`+.ly`

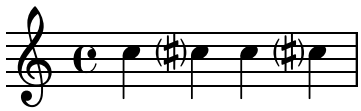
0.1 Introduction

This document presents a feature test for LilyPond 2.2.6. When the text correspond with the shown notation, we consider LilyPond Officially BugFree (tm). This document is intended for finding bugs, and documenting bugfixes.

TODO: order of tests (file names!), test only one feature per test. Smaller and neater tests.

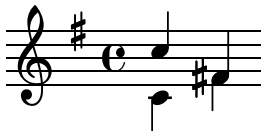
`accidental-cautionary.ly`

Cautionary accidentals are indicated using either parentheses (default) or smaller accidentals.



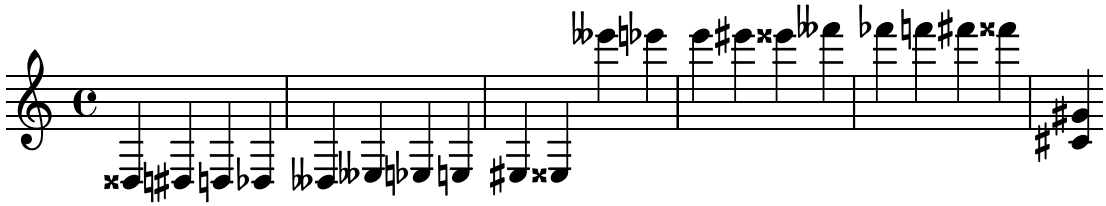
`accidental-double.ly`

If two forced accidentals happen at the same time, only one sharp sign is printed.



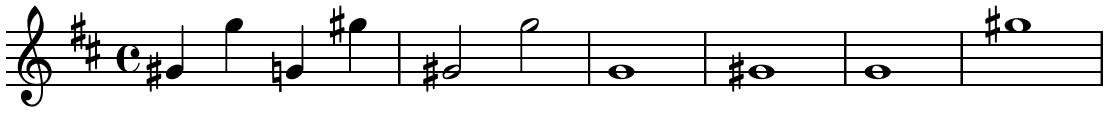
`accidental-ledger.ly`

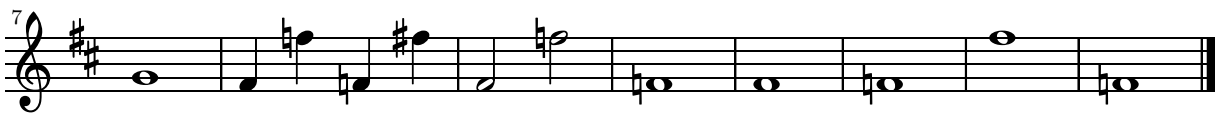
Ledger lines are shortened when there are accidentals. This happens only for the single ledger line close to the note head, and only if the accidental is horizontally close to the head.



`accidental-octave.ly`

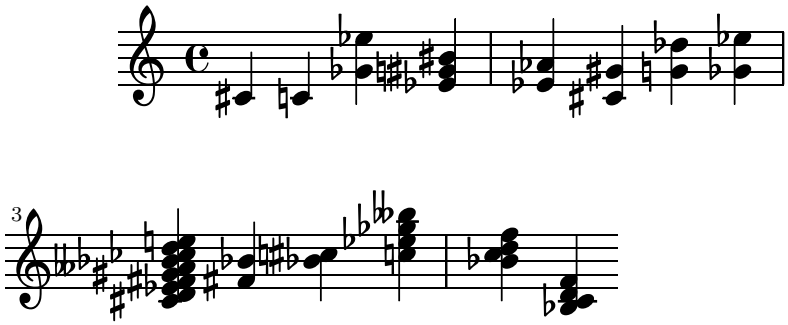
This shows how accidentals in different octaves are handled. (DOCME)





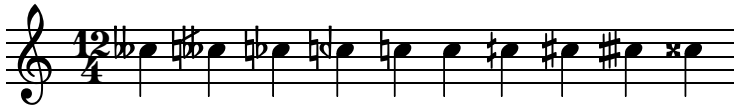
`accidental-placement.ly`

Accidentals are placed as closely as possible. Accidentals in corresponding octaves are aligned. The top accidental should be nearest to the chord. The flats in a sixth should be staggered.



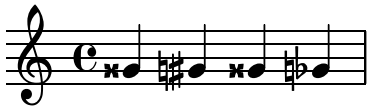
accidental-quarter.ly

Quarter tone notation is supported, including threequarters flat.



accidental-single-double.ly

A sharp sign after a double sharp sign, as well as a flat sign after a double flat sign is automatically prepended with a natural sign.



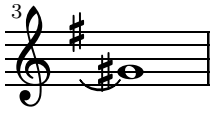
gisis gis gisis ges

accidental-tie.ly

The second and third notes should not get accidentals, because they are tied to a note. However, an accidental is present if the line is broken at the tie, which happens for the G sharp.

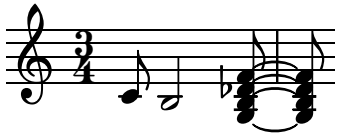
f f f fis gis

gis



accidental-unbroken-tie-spacing.ly

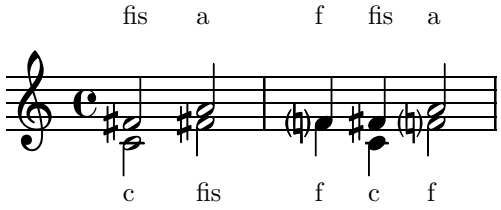
Tied accidentaled notes (which cause reminder accidentals) do not wreak havoc in the spacing when unbroken.



accidental-voice.ly

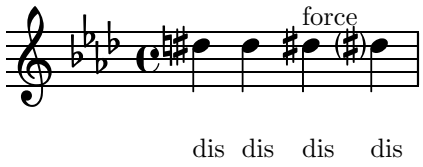
This shows how modern cross voice auto cautionary accidentals are handled. The first two fisses get accidentals because they belong to different voices. The first f gets cautionary natural

because of previous measure. The last f gets cautionary natural because fis was only in the other voice.



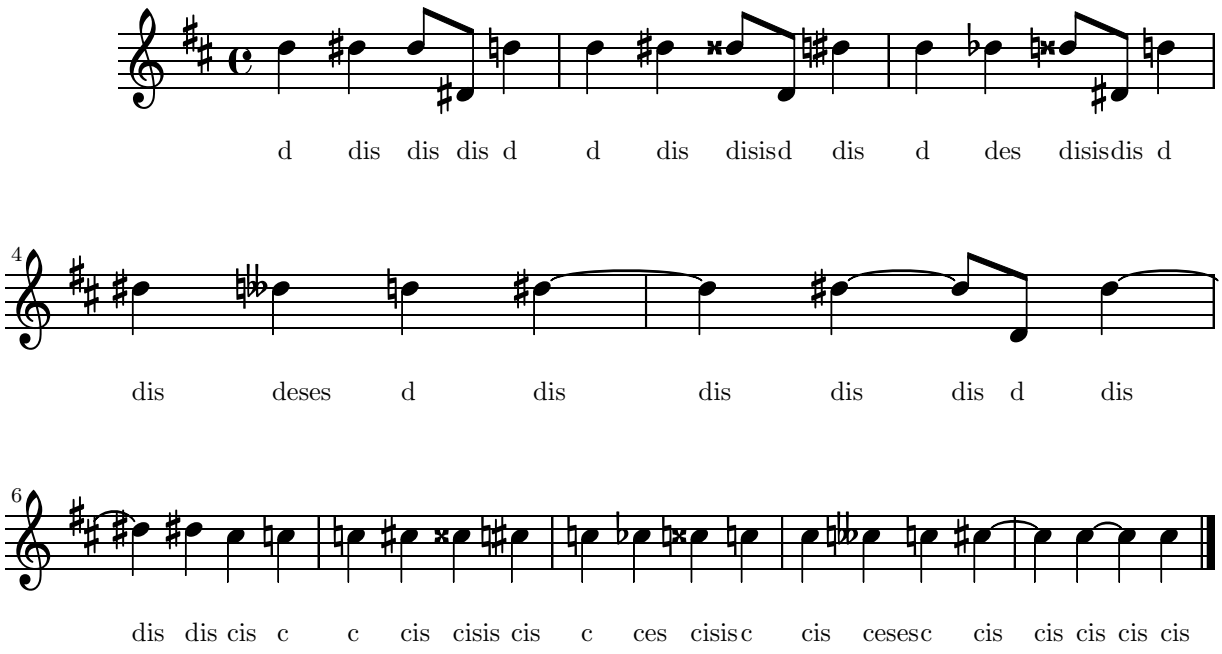
`accidental.ly`

Accidentals work: the second note does not get a sharp. The third and fourth show forced and courtesy accidentals.



`accidentals.ly`

This shows how accidentals are handled.

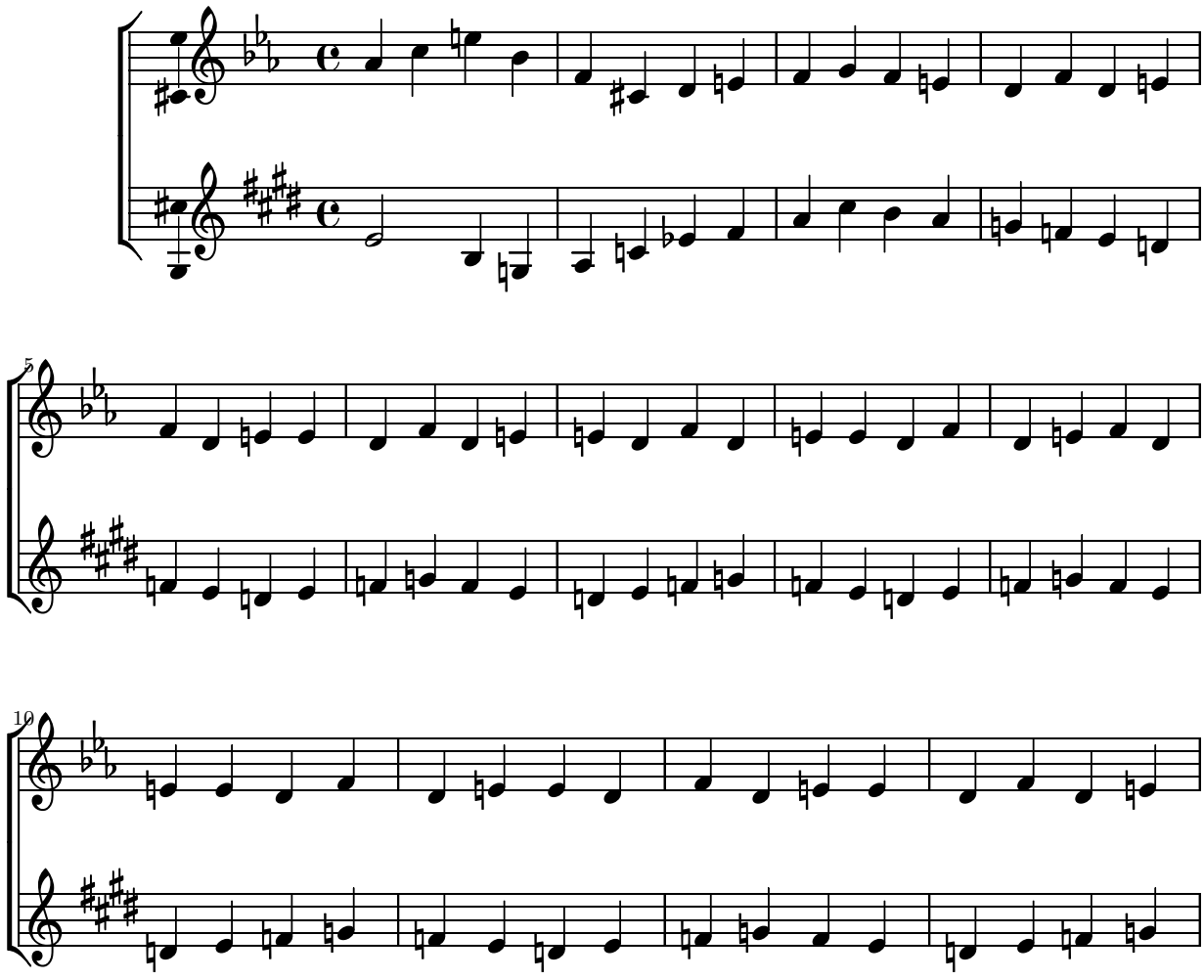


`ambitus.ly`

Ambituses indicate pitch ranges for voices.

By default, the ambitus grob is put before the clef. You can control this behaviour through the `breakAlignOrder` property of the score context by redefining the order.

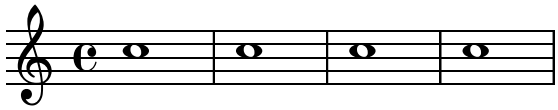
The shape of the note heads to use can be changed via the `note-head-style` property, which holds the glyph name of the note head. The vertical line between the upper and lower head can be switched on or off via the `join-heads` property.



`apply-context.ly`

With `\applycontext`, `\properties` can be modified procedurally. Applications include: checking bar numbers, smart octavation.

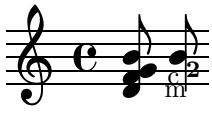
This example prints a bar-number during processing on stdout.



`apply-output.ly`

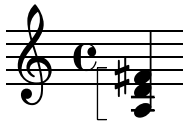
The `\applyoutput` expression is the most flexible way to tune properties for individual grobs.

Here, the layout of a note head is changed depending on its vertical position.



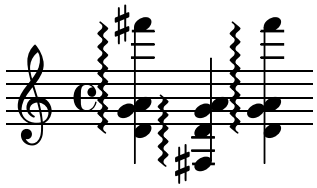
`arpeggio-bracket.ly`

A square bracket on the left indicates that the player should not arpeggiate the chord.



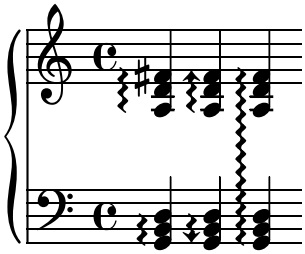
`arpeggio-collision.ly`

Arpeggio stays clear of accidentals and flipped note heads.



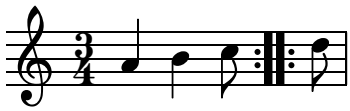
`arpeggio.ly`

Arpeggios are supported, both cross-staff and broken single staff.



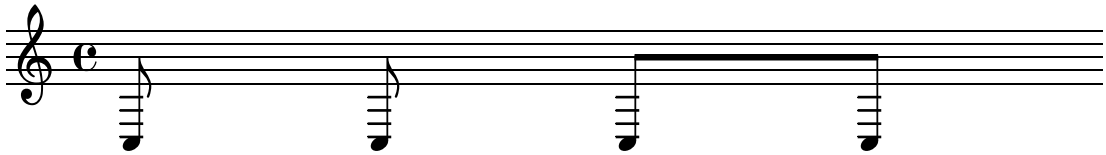
`auto-beam-bar.ly`

No auto beams will be put over (manual) repeat bars.



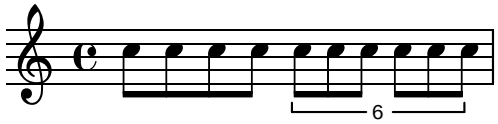
`auto-beam-no-beam.ly`

The autobeamer may be switched off for a single note with `oBeam`.



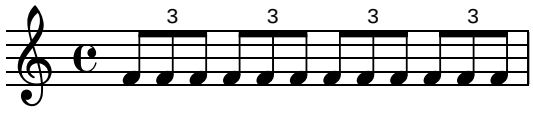
`auto-beam-triplet.ly`

Automatic beaming is also done on tuplets.



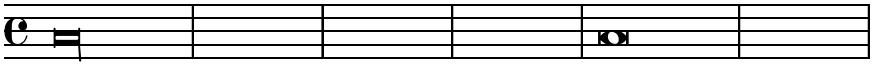
`auto-beam-tuplets.ly`

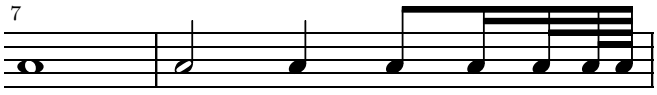
Tuplet-spanner should not put (visible) brackets on beams even if they're auto generated.



`auto-beam.ly`

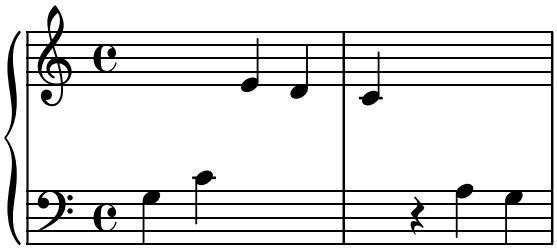
Beams are place automatically; the last measure should have a single beam.





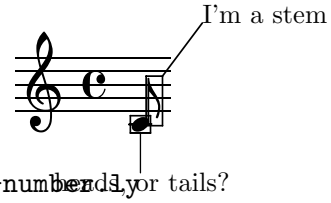
`auto-change.ly`

Auto change piano staff switches voices between up and down staves automatically rests are switched along with the coming note. When central C is reached, staff is not yet switched (by default).



`balloon.ly`

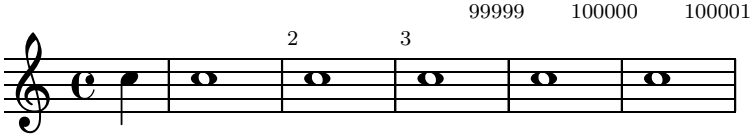
With balloon texts, objects in the output can be marked, with lines and explanatory text added.



