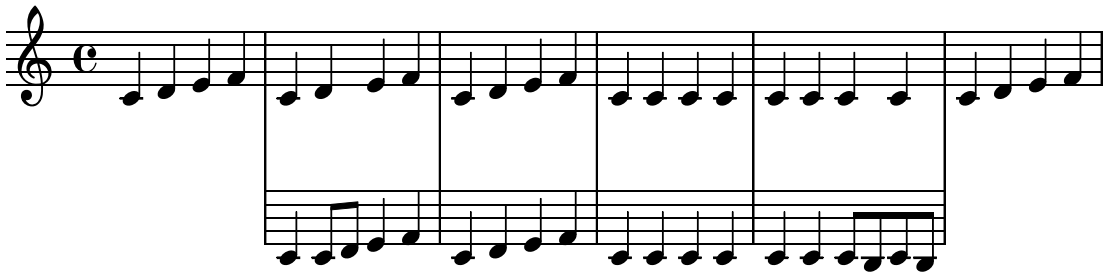
Tablature is internally done by overriding the note-head formatting function and let it act on a 6-line staff. A special engraver takes then care of choosing the fret and converting the pitch to a number.

Thus, by providing the fret numbers, the same music can be generated both for a normal and tabulature staffs. By default, the fret is the smallest possible, according to `minimumFret`.



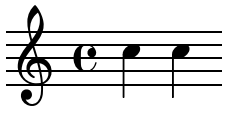
`temporary-stave.ly`

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.



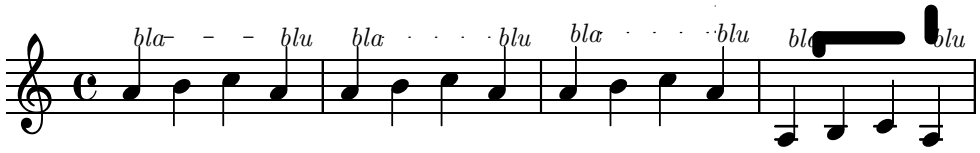
`text-rotate.ly`

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).

rotatebox90Chor 

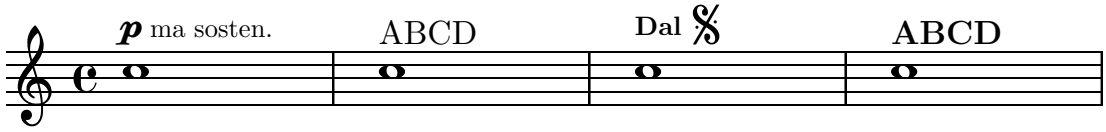
`text-spanner.ly`

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



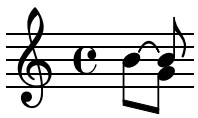
`textscript.ly`

There are different fonts and glyphs to be used with `\markup` command.



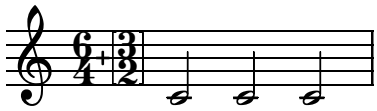
`tie-cross-voice.ly`

Cross voice ties can be faked by using transparent noteheads.