`bar-number.ly`

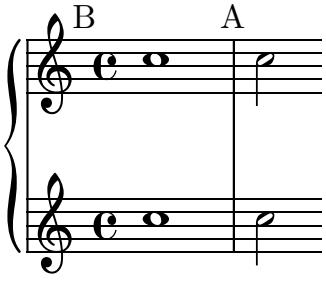
Bar number may be set and their padding adjusted individually. The counting of bar numbers is started after the anacrusis.

To prevent clashes at the beginning of a line, the padding may have to be increased.



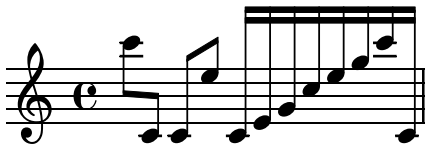
`bar-scripts.ly`

Markings can be attached to (invisible) barlines.



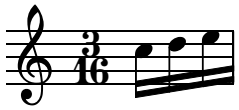
`beam-auto-knee.ly`

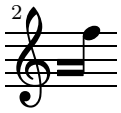
A knee is made automatically when a horizontal beam fits in a gap between note heads that is larger than a predefined threshold.



`beam-break.ly`

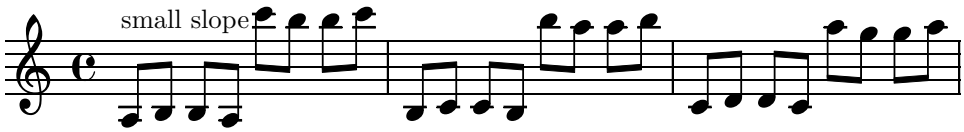
Beams can be printed across line breaks, if forced.

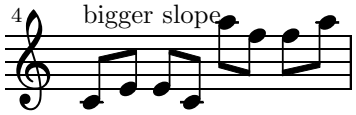




`beam-center-slope.ly`

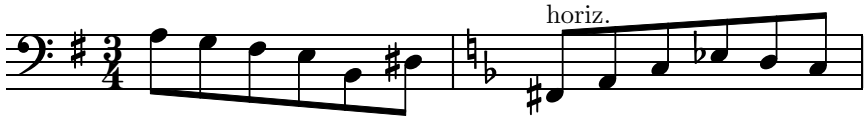
Simple beams on middle staffline are allowed to be slightly sloped, even if the notes have ledgers. Beams reaching beyond middle line can have bigger slope.

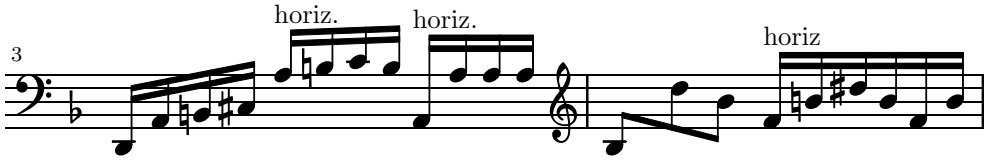




`beam-concave.ly`

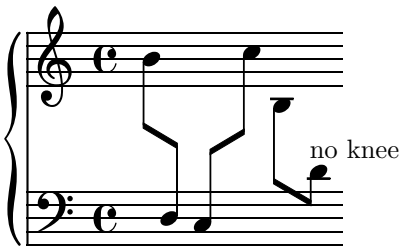
Concave beams should be horizontal. Informally spoken, concave refers to the shape of the notes that are opposite a beam. If an up-beam has high notes on its center stems, then we call it concave. This example shows borderline cases. Only the beams that are marked 'horiz' should be printed horizontally.





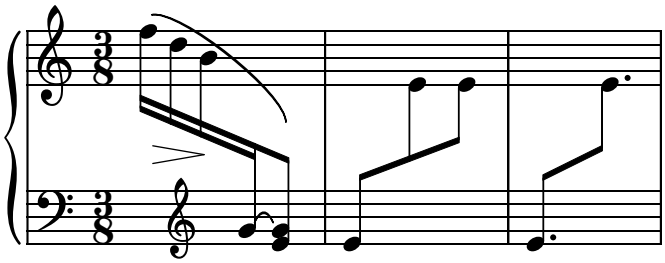
`beam-cross-staff-auto-knee.ly`

Automatic cross-staff knees work also (here they were produced with explicit staff switches).



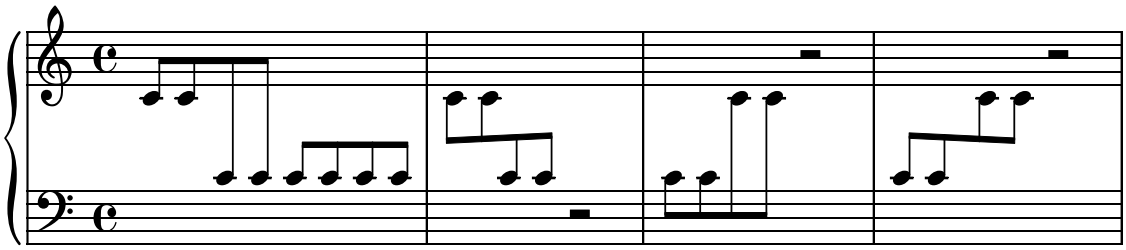
`beam-cross-staff-slope.ly`

Cross staff (kneed) beams do not cause extreme slopes.



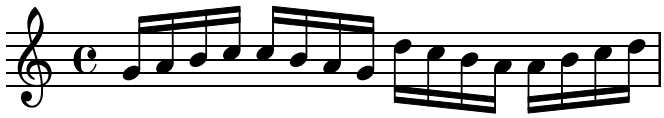
`beam-cross-staff.ly`

Beams can be typeset over fixed distance aligned staves, beam beautification does not really work, but knees do. Beams should be behave well, wherever the switching point is.



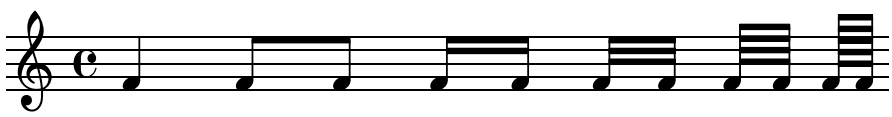
`beam-damp.ly`

Beams are less steep than the notes they encompass.



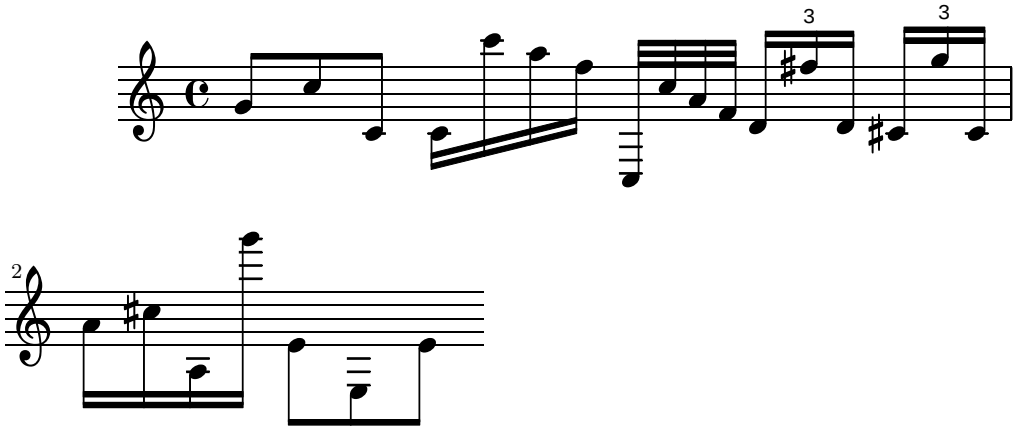
`beam-default-lengths.ly`

Beamed stems have standard lengths if possible. Quantization is switched off in this example.



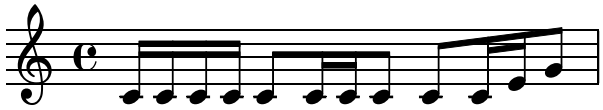
`beam-extreme.ly`

Beams should behave reasonably well, even under extreme circumstances. Stems may be short, but noteheads should never touch the beam. Note that under normal circumstances, these beams would get knees here `Beam.auto-knee-gap` was set to false.



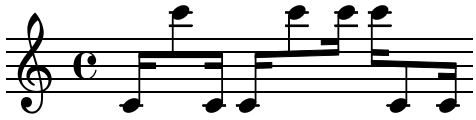
`beam-french.ly`

In french style beaming, the stems do not go between beams.



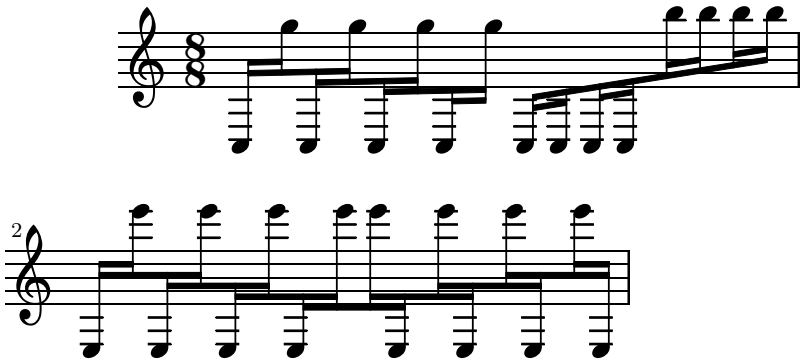
`beam-funky-beamlet.ly`

Funky kneed beams with beamlets also work. The beamlets should be pointing to the note head.



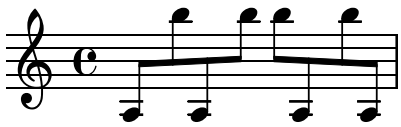
`beam-funky.ly`

In complex configurations of knee beaming, according to Paul Roberts, the first stem of a beam determines the direction of the beam, and as such the way that following (kneed) stems attach to the beam. This is in disagreement with the current algorithm.



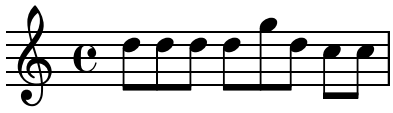
`beam-knee-symmetry.ly`

Point-symmetric beams should receive the same quanting. There is no up/down bias in the quanting code.



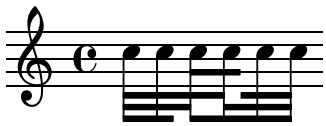
`beam-length.ly`

Beams should look the same.



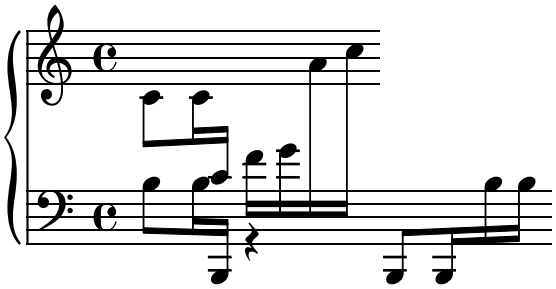
`beam-manual-beaming.ly`

Beaming can be overidden for individual stems.



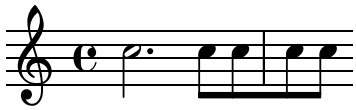
`beam-multiple-cross-staff.ly`

Kneed beams (often happens with cross-staff beams) should look good when there are multiple beams: all the beams should go on continuously at the staff change. Stems in both staves reach up to the last beam.



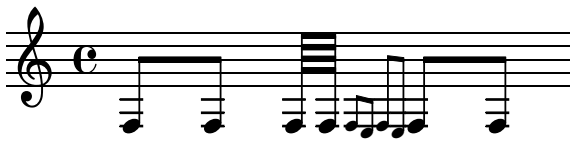
`beam-over-barline.ly`

Explicit beams may cross barlines.



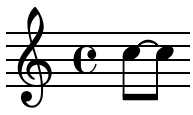
`beam-position.ly`

Beams on ledgered notes should always reach the middle staff line. The second beam counting from the note head side, should never be lower than the second staff line. This does not hold for grace note beams. Override with `no-stem-extend`.



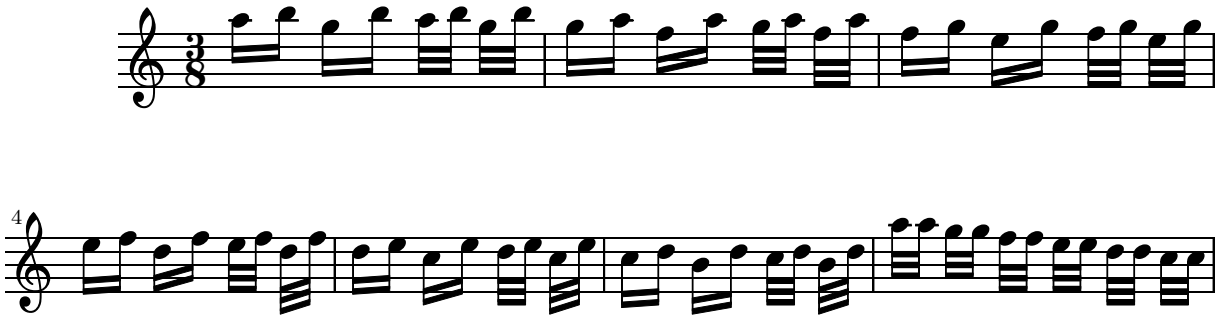
`beam-postfix-notation.ly`

Beams and ties may be entered in postfix notation, separating the notes and the brackets with a dash.



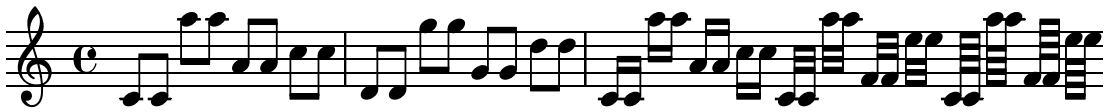
`beam-quanting-32nd.ly`

Stem lengths take precedence over beam quants: 'forbidden' quants are only avoided for 32nd beams when they are outside of the staff. However, that leads to very long stems, which is even worse.



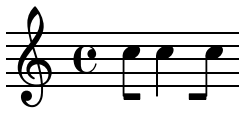
`beam-quanting-horizontal.ly`

In this test for beam quant positions for horizontal beams, staff lines should be covered in all cases. For 32nd beams, the free stem lengths are between 2 and 1.5.



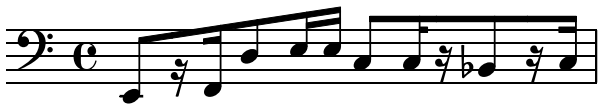
`beam-quarter.ly`

Quarter notes may be beamed: the beam is halted momentarily.



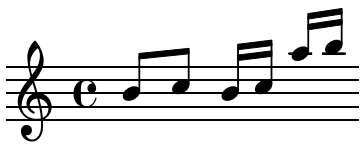
`beam-rest.ly`

The number of beams does not change on a rest.



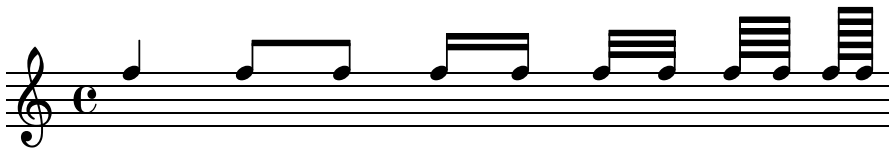
`beam-second.ly`

Engraving second intervals is tricky. We used to have problems with seconds being too steep, or getting too long stems. In a file like this, showing seconds, you'll spot something fishy very quickly.



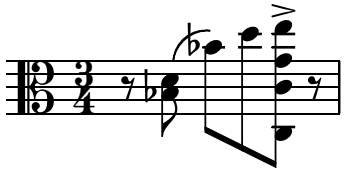
`beam-shortened-lengths.ly`

Beams in unnatural direction, have shortened stems, but do not look too short.



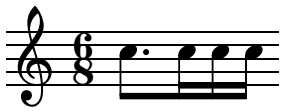
`beamed-chord.ly`

Hairy case for beam, chord, and automatic knees.



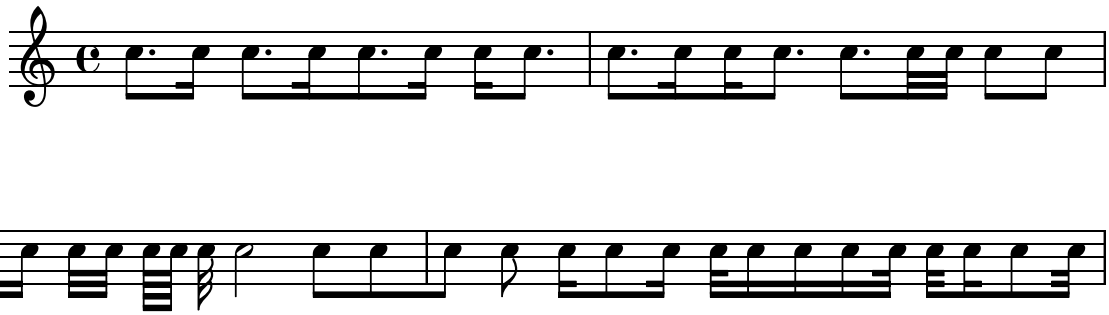
`beaming-ternary-metrum.ly`

Automatic beaming works also in ternary time sigs.



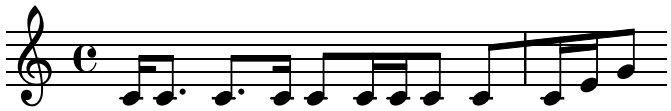
`beaming.ly`

Beaming is generated automatically. Beams may cross bar lines. In that case, line breaks are forbidden. Yet clef and key signatures are hidden just as with breakable bar lines.



`beams.ly`

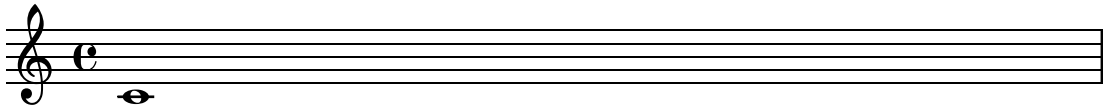
Beaming can be also given explicitly.



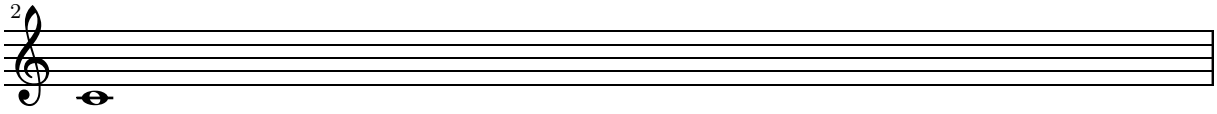
`between-systems.ly`

By inserting TEX commands between systems, you can force pagebreaks.

In reality, you'd use the LateX command `\newpage` instead of (pagebreak) of course.

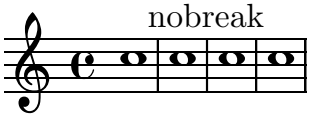


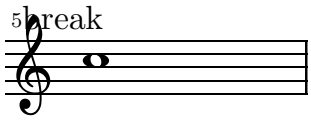
(pagebreak)

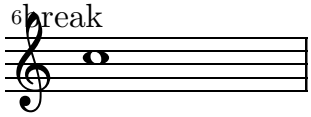


`break.ly`

Breaks can be encouraged and discouraged using `\break` and `\noBreak`.

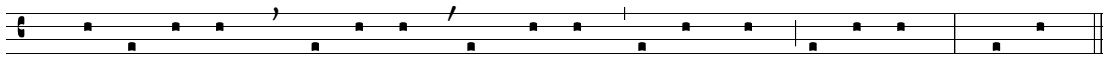






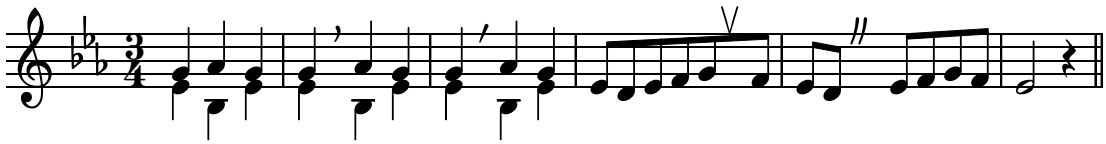
`breathing-sign-ancient.ly`

Gregorian chant notation sometimes also uses commas and ticks, but in smaller font size (we call it 'virgula' and 'caesura'). However, the most common breathing signs are divisio minima/maior/maxima and finalis, the latter three looking similar to bar glyphs.



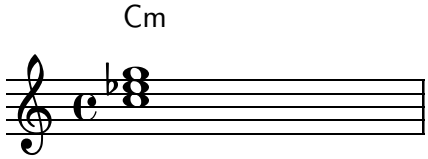
`breathing-sign.ly`

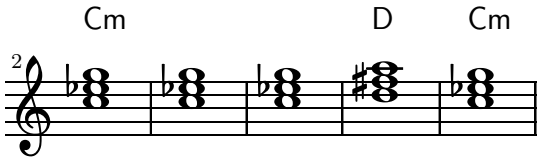
Breathing signs are available in different tastes: commas (default), ticks, vees and 'railroad tracks' (caesura).

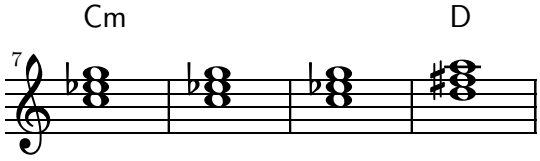


`chord-changes.ly`

Property chordChanges: display chord names only when there's a change in the chords scheme, but always display the chord name after a line break.

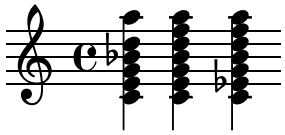






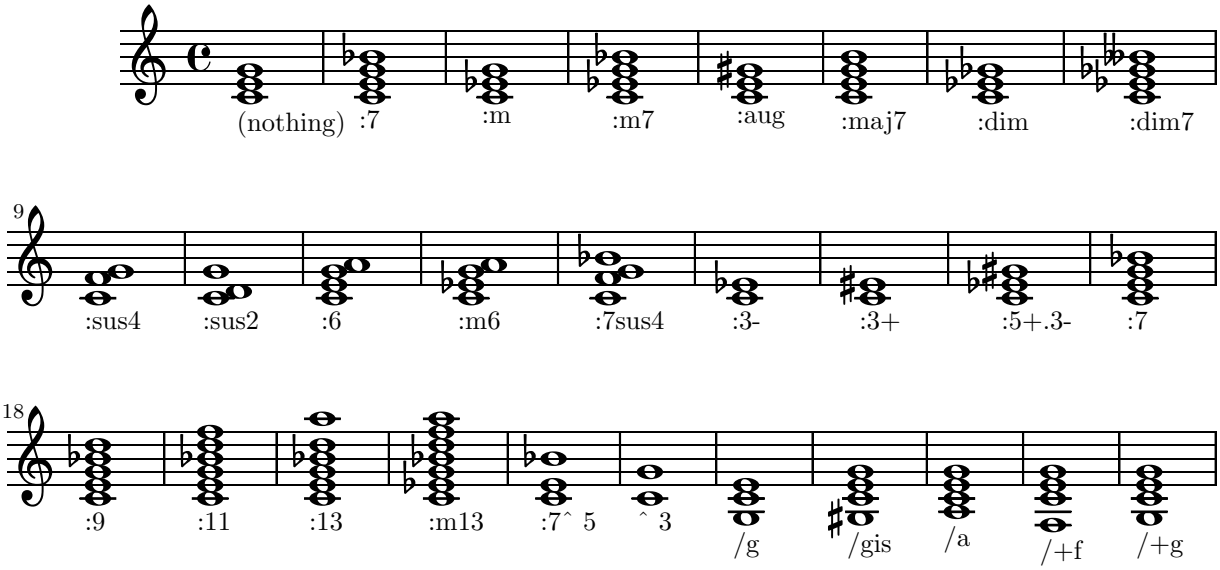
`chord-name-entry-11.ly`

The 11 is only added to major-13 if it is mentioned explicitly.



chord-name-entry.ly

Chords can be produced with the new chordname entry code (`\chords` mode), using a pitch and a suffix. Here, the suffixes are printed below pitches.



chord-name-exceptions.ly

The property `chordNameExceptions` can used to store a list of special notations for specific chords.

Putting the exceptions list encoded as

```
\notes { <c f g bes>1\markup { \super "7" "wahh" } }
```

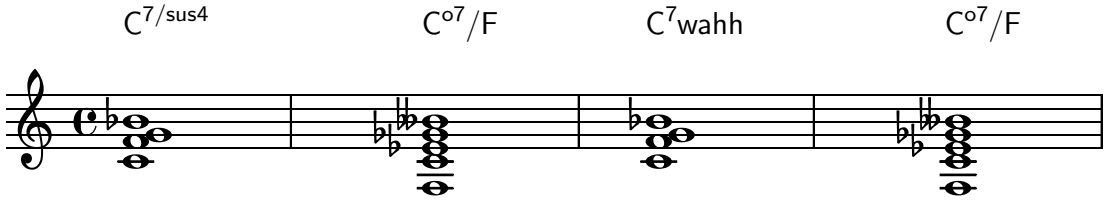
into `chordNameExceptions` takes a little manoeuvring. The following code transforms `chExceptionMusic` (which is a sequential music) into a list of exceptions.

```
(sequential-music-to-chord-exceptions chExceptionMusic \#t)
```

Then,

```
(append
  ... ignatzekExceptions)
```

adds the new exceptions to the default ones, which are defined in '`ly/chord-modifier-init.ly`'.

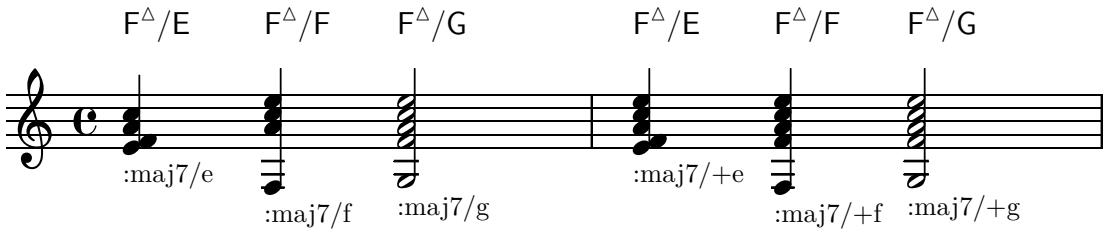


chord-name-major7.ly

The layout of the major 7 can be tuned with `majorSevenSymbol`.

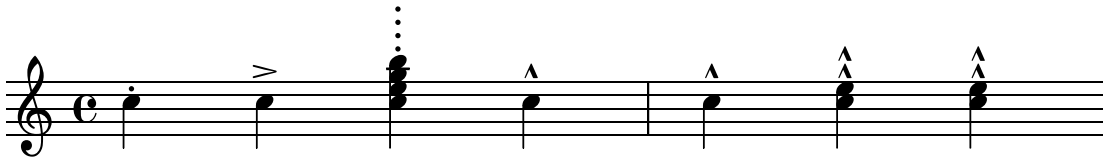
chord-names-bass.ly

In ignatzek inversions, a note is dropped down to act as the bass note of the chord. Bass note may be also added explicitly. Above the staff: computed chord names. Below staff: entered chord name.



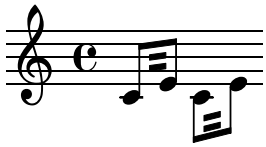
`chord-scripts.ly`

Scripts can also be attached to chord elements.



`chord-tremolo-short.ly`

Tremolo repeats can be constructed for short tremolos (total duration smaller than 1/4) too. Only some of the beams are connected to the stems.

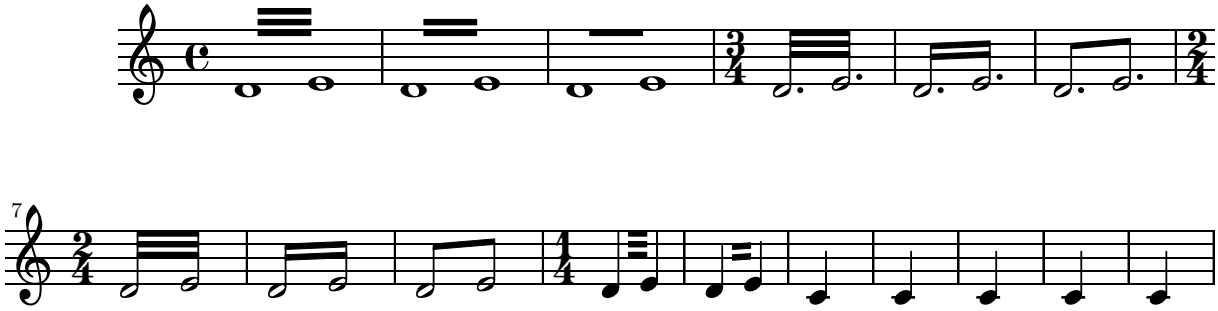


`chord-tremolo.ly`

Chord tremolos look like beams, but are a kind of repeat symbol. To avoid confusion, chord tremolo beams do not reach the stems, but leave a gap. Chord tremolo beams on half notes are not ambiguous, as half notes cannot appear in a regular beam, and should reach the stems.

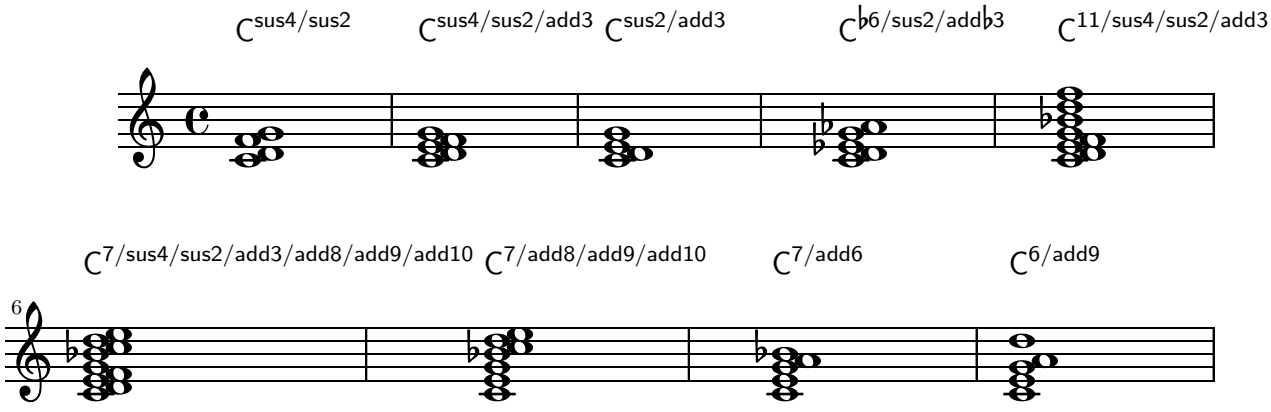
In this example, each tremolo lasts exactly one measure.

(To ensure that the spacing engine is not confused we add some regular notes as well.)



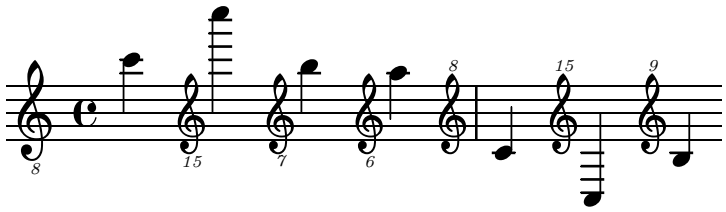
`chords-funky-ignatzek.ly`

Jazz chords may have unusual combinations.



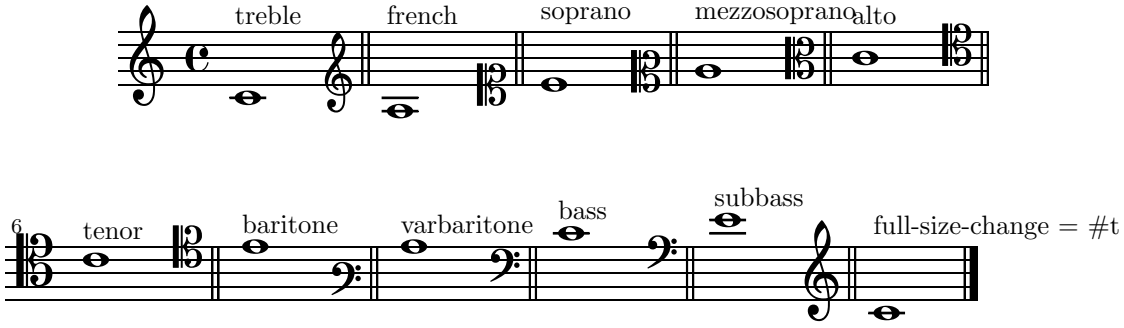
`clef-oct.ly`

Octavation signs may be added to clefs. These octavation signs may be placed below or above (meaning an octave higher or lower), and can take any value, including 15 for two octaves.



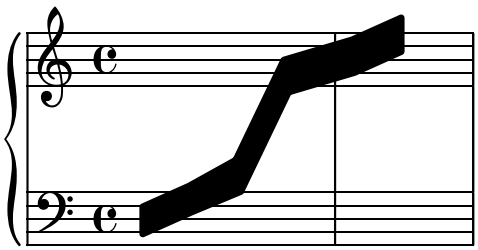
`clefs.ly`

Clefs with `full-size-change` should be typeset in full size. For octaviated clefs, the "8" should appear closely above or below the clef respectively.



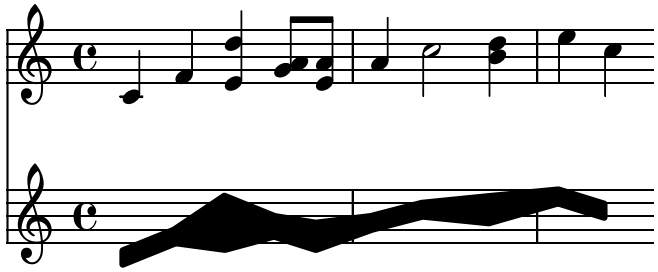
`cluster-cross-staff.ly`

Clusters can be written across staves.



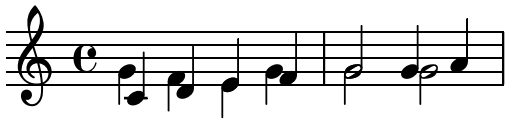
`cluster.ly`

Clusters are a device to denote that a complete range of notes is to be played.



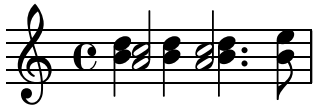
`collision-2.ly`

Single head notes may collide.



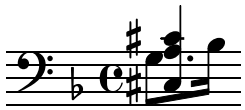
`collision-dots-invert.ly`

When notes are colliding, the resolution depends on the dots: notes with dots should go to the right, if there could be confusion to which notes the dots belong.