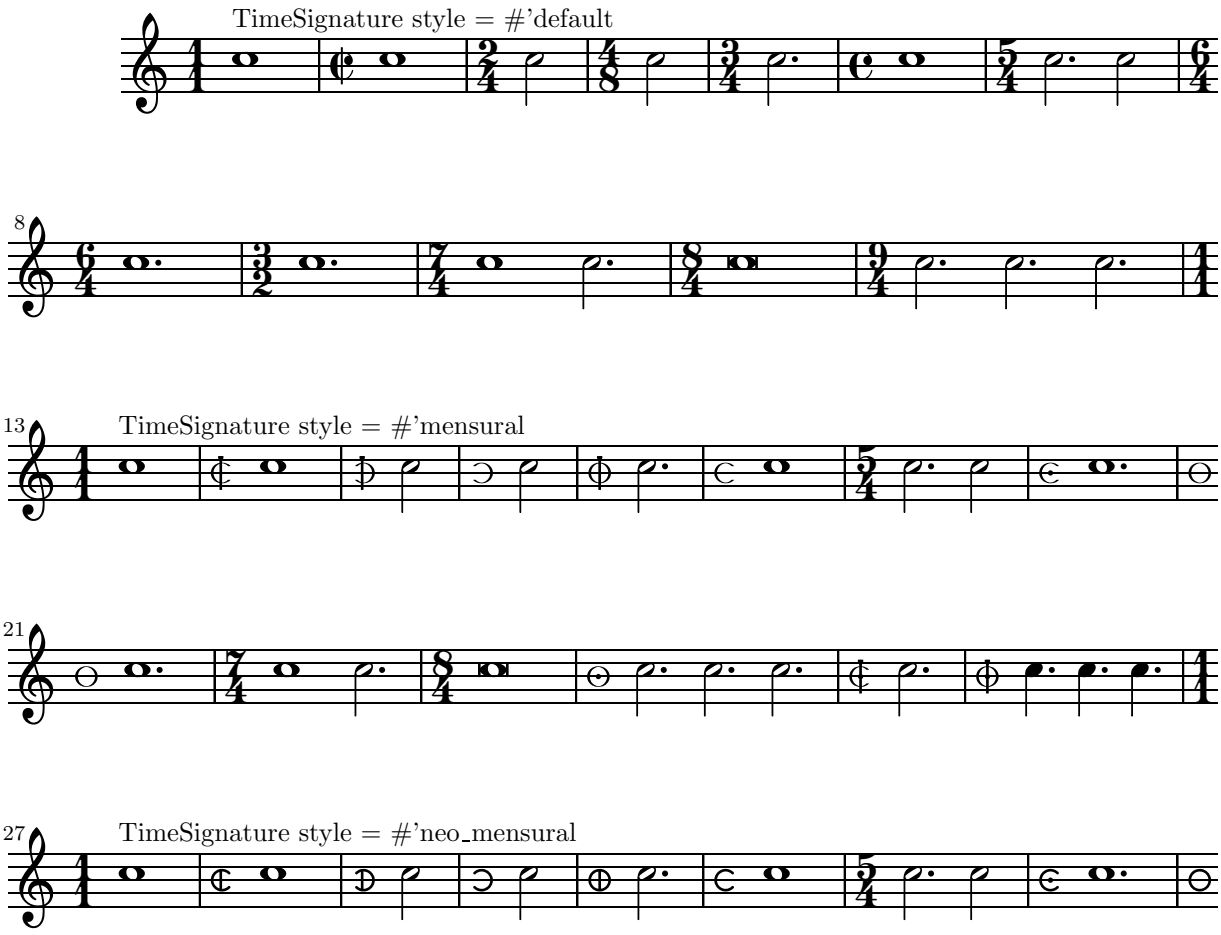
`time-signature-double.ly`

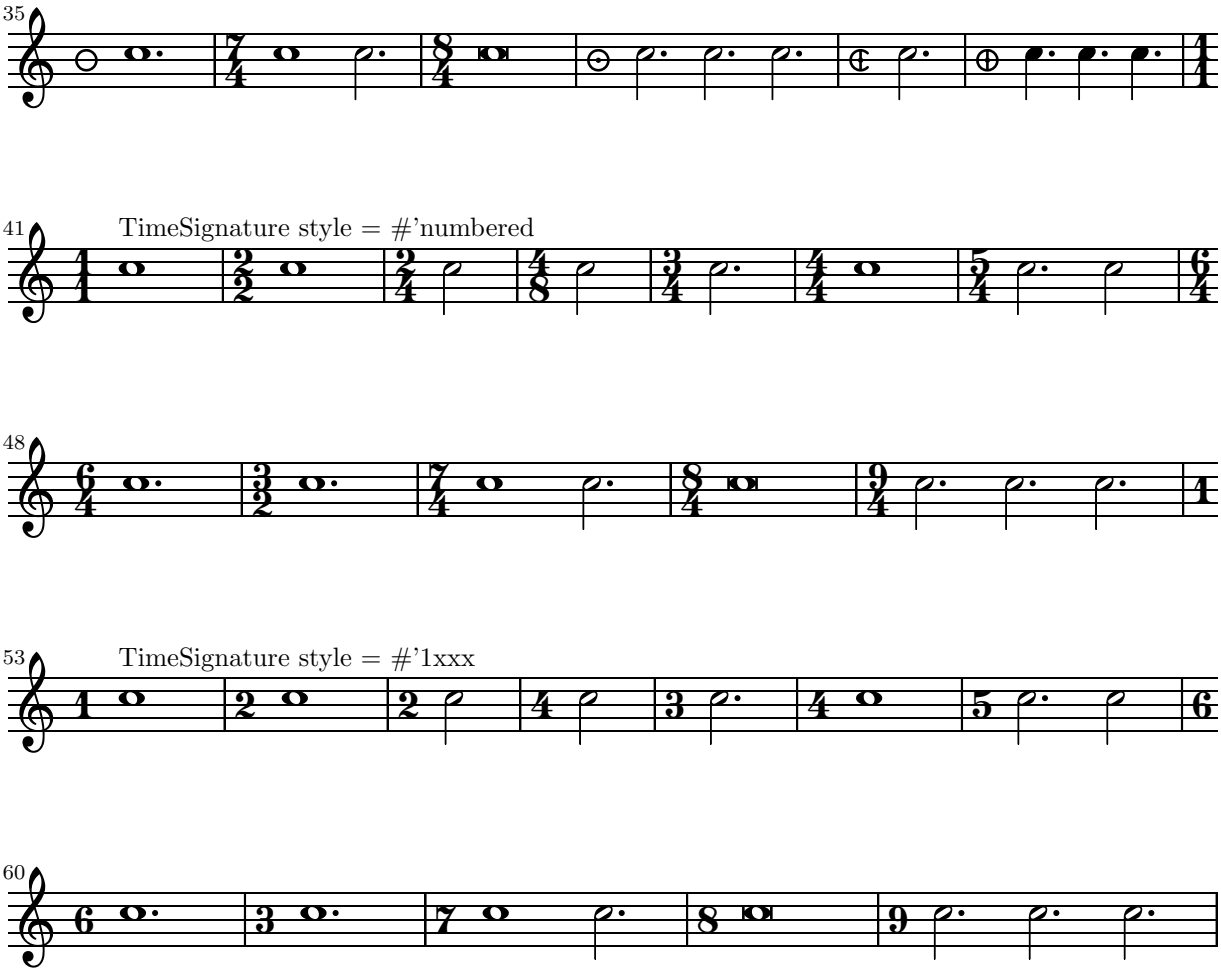
Double time signatures are not supported explicitly, but they can be faked with markups and overriding formatting routines.