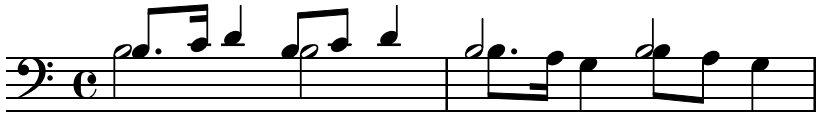
`collision-dots-move.ly`

If collision resolution finds dotted note head must remain on left hand side, move dots to the right.



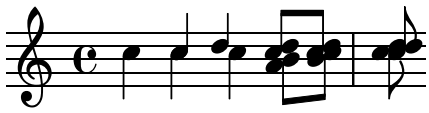
`collision-dots.ly`

Collision resolution tries to put notes with dots on the right side.



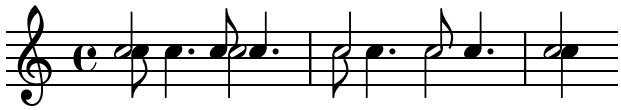
`collision-head-chords.ly`

Note heads in collisions should be merged if they have the same positions in the extreme note heads.



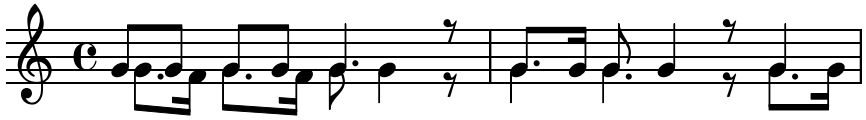
`collision-heads.ly`

If `merge-differently-headed` is enabled, then open note heads may be merged with black noteheads, but only if the black note heads are from 8th or shorter notes.



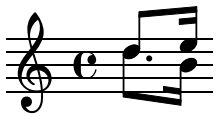
`collision-merge-differently-dotted.ly`

If NoteCollision has merge-differently-dotted = ##t note heads that have differing dot counts may be merged anyway. Dots should not disappear when merging similar note heads.



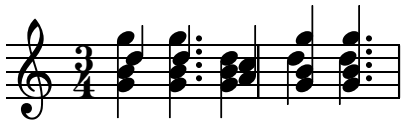
`collision-merge-dots.ly`

When merging heads, the dots are merged too.



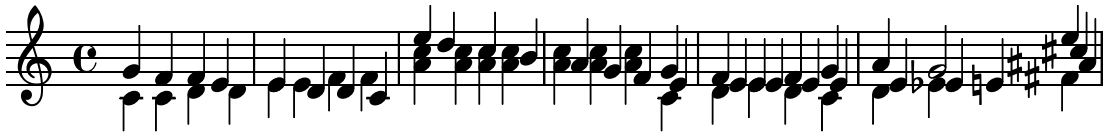
`collision-mesh.ly`

Oppositely stemmed chords, meshing into each other, are resolved.



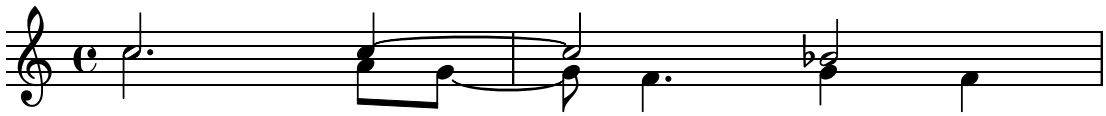
`collisions.ly`

In addition to normal collision rules, there is support for polyphony, where the collosion are avoided by shifting middle voices horizontally.



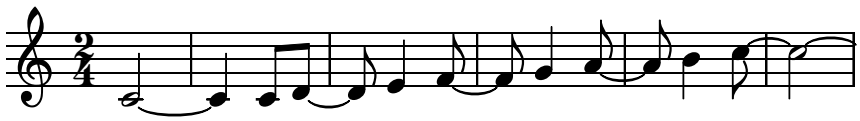
`completion-heads-polyphony.ly`

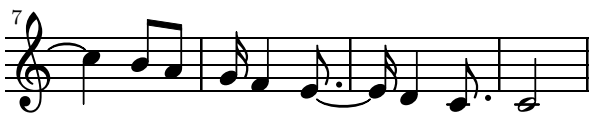
Completion heads are broken across bar lines. This was intended as a debugging tool, but it can be used to ease music entry. Completion heads are not fooled by polyphony with a different rhythm.



`completion-heads.ly`

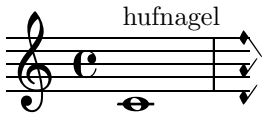
If the `Note_heads_engraver` is replaced by the `Completion_heads_engraver`, notes that cross bar lines are split into tied notes.

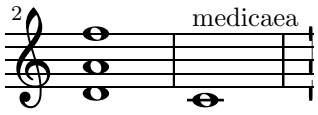


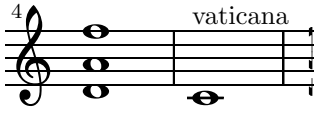


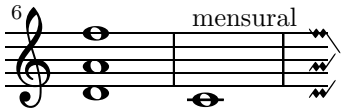
`custos.ly`

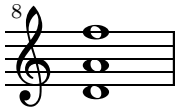
Custodes may be engraved in various styles.





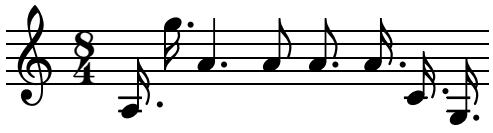






`dot-flag-collision.ly`

Dots move to the right when a collision with the (up)flag happens.

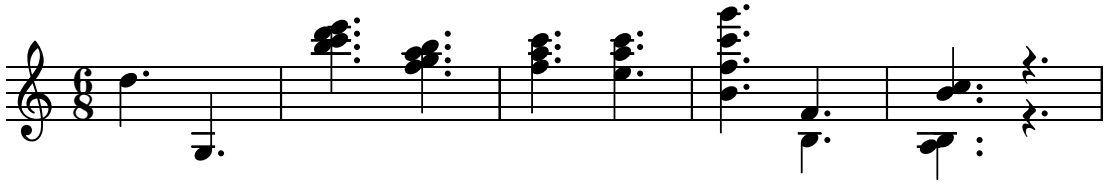


`dots.ly`

Noteheads can have dots, and rests too. Augmentation dots should never be printed on a staff line, but rather be shifted vertically. They should go up, but in case of multiple parts, the down stems have down shifted dots. In case of chords, all dots should be in a column. The dots follow the shift of rests when avoiding collisions.

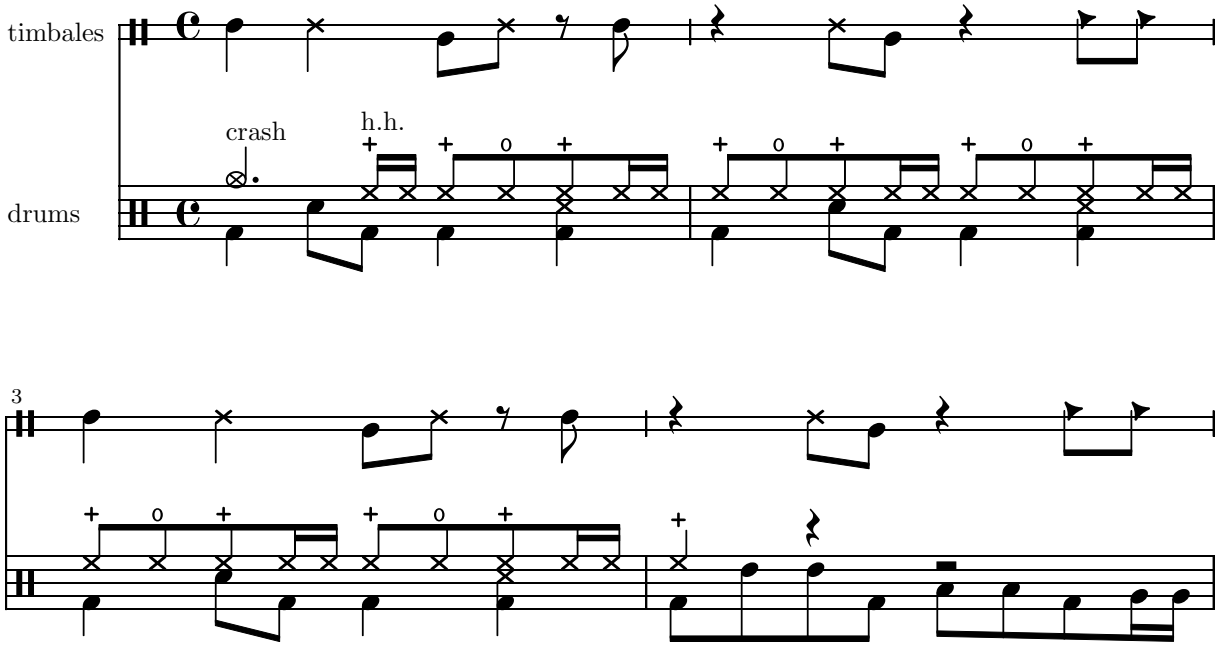
The priorities to print the dots are (ranked in importance):

- keeping dots off staff lines,
- keeping dots close to their note heads,
- moving dots in the direction specified by the voice,
- moving dots up.



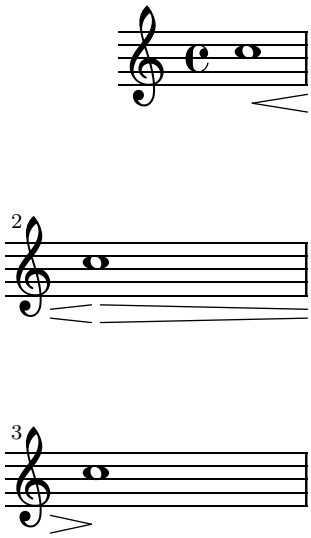
`drums.ly`

In drum notation, there is a special clef symbol, drums are placed to their own staff positions and have note heads according to the drum, an extra symbol may be attached to the drum, and the number of lines may be restricted.



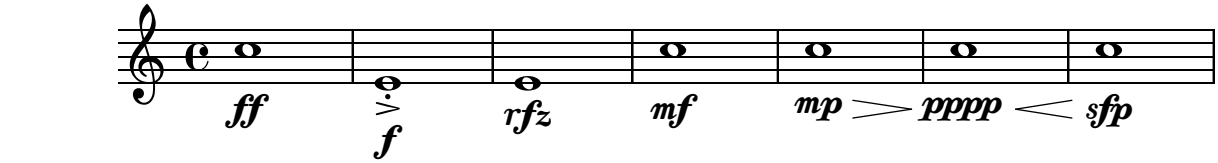
`dynamics-broken-hairpin.ly`

Broken crescendi should be open on one side.



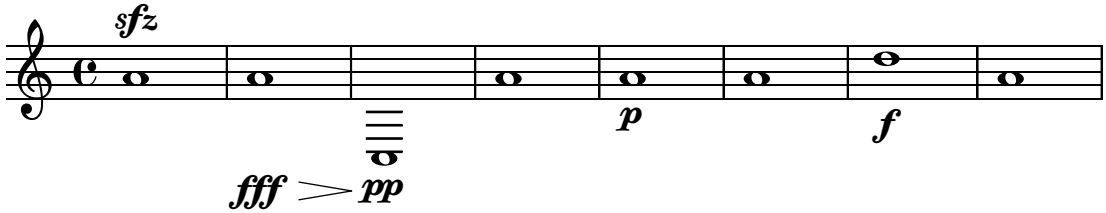
`dynamics-glyphs.ly`

Dynamic letters are kerned, and their weight matches that of the hairpin signs. The dynamic scripts should be horizontally centered on the note head. Scripts that should appear closer to the note head (staccato, accent) are reckoned with.



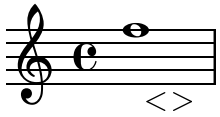
`dynamics-line.ly`

Dynamics appear below or above the staff. If multiple dynamics are linked with (de)crescendi, they should be on the same line. Isolated dynamics may be forced up or down.



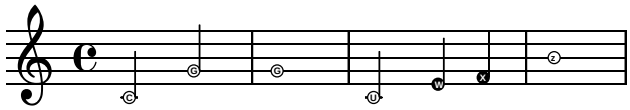
`dynamics-unbound-hairpin.ly`

Crescendi may start off-notes, however, they should not collapse into flat lines.



`easy-notation.ly`

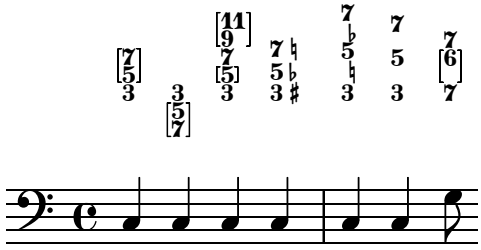
Easy-notation (or Ez-notation) prints names in note heads. You also get ledger lines, of course.



`figured-bass.ly`

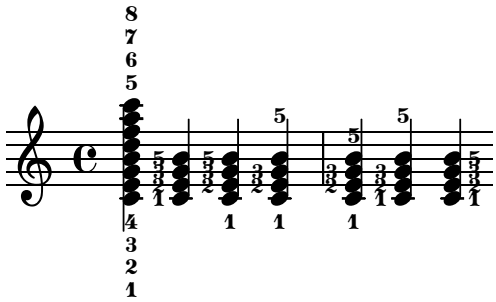
Figured bass is created by the FiguredBass context which eats figured bass requests and rest-requests. You must enter these using the special `\figures { }` mode, which allows you to type numbers, like `<4 6+>`.

You can also type letters by entering quoted strings.



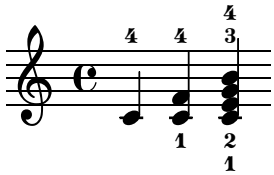
`finger-chords.ly`

With the new chord syntax, it is possible to associate fingerings uniquely with notes. This makes horizontal fingering much easier to process.



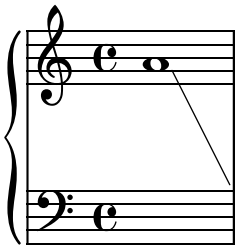
`fingering.ly`

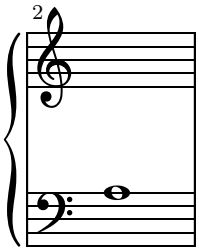
Automatic fingering tries to put fingering instructions next to noteheads.



`follow-voice-break.ly`

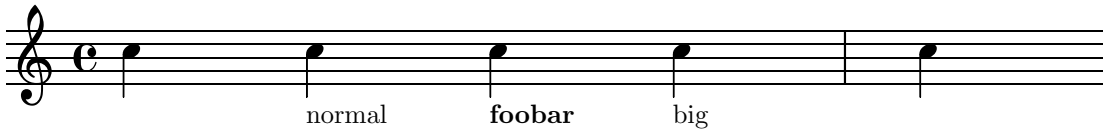
The line-spanners connects to the Y position of the note on the next line. When put across line breaks, only the part before the line break is printed.





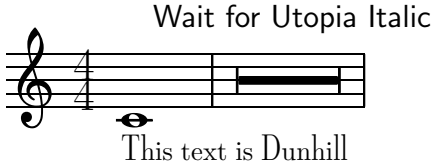
`font-magnification.ly`

The magnification can be set for any font. Note that this does not change variable symbols such as beams or slurs.



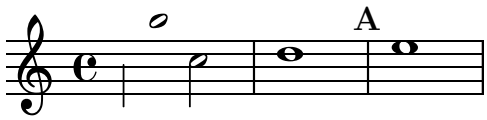
`font-name.ly`

Other fonts can be used by setting font-name for the appropriate object. This may include Postscript fonts that are available through (La)TeX.



`generic-output-property.ly`

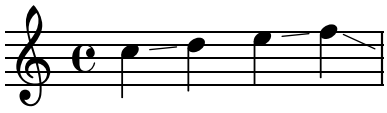
As a last resort, the placement of grobs can be adjusted manually, by setting the `extra-offset` of a grob. A

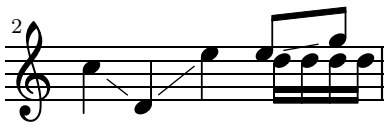


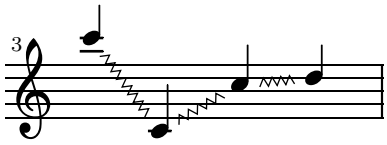
`glissando.ly`

Between notes, there may be simple glissando lines. Here, the first two glissandi are not consecutive.

The engraver does no time-keeping, so it involves some trickery to get `<< { s8 s8 s4 } { c4 \gliss d4 } >>` working correctly.

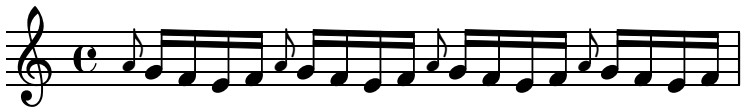






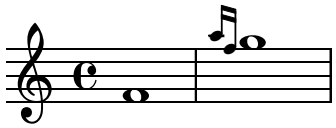
`grace-auto-beam.ly`

The autobeamer is not confused by grace notes.



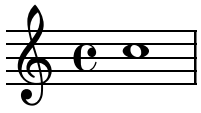
`grace-bar-line.ly`

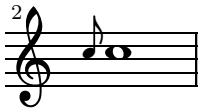
Bar line should come before the grace note.