`time.ly`

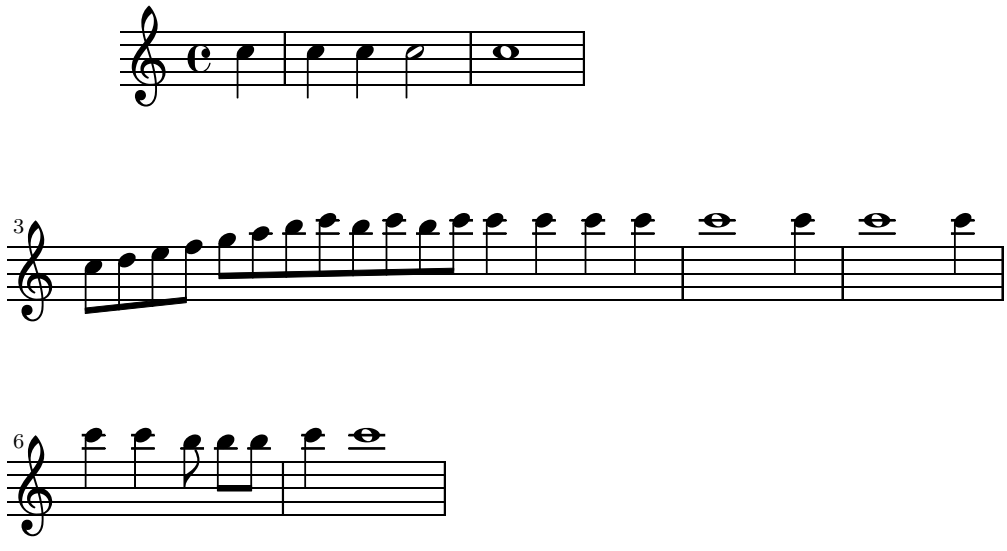
The different styles for time signatures are shown in this file.





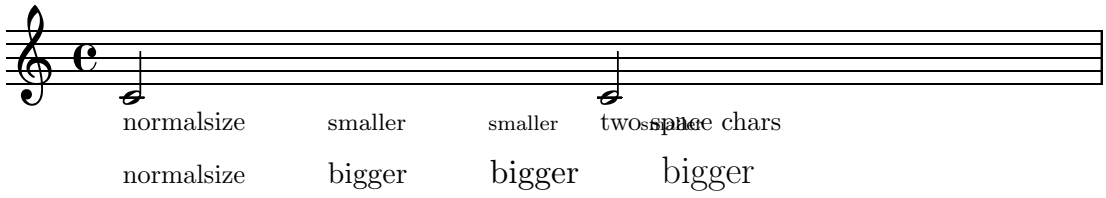
`timing.ly`

You can alter the length of bars by setting explicitly `measureLength` or by resetting `measurePosition`.



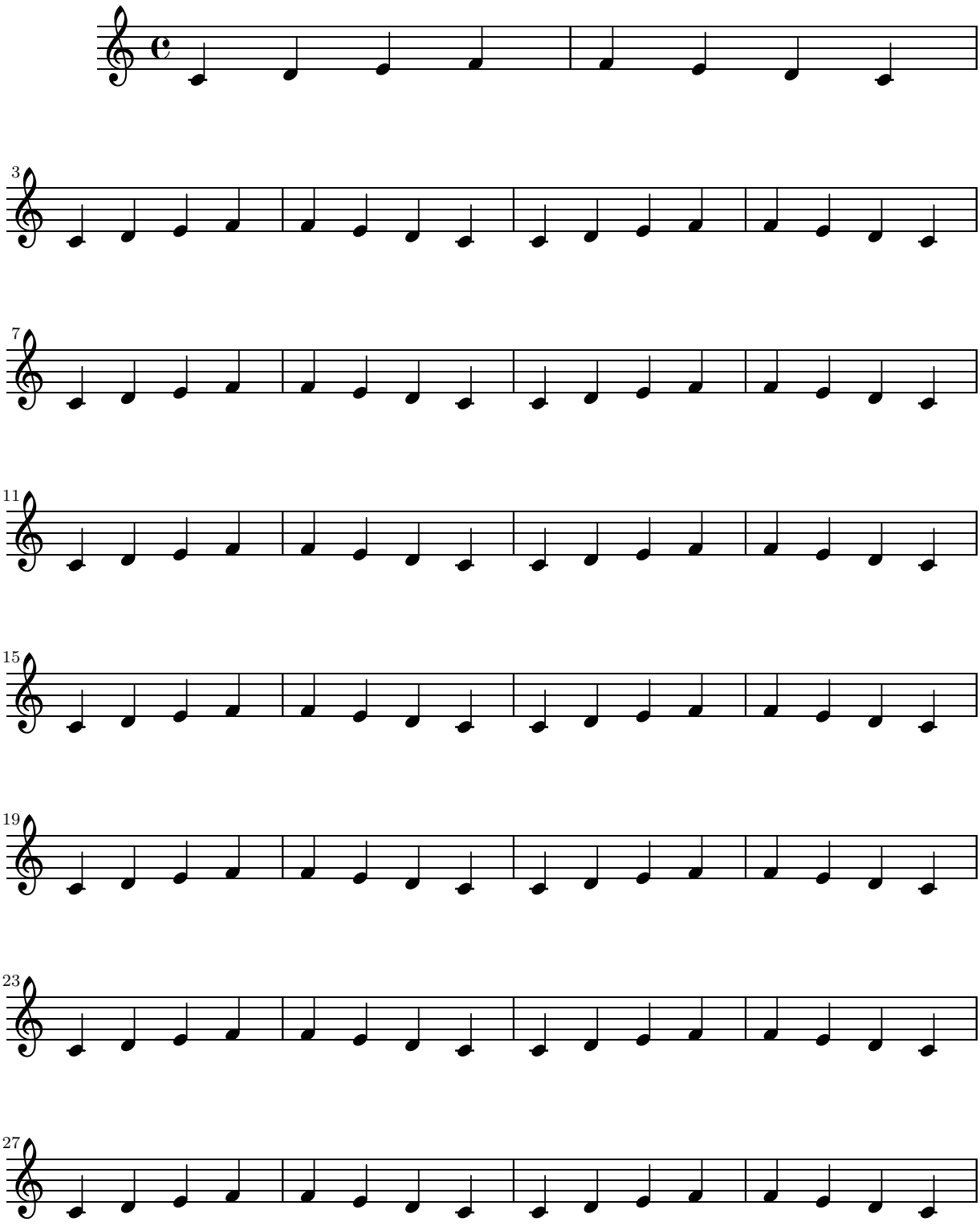
`title-markup.ly`

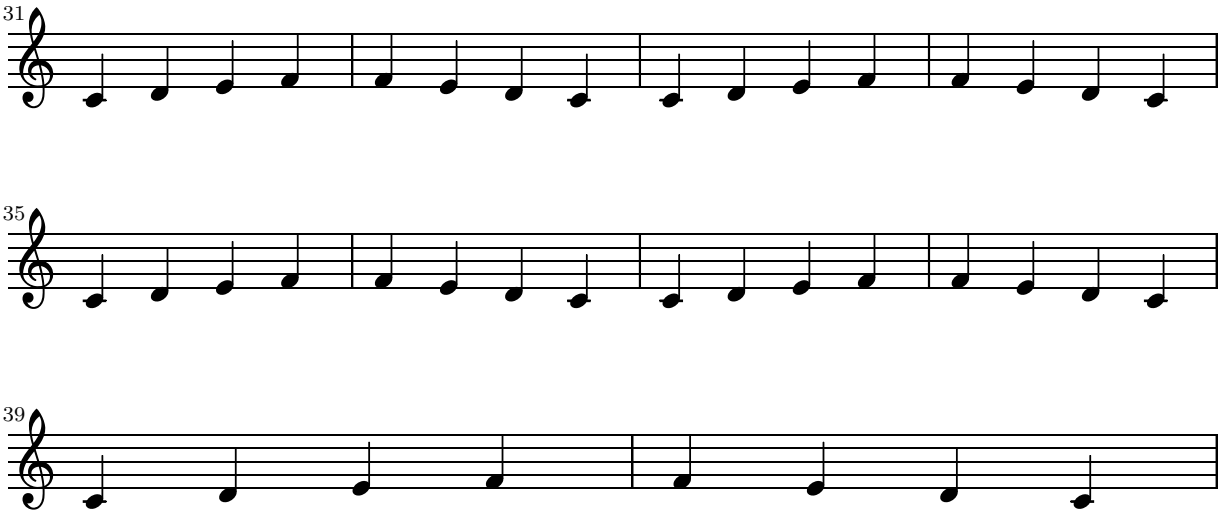
Make titles using markup. Only in direct PostScript output.



`title.ly`

This example tests titling. By processing with `lilypond` (not `lilypond-book`), you will see all the titles.



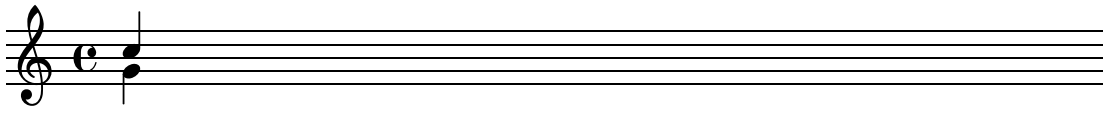


`to-xml.ly`

The input representation is very generic. Therefore, it should not be hard to convert it to XML or a similar format:

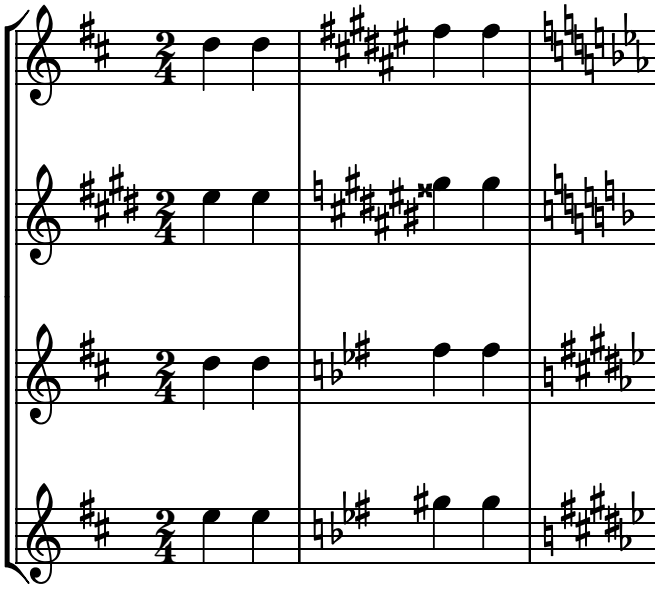
```
<music
   type="score">
<SequentialMusic>
<SimultaneousMusic>
<EventChord>
<NoteEvent>
<pitch
   octave="1"
   notename="0"
   alteration="0">
</pitch>
<duration
   log="2"
   dots="0"
   numer="1"
   denom="1">
</duration>
</NoteEvent>
</EventChord>
<VoiceSeparator>
</VoiceSeparator>
<EventChord>
<NoteEvent>
<pitch
   octave="0"
   notename="4"
   alteration="0">
</pitch>
<duration
   log="2"
   dots="0"
   numer="1"
   denom="1">
</duration>
```

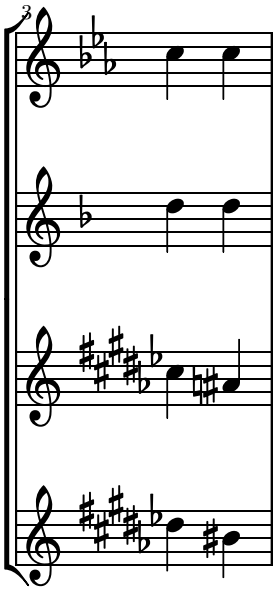
```
</NoteEvent>
</EventChord>
</SimultaneousMusic>
</SequentialMusic></music>
```



`transposition.ly`

Transposing has also an effect key signature, if it is given using `\key`. If `keySignature` is set explicitly instead, the key signature is not transposed.