`grace-bar-number.ly`

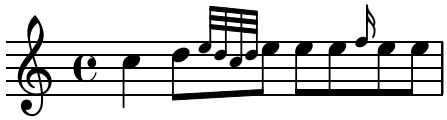
Grace notes do tricky things with timing. If a measure starts with a grace note, the measure does not start at 0, but earlier. Nevertheless, lily should not get confused. For example, line breaks should be possible at grace notes, and the bar number should be printed correctly.





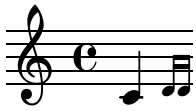
`grace-beam.ly`

Grace beams and normal beams may occur simultaneously. Unbeamed grace notes are not put into normal beams.



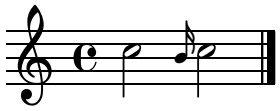
`grace-end.ly`

Grace notes after the last note do not confuse the timing code.



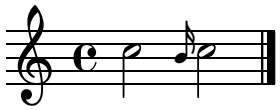
`grace-nest.ly`

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



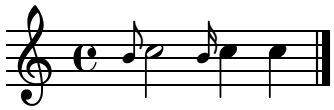
`grace-nest1.ly`

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



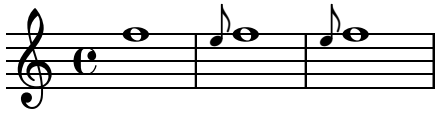
`grace-nest2.ly`

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



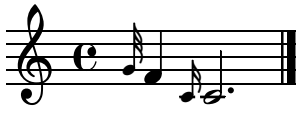
`grace-nest3.ly`

In nested syntax, graces are still properly handled.



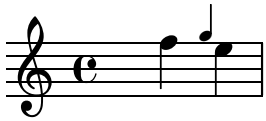
`grace-nest4.ly`

Also in the nested syntax here, grace notes appear rightly.



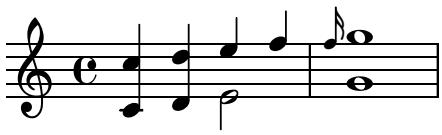
`grace-nest5.ly`

Graces notes may have the same duration as the main note.



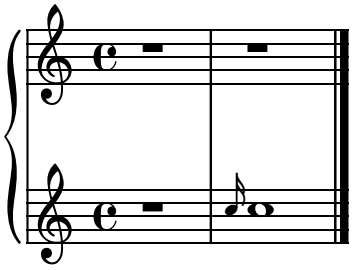
`grace-part-combine.ly`

Grace notes may be put in a `partcombiner`.



`grace-staff-length.ly`

Stripped version of trip.ly. Staves should be of correct length.

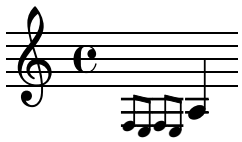


`grace-start.ly`

Pieces may begin with grace notes.

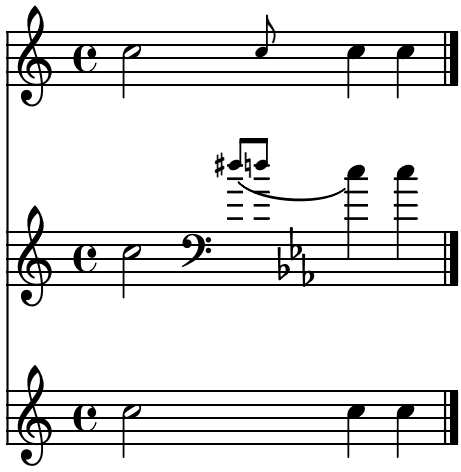
`grace-stems.ly`

Here `startGraceMusic` should set `no-stem-extend` to true; the two grace beams should be the same here.



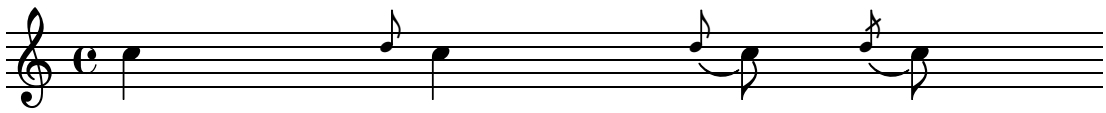
`grace-sync.ly`

Grace notes in different voices/staves are synchronized.



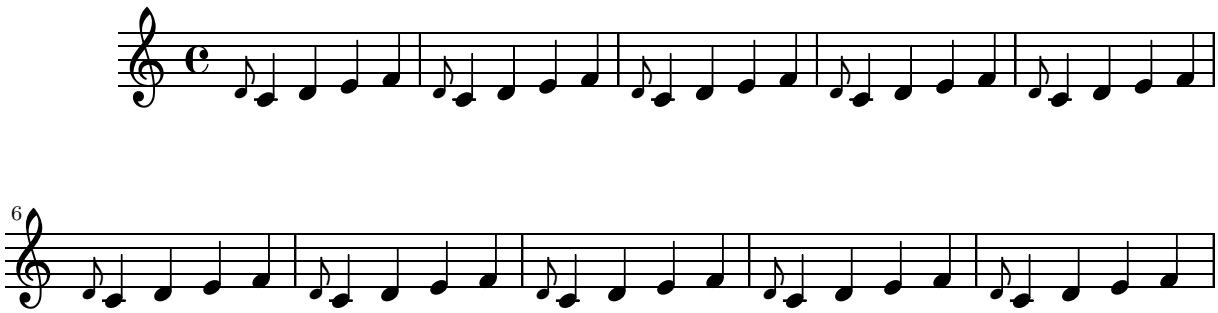
`grace-types.ly`

There are three different kinds of grace types: the base grace switches to smaller type, the appoggiatura inserts also a slur, and the acciaccatura inserts a slur and slashes the stem.



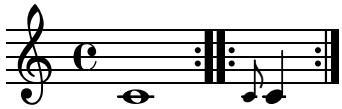
`grace-unfold-repeat.ly`

When grace notes are entered with unfolded repeats, line breaks take place before grace notes.



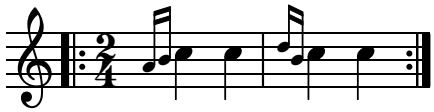
`grace-volta-repeat-2.ly`

A volta repeat may begin with a grace. Consecutive ending and starting repeat bars are into one `:||:`.



`grace-volta-repeat.ly`

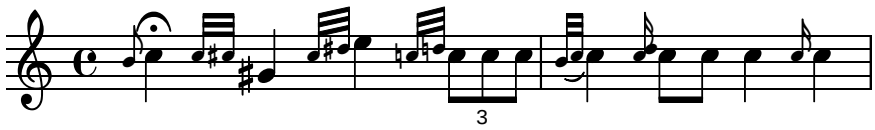
Repeated music can start with grace notes. Bar checks preceding the grace notes do not cause synchronization effects.



`grace.ly`

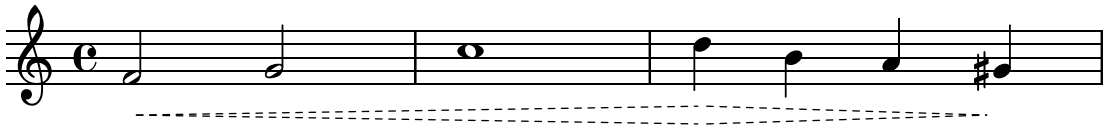
Grace notes are typeset as an encapsulated piece of music. You can have beams, notes, chords, stems etc. within a `\grace` section. Slurs that start within a grace section, but are not ended are attached to the next normal note. Grace notes have zero duration. If there are tuplets, the grace notes will not be under the brace. Grace notes can have accidentals, but they are (currently) spaced at a fixed distance. Grace notes (of course) come before the accidentals of the main note. Grace notes can also be positioned after the main note.

Grace notes without beams should have a slash, if `flagStyle` is not set. Main note scripts do not end up on the grace note.



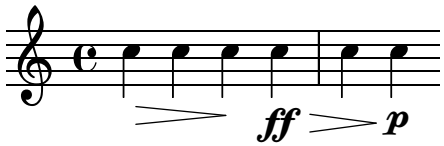
`hairpin-dashed.ly`

Hairpin crescendi may be dashed.



`hairpin-ending.ly`

Hairpin dynamics start under notes if there are no text-dynamics. If there are text dynamics, the hairpin does not run into them.

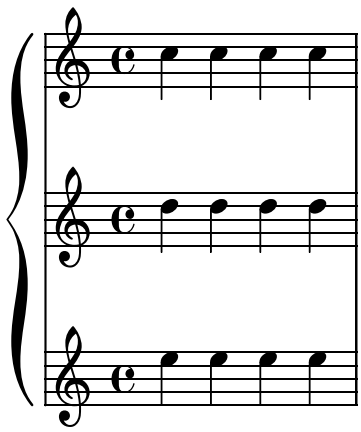


`hara-kiri-pianostaff.ly`

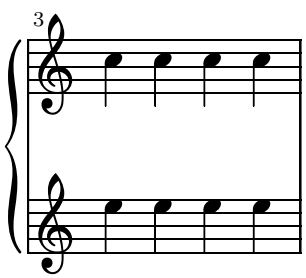
Hara-kiri staves kill themselves if they are empty. This example really contains three staves, but as they progress, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered a bug, however, the scenario of all staves disappearing does not happen in practice.)

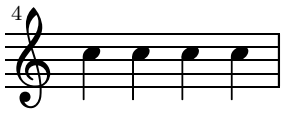
Any staff brackets and braces are removed, both in the single staff and no staff case.

This example was done with a pianostaff, which has fixed distance alignment; this should not confuse the mechanism.



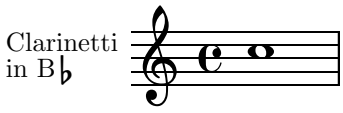
2

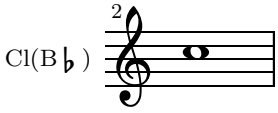




`instrument-name-markup.ly`

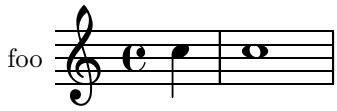
Instrument names are set with `Staff.instrument` and `Staff.instr`. You can enter markup texts to create more funky names, including alterations.





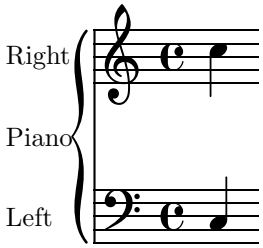
`instrument-name-partial.ly`

Instrument names are also printed on partial starting measures.



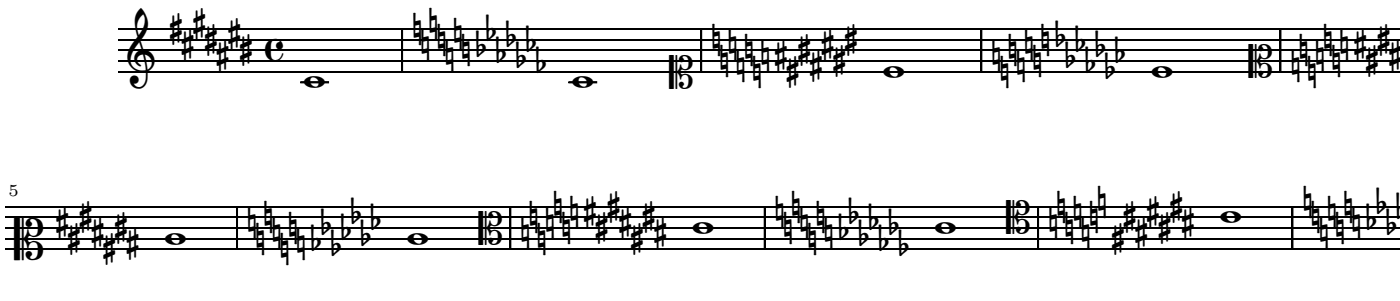
`instrument-name.ly`

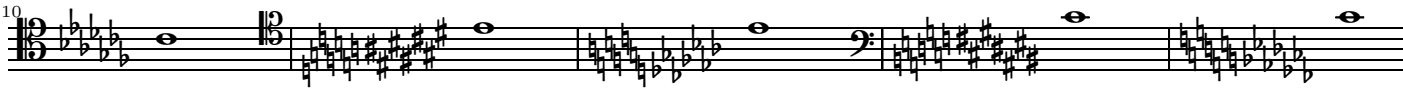
Staff margins are also markings attached to barlines. They should be left of the staff, and be centered vertically with respect to the staff. They may be on normal staves, but also on compound staves, like the PianoStaff.



`key-clefs.ly`

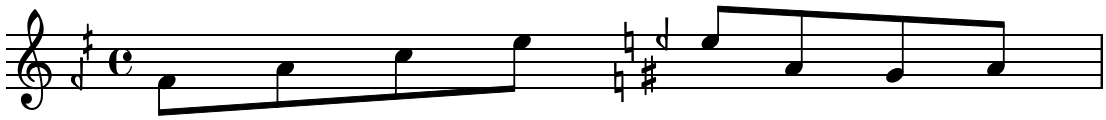
Each clef have own accidental placing rules.





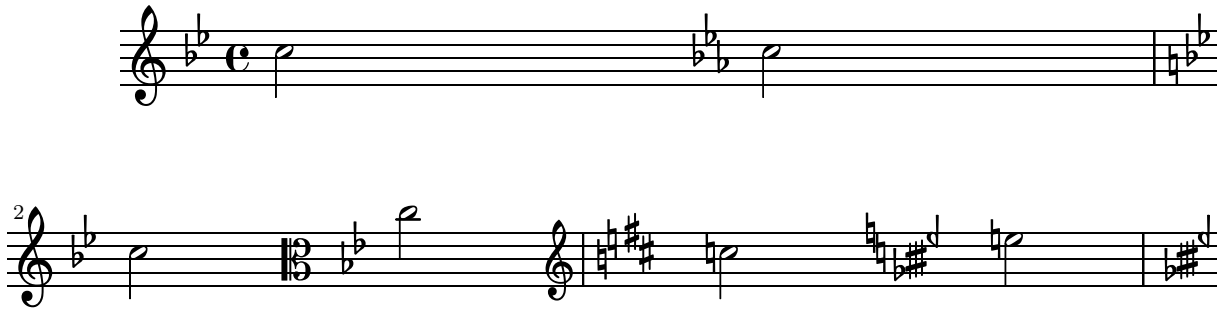
key-signature-scordatura.ly

By setting `Staff.keySignature` directly, key signatures can be set invidually per pitch.



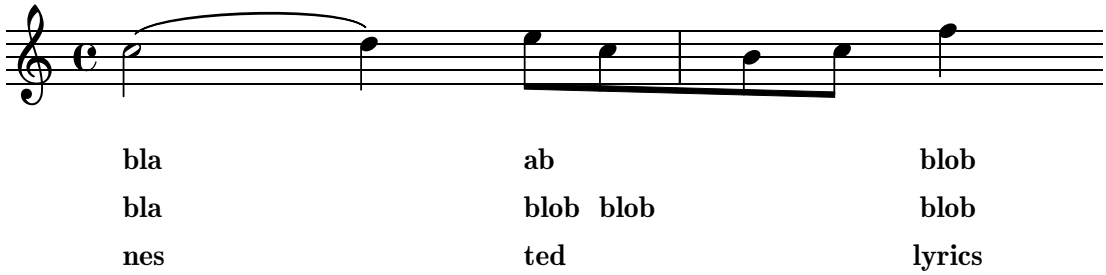
keys.ly

Key signatures may appear on key changes, even without a barline. In the case of a line break, the restoration accidentals are printed at end of a line. If `createKeyOnClefChange` is set, key signatures are created also on a clef change.



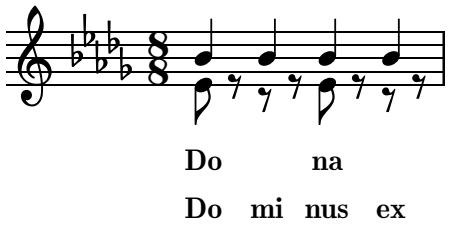
lyric-combine-new.ly

With the `\lyricsto` mechanism, individual lyric lines can be associated with one melody line. For each lyric line, can be tuned whether to follow melismata or not.



lyric-combine-polyphonic.ly

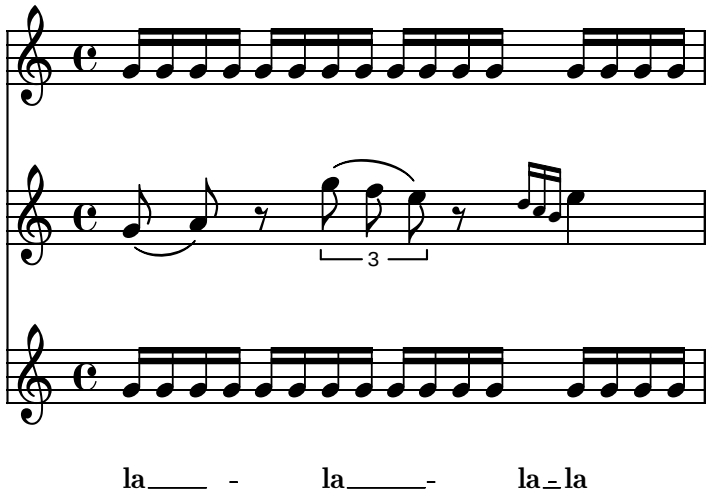
Polyphonic rhythms and rests do not disturb `\lyricsto`.

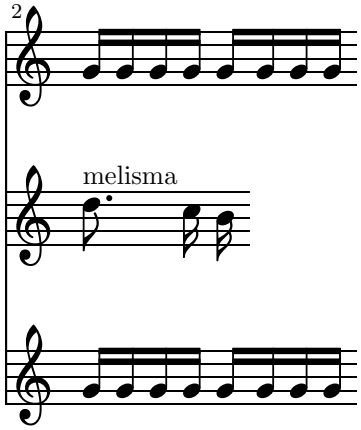


lyric-combine.ly

Lyrics can be set to a melody automatically. Excess lyrics will be discarded. Lyrics will not be set over rests. You can have melismata either by setting a property melismaBusy, or by setting automaticMelismas (which will set melismas during slurs and ties). If you want a different order than first Music, then Lyrics, you must precook a chord of staves/lyrics and label

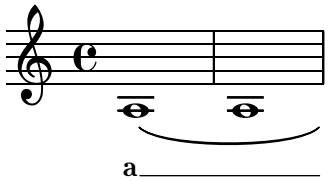
those. Of course, `\rhythm` ignores any other rhythms in the piece. Hyphens and extenders do not assume anything about lyric lengths, so they continue to work.

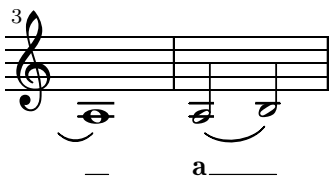


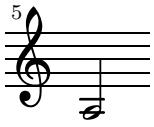


`lyric-extender-broken.ly`

Lyric extenders run to the end of the line if it continues the next line. Otherwise, it should run to the last note of the melisma.



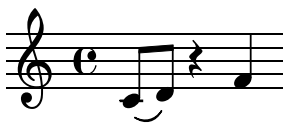




ha

`lyric-extender.ly`

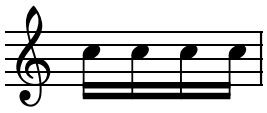
In lyric extenders, a syllable may be extended over several notes.



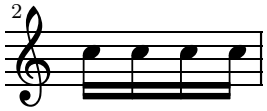
ah_ ha

`lyric-hyphen-break.ly`

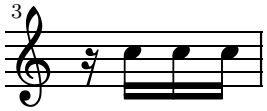
Hyphens are print at the beginning of the line only when they go past the first note.



bla-bla-bla-bla-



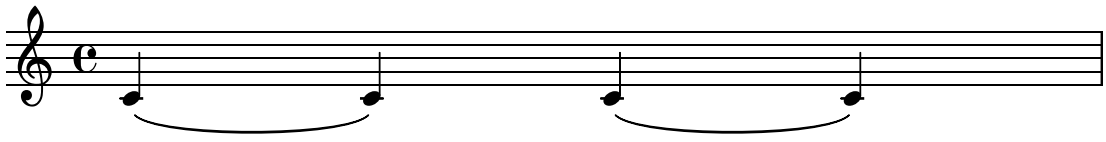
bla-bla-bla-bla-



- bla-bla-bla

`lyric-hyphen.ly`

In lyrics, hyphens may be used.



bla - alb xxx - yyy

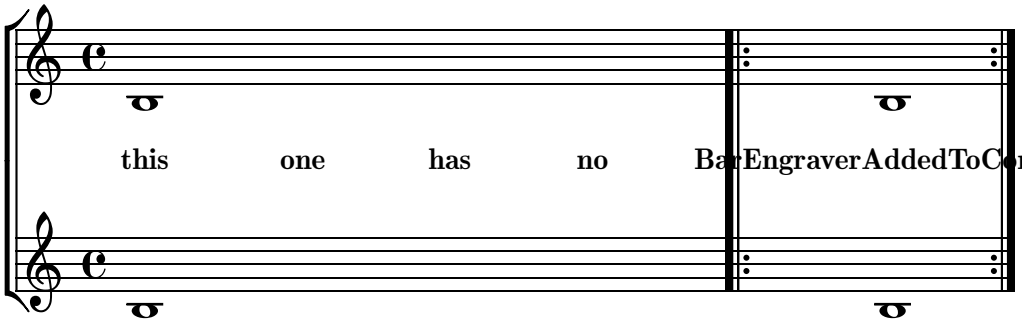
`lyric-phrasing.ly`

Normally, the lyric is centered on the note head. However, on melismata, the text is left aligned on the left-side of the note head.

allllll_____ tijd izzz

lyrics-bar.ly

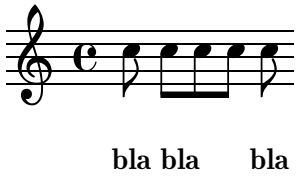
Adding a `Bar_engraver` to the Lyrics context makes sure that lyrics do not collide with barlines.



3

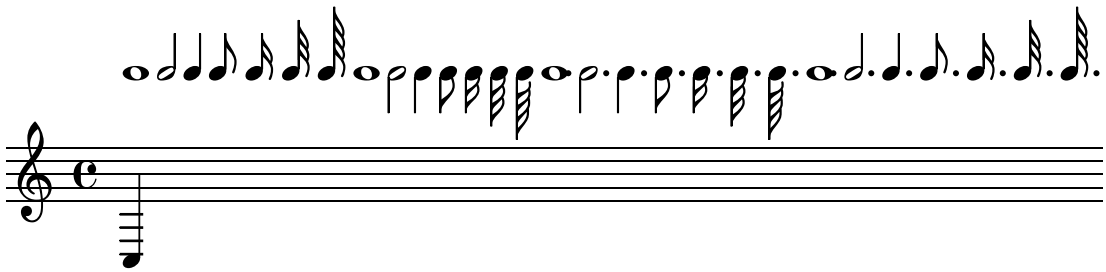
lyrics-melisma-beam.ly

Melismata are triggered by manual beams.



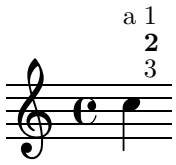
markup-note.ly

The note markup function may be used to make metronome markings. It works for a variety of flag, dot and duration settings.



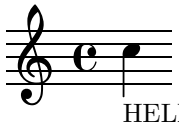
markup-stack.ly

Markup scripts may be stacked.



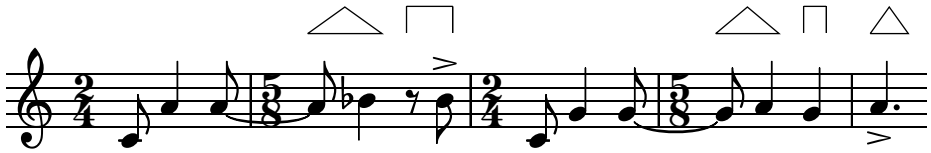
markup-user.ly

Own markup commands may be defined by using the `def-markup-command` scheme macro.



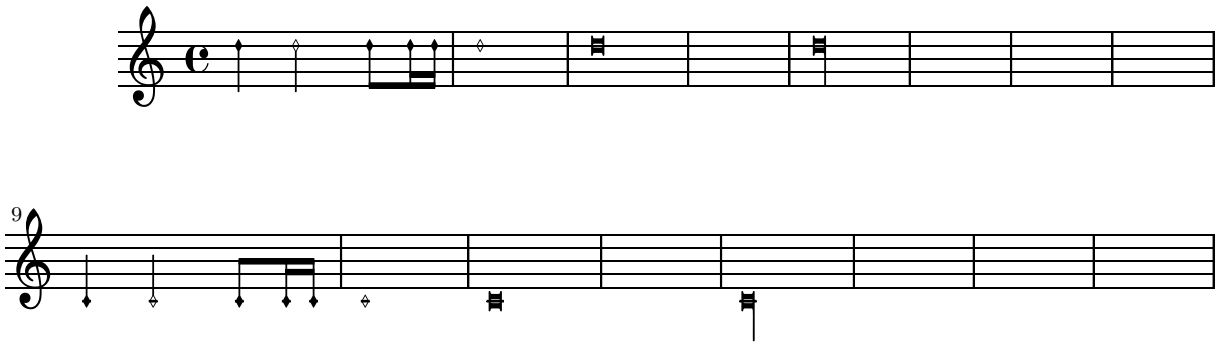
measure-grouping.ly

The `Measure_grouping_engraver` adds triangles and brackets above beats when the beats of a time signature are grouped.



`mensural.ly`

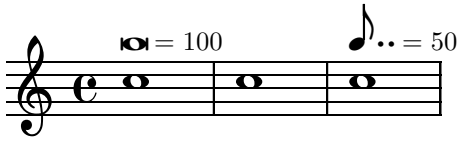
There is limited support for mensural notation: note head shapes are available. Mensural stems are centered on the note heads, both for up and down stems.



`metronome-marking.ly`

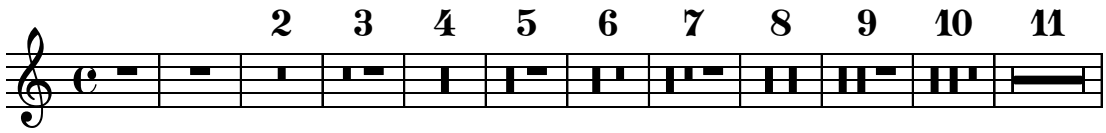
Here `\tempo` directives are printed as metronome markings.

The marking is left aligned with the time signature, if there is one.



`mm-rests2.ly`

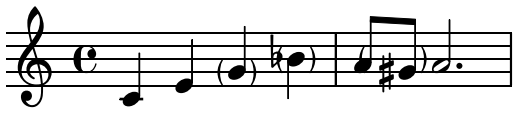
If `Score.skipBars` is set, the signs for four, two, and one measure rest are combined to produce the graphical representation of rests for up to 10 bars. The number of bars will be written above the sign.



`molecule-hacking.ly`

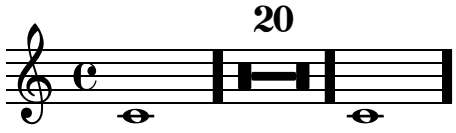
You can write stencil callbacks in Scheme, thus providing custom glyphs for notation elements. A simple example is adding parentheses to existing stencil callbacks.

The parenthesized beam is less successful due to implementation of the Beam. The note head is also rather naive, since the extent of the parens are also not seen by accidentals.



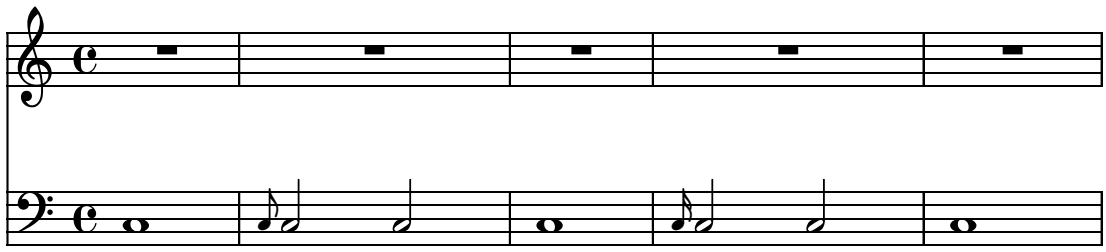
`multi-measure-rest-center.ly`

The multimeasure rest is centered exactly between bar lines.



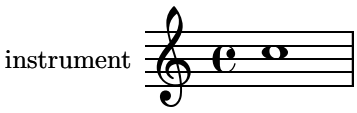
`multi-measure-rest-grace.ly`

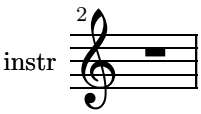
Multi-measure rests are centered also in the case of grace notes.



`multi-measure-rest-instr-name.ly`

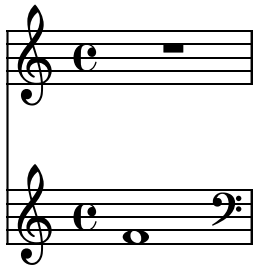
There are both long and short instrument names. Engraving instrument names should not be confused by the multimeasure rests.





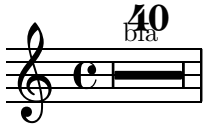
`multi-measure-rest-multi-staff-center.ly`

The centering of multi-measure rests is independent on prefatory matter in other staves.



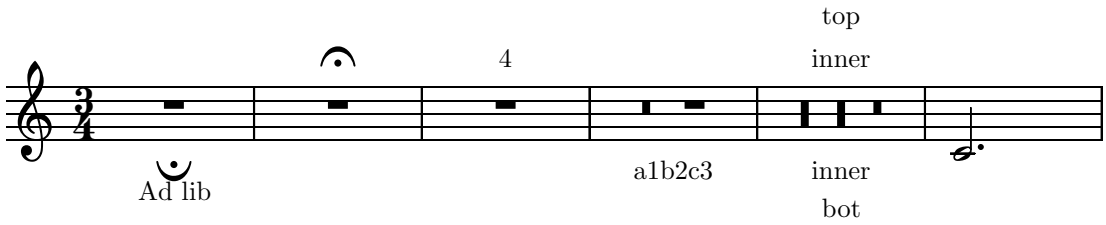
`multi-measure-rest-spacing.ly`

By setting texts starting with a multi-measure rest, an extra spacing column is created. This should not cause problems.



`multi-measure-rest-text.ly`

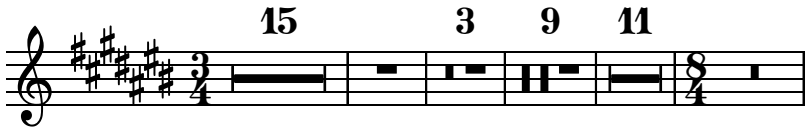
Texts may be added to the multi-measure rests.



`multi-measure-rest.ly`

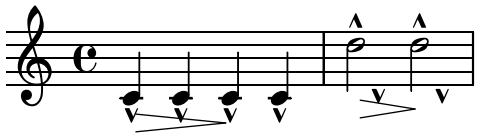
Multi-measure rests do not collide with barlines and clefs. They are not expanded when you set `Score.skipBars`. Although the multi-measure-rest is a Spanner, minimum distances are set to keep it colliding from barlines.

Rests over measures during longer than 2 wholes use breve rests. When more than 10 or more measures (tunable through `expand-limit`) are used then a different symbol is used.



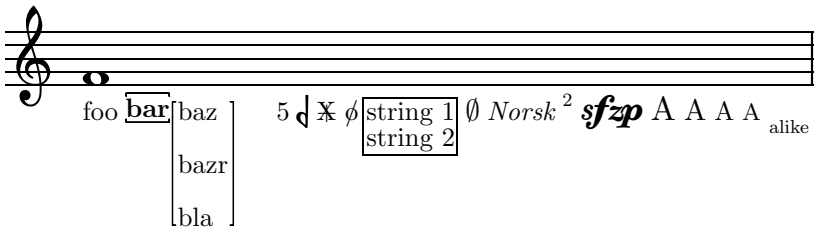
`music-map.ly`

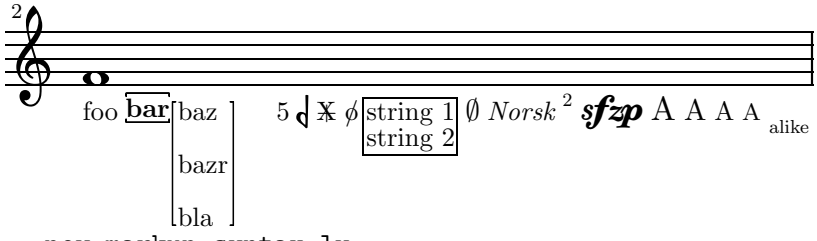
With `music-map`, you can apply functions operating on a single piece of music to an entire music expression. In this example, the the function `notes-to-skip` changes a note to a skip. When applied to an entire music expression in the 1st measure, the scripts and dynamics are left over. These are put onto the 2nd measure.



`new-markup-scheme.ly`

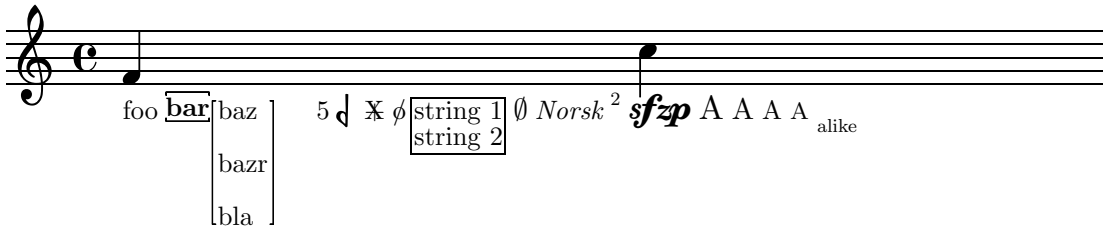
There is a Scheme macro `markup` to produce markup texts using a similar syntax as `\markup`.





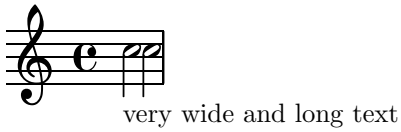
`new-markup-syntax.ly`

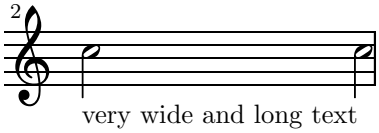
With the new markup syntax, text may be written in various manners.



`non-empty-text.ly`

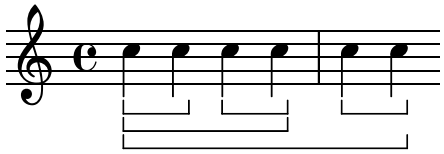
By default, text is set with empty horizontal dimensions. The boolean property `no-spacing-rods` in `TextScript` is used to control the horizontal size of text.





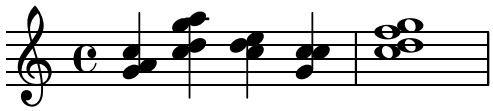
`note-group-bracket.ly`

Note grouping events are used to indicate where analysis brackets start and end.



`note-head-chord.ly`

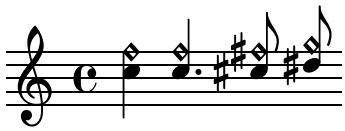
Note heads are flipped on the stem to prevent collisions. It also works for whole heads that have invisible stems.