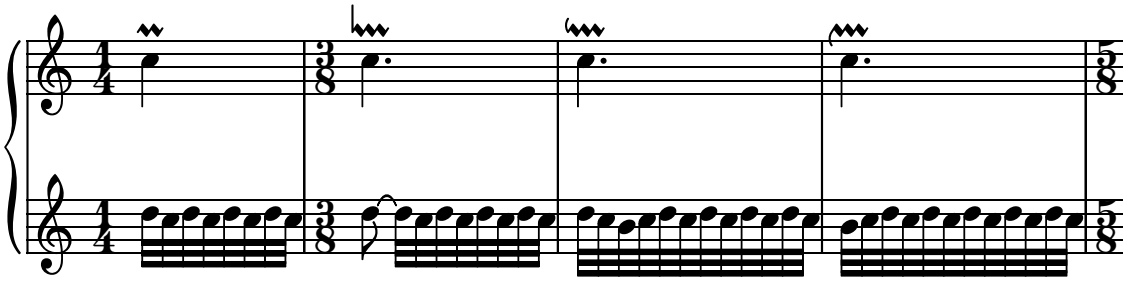


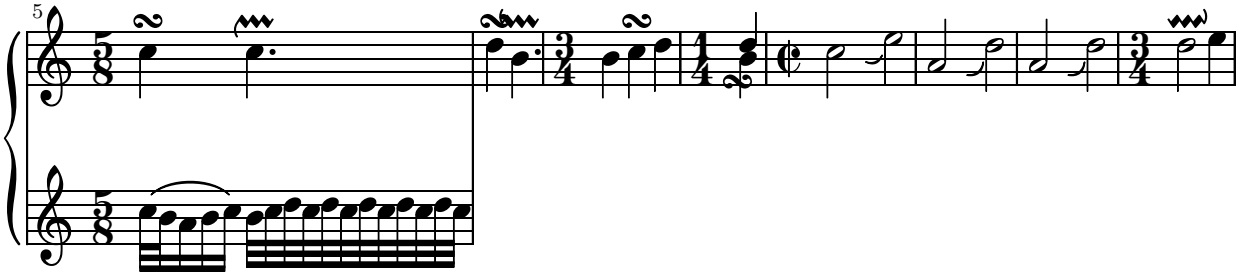
`trill.ly`

The extended trill may be produced using `TextSpanner` with `trill` spanner style.

`trills.ly`

Trills, pralls and turns may also be written out in full. Here the D'Anglebert system (1689) is shown.

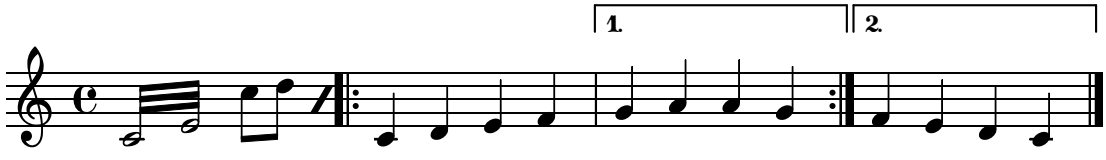


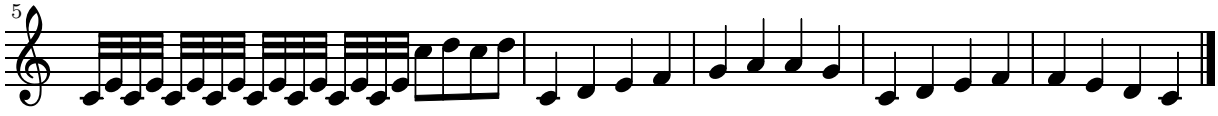


Double cadence

`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: 2.2.6

Processed with LilyPond version 2.2.6

`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. This requires adding an engraver to `ChordNames`, and setting `voltaOnThisStaff` correctly.