`note-head-harmonic.ly`

The handling of stems for harmonic notes must be completely identical to normal note heads.

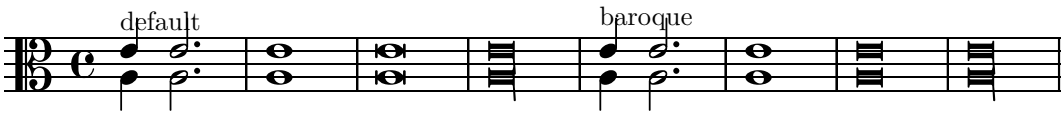
Harmonic heads do not get dots. If `harmonicAccidentals` is unset, they also don't get accidentals.

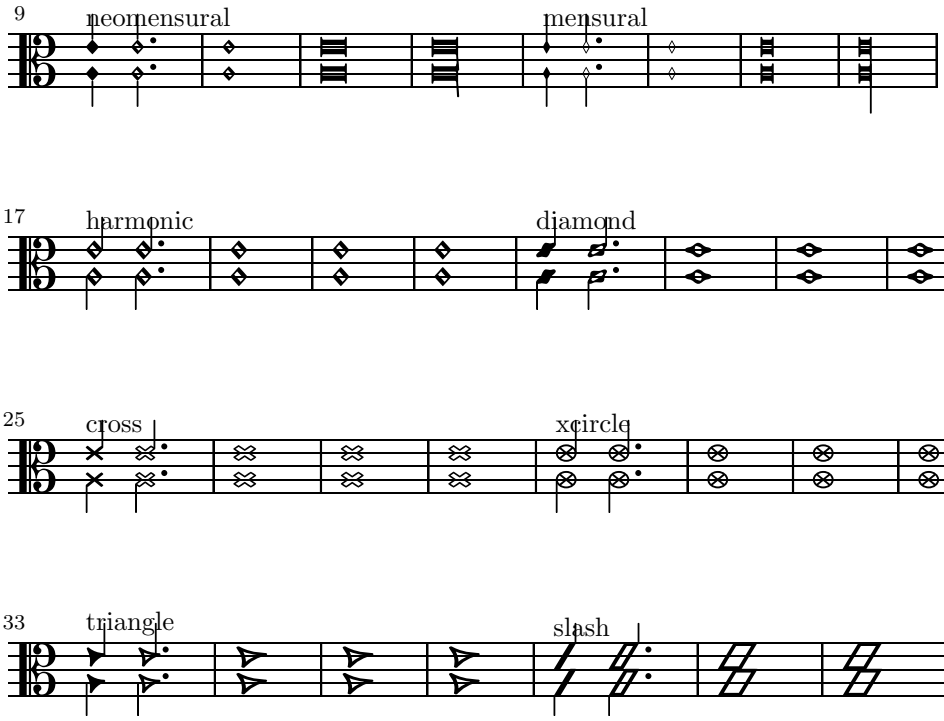


`note-head-style.ly`

Note head shapes may be set from several choices. The stem endings should be adjusted according to the note head. If you want different note head styles on one stem, you must create a special context.

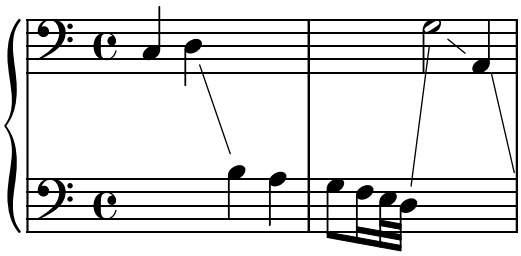
Harmonic notes have a different shape and different dimensions.

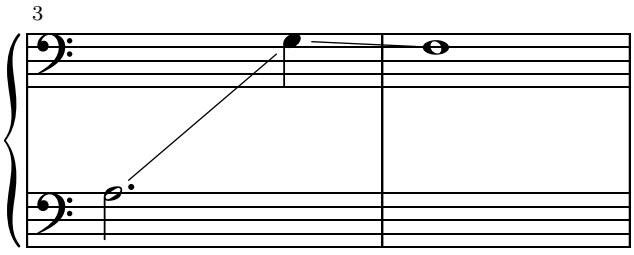




`note-line.ly`

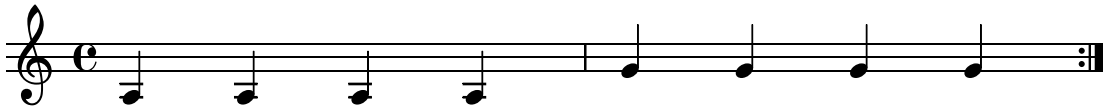
Note head lines (e.g. glissando) run between centers of the note heads.





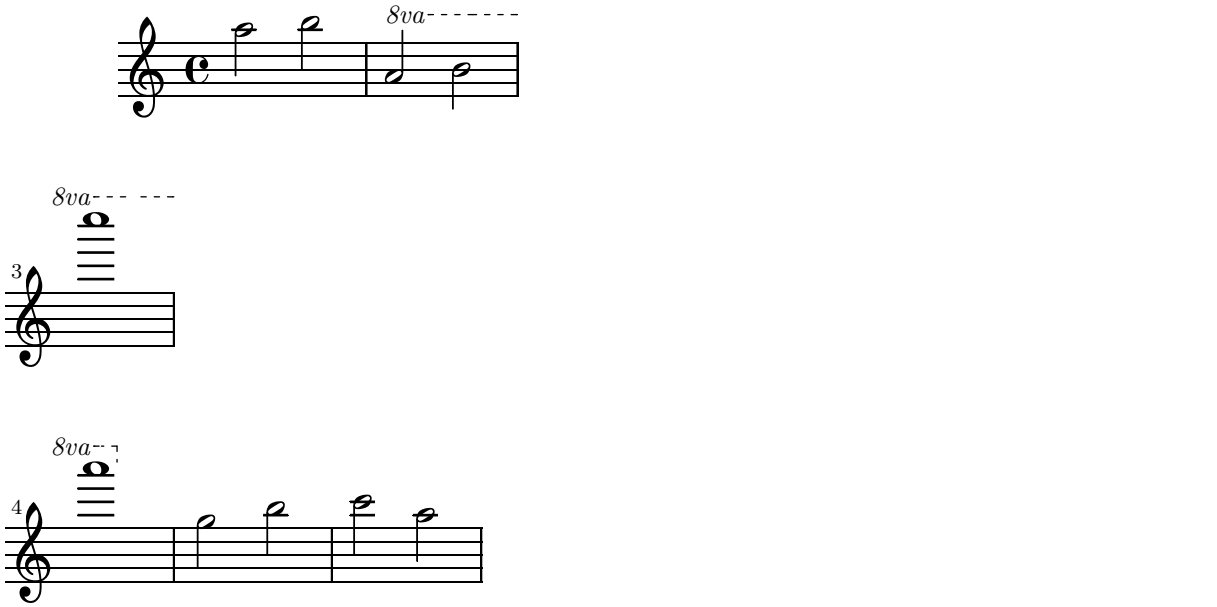
`number-staff-lines.ly`

The number of stafflines of a staff can be set. Ledger lines both on note heads and rests, as well as barlines, are adjusted accordingly.



`ottava-broken.ly`

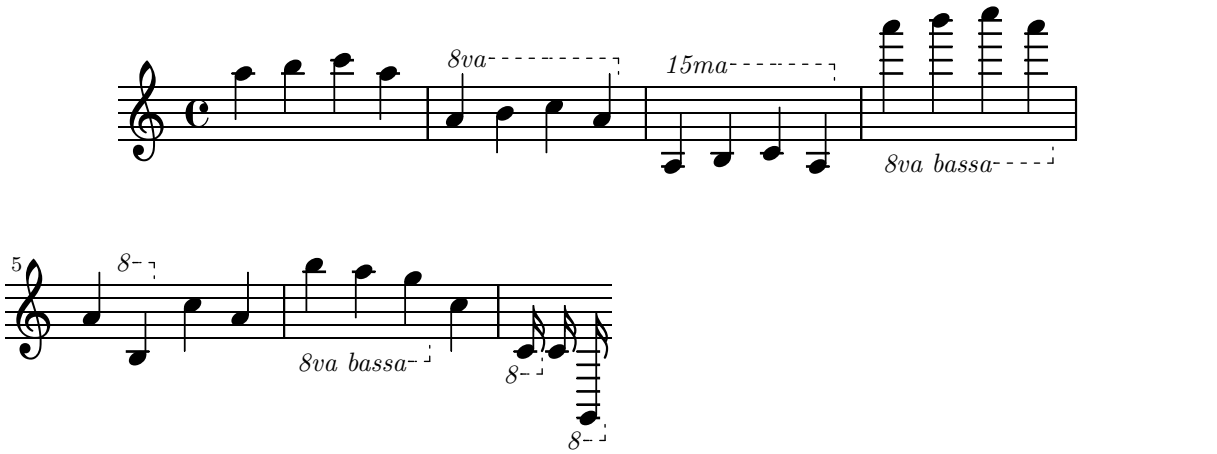
At line breaks, ottava brackets have no vertical line and their horizontal line does not stick out.



`ottava.ly`

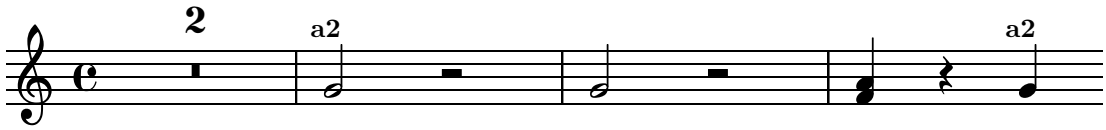
Ottava brackets are supported, through the use of the scheme function `set-octavation`.

The spanner should go below a staff for 8va bassa, and the ottavation string can be tuned with `Staff.ottavation`.



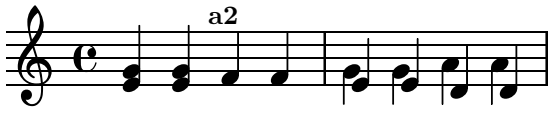
`part-combine-a2.ly`

The a2 string is printed only on notes (i.e. not on rests), and only after chords, solo or polyphony.



`part-combine-cross.ly`

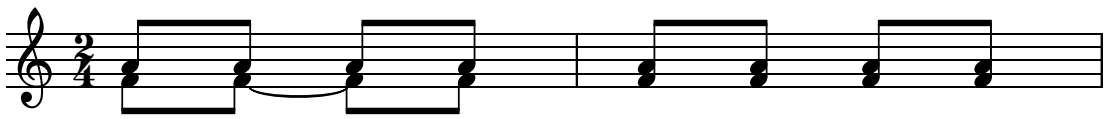
The part combiner stays apart for crossing voices.



`part-combine-global.ly`

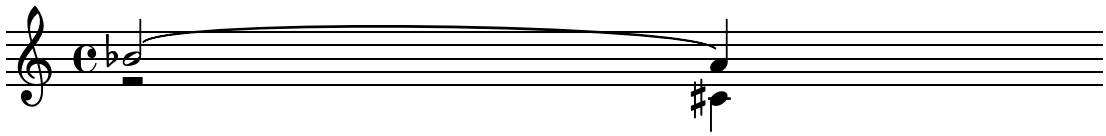
The analysis of the part combiner is non-local: in the following example, the decision for using separate voices in the 1st measure is made on the 2nd note, but influences the 1st note.

In the 2nd measure, the pattern without the tie, leads to combined voices.



`part-combine-solo-global.ly`

In this example, solo1 should not be printed over the 1st note, because of the slur which is present from the one-voice to the two-voice situation.

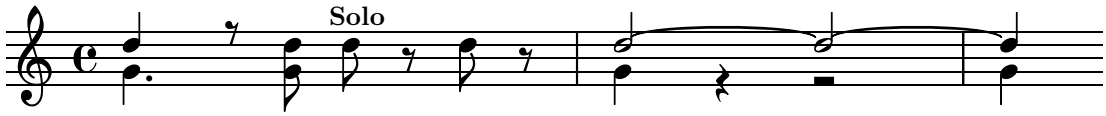


`part-combine-solo.ly`

A solo string can only be printed when a note starts. Hence, in this example, there is no Solo-2 although the 2nd voice has a dotted quarter, while the first voice has a rest.

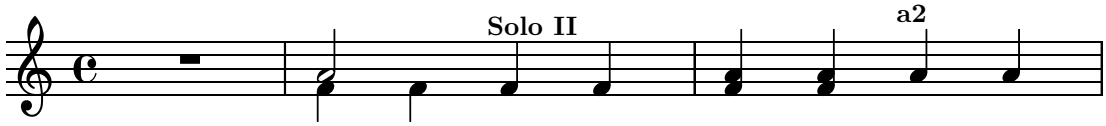
A Solo indication is only printed once; (shared) rests do not require reprinting a solo indication.

Solo 1/2 can not be used when a spanner is active, so there is no solo over any of the tied notes.



`part-combine-text.ly`

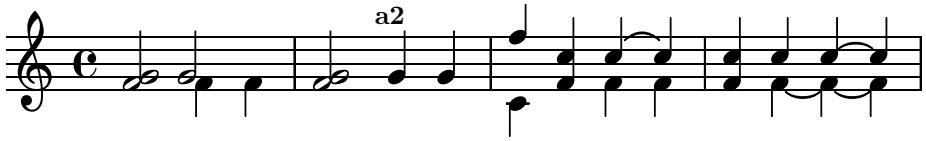
The new part combiner detects a2, solo1 and solo2, and prints i texts accordingly.

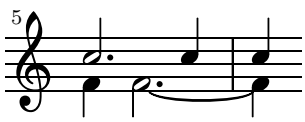


`part-combine.ly`

The new part combiner stays apart from:

- different durations,
- different articulations (taking into account only slur/beam/tie), and
- wide pitch ranges.

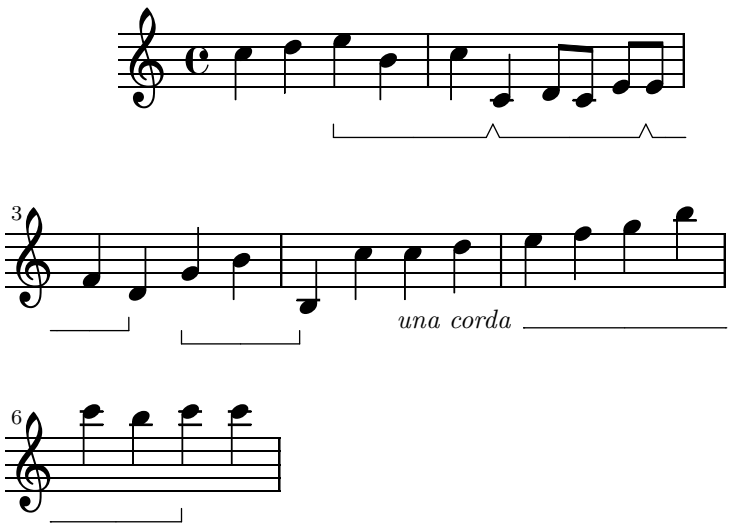




`pedal-bracket.ly`

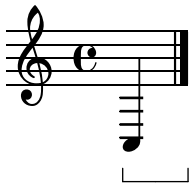
The brackets of a piano pedal should start and end at the left side of the note. If a note is shared between two brackets, these ends are flared.

At a line-break, there are no vertical endings.



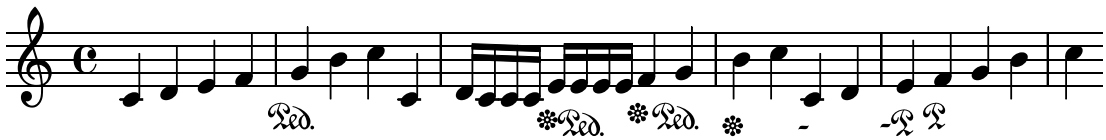
`pedal-end.ly`

Unterminated piano pedal brackets run to the end of the piece.



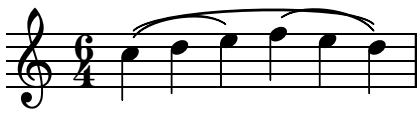
`pedal-ped.ly`

The standard piano pedals style comes with Ped symbols. The pedal string can be also tuned, for example, to a shorter tilde/P variant at the end of the melody.



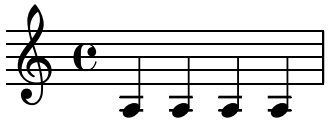
`phrasing-slur.ly`

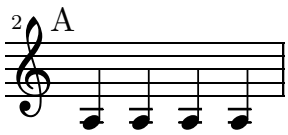
Ordinary slurs should work well with phrasing slur.



`prefatory-empty-spacing.ly`

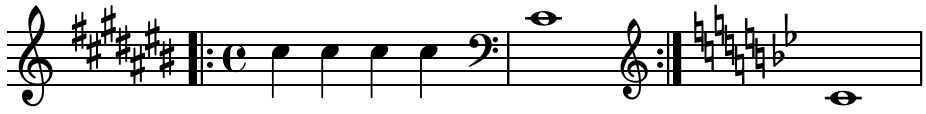
The A is atop an invisible barline. The barline, although invisible, is also translated because it is the last one of the break alignment.





`prefatory-spacing-matter.ly`

Distances between prefatory items (e.g. clef, bar, etc.) are determined by engraving standards. These distances depend on which items are combined. Mid-line, the order for clef and bar-line is different from the start of line.

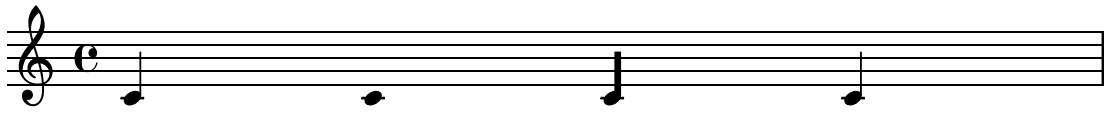


`property-grace-polyphony.ly`

Property overrides and reverts from `\grace` do not interfere with the overrides and reverts from polyphony.

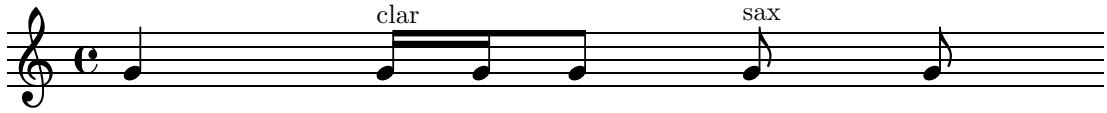
`property-once.ly`

Once properties take effect during a single time step only.



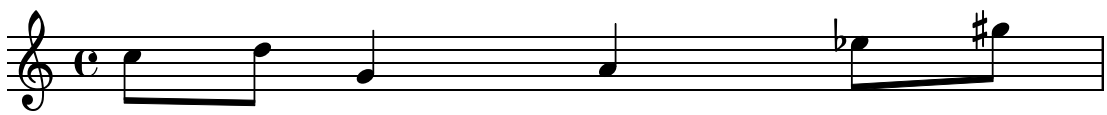
`quote-transposition.ly`

Quotations take into account the transposition of both source and target. In this example, all instructions play sounding central C, the target is a instrument in F.



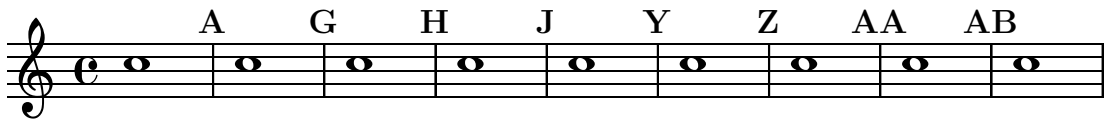
`quote.ly`

With `\quote`, fragments of previously entered music may be quoted.



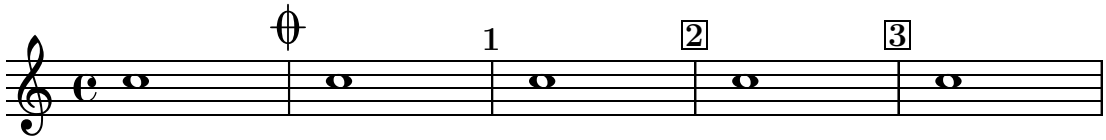
`rehearsal-mark-letter.ly`

Rehearsal marks in letter style: the I is skipped, and after Z, double letters are used. The mark may be set with `\mark NUMBER`, or with `Score.rehearsalMark`.



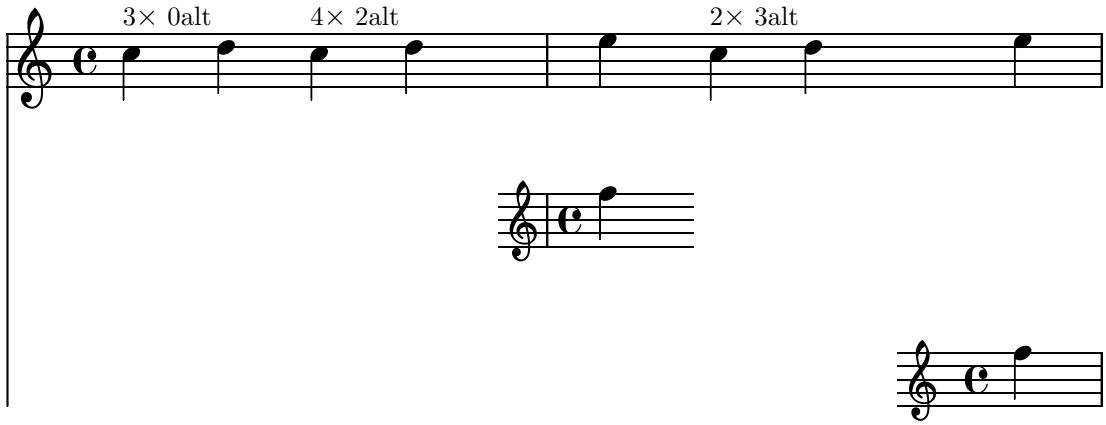
`rehearsal-mark-number.ly`

Marks can be printed as numbers. By setting `markFormatter` we may choose a different style of mark printing. Also, marks can be specified manually, with a markup argument.



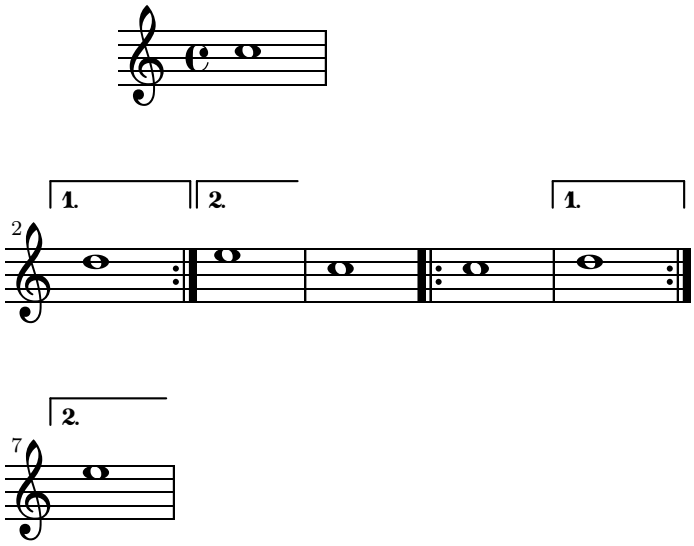
`repeat-fold.ly`

Folded repeat may not make sense without alternatives, and there should not be more alternatives than repeats.



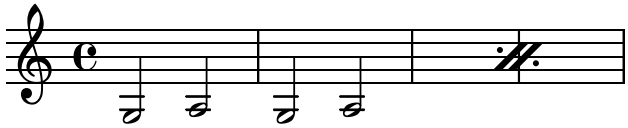
`repeat-line-break.ly`

Across linebreaks, the left edge of a first and second alternative bracket should be equal.



`repeat-percent-skipbars.ly`

Percent repeats are not skipped, even when `skipBars` is set.

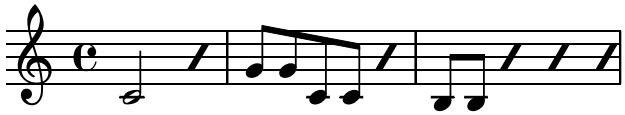


`repeat-percent.ly`

Measure repeats may be nested with beat repeats.

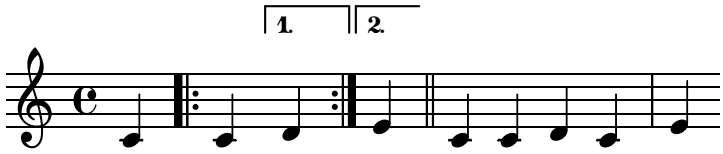
`repeat-slash.ly`

Within a bar, beat repeats denote that a music snippet should be played again.



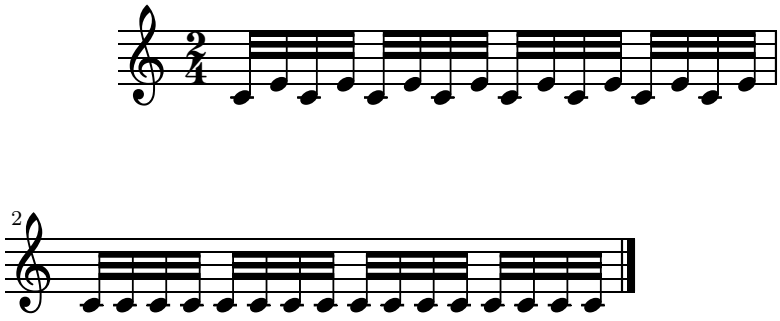
`repeat-unfold-all.ly`

Volta repeats may be unfolded through the Scheme function `unfold-repeats`.



`repeat-unfold-tremolo.ly`

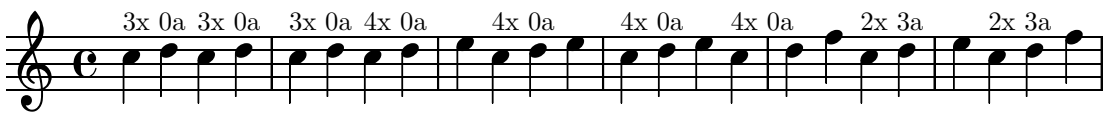
Unfolding tremolo repeats. Both fragments fill one 2/4 measure with 32nd notes exactly.



`repeat-unfold.ly`

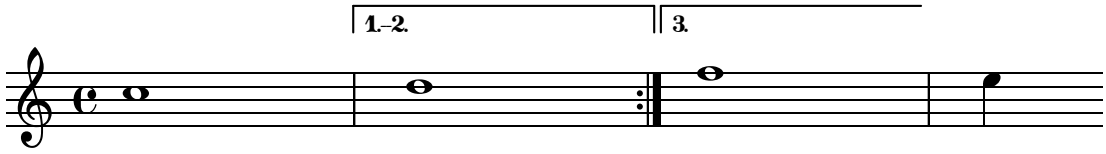
LilyPond has three modes for repeats: folded, unfolded and semi-unfolded. Unfolded repeats are fully written out. Semi unfolded repeats have the body written and all alternatives sequentially. Folded repeats have the body written and all alternatives simultaneo.ly. If the number of alternatives is larger than the repeat count, the excess alternatives are ignored. If the number of alternatives is smaller, the first alternative is multiplied to get to the number of repeats.

Unfolded behavior:



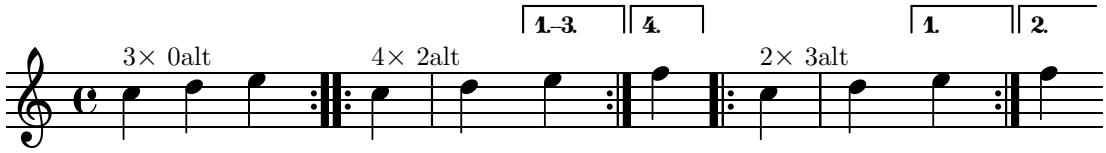
`repeat-volta-skip-alternatives.ly`

When too few alternatives are present, the first alternative is repeated, by printing a range for the 1st repeat.



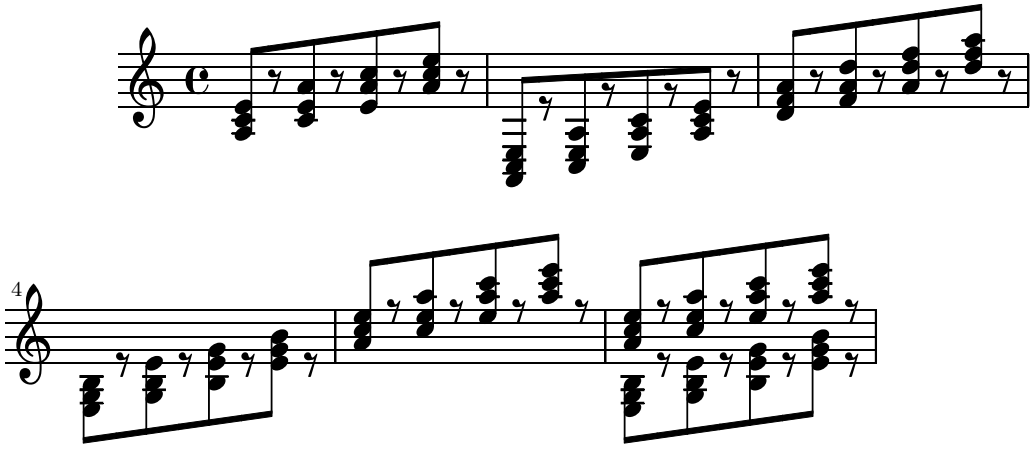
`repeat-volta.ly`

Volta (Semi folded) behavior. Voltas can start on non-barline moments. If they don't barlines should still be shown.



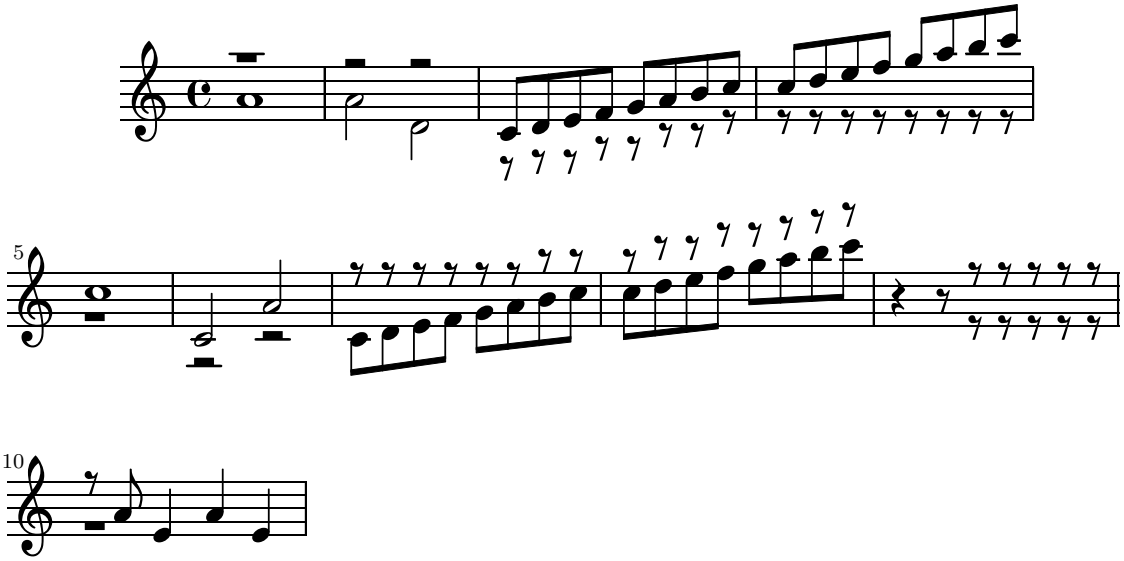
`rest-collision-beam.ly`

Rests under beams are only moved if necessary.



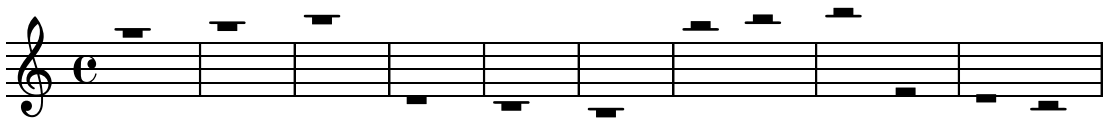
`rest-collision.ly`

Rests should not collide with beams, stems and noteheads. Rests may be under beams. Rests should be move by integral number of spaces inside the staff, and by half spaces outside. Notice that the half and whole rests just outside the staff get ledger lines in different cases.



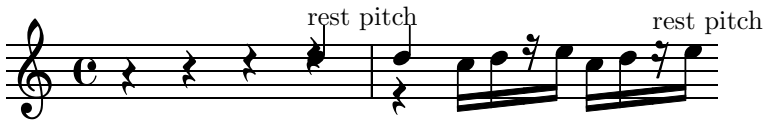
`rest-ledger.ly`

Whole and half rests moving outside the staff should get ledger lines.



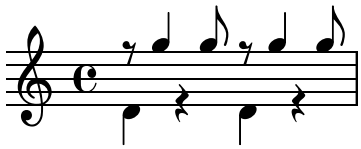
`rest-pitch.ly`

Rests can have pitches–these will be affected by transposition and relativization. If a rest has a pitch, rest/rest and beam/rest collision resolving will leave it alone.



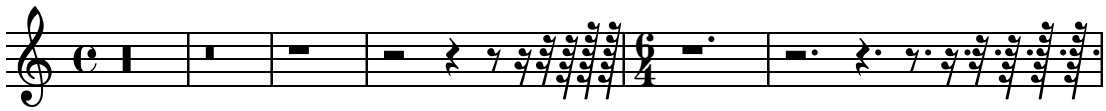
`rest-polyphonic.ly`

In polyphonic situations, rests are moved down even if there is no opposite note or rest. The amount is two `staff-space`s.



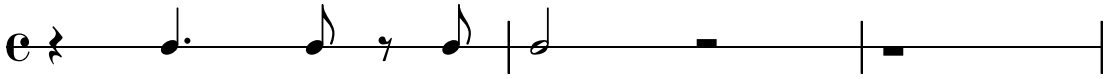
`rest.ly`

There is a big variety of rests. Note that the dot of 8th, 16th and 32nd rests rest should be next to the top of the rest. All rests except the whole rest are centered on the middle staff line.



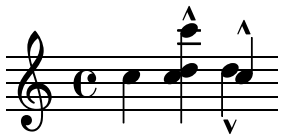
`rhythmic-staff.ly`

In rhythmic staves stems should go up, and bar lines have the size for a 5 line staff. The whole note hangs from the rhythmic staff.



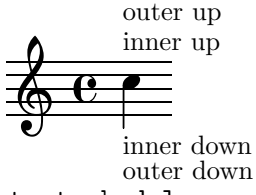
`script-collision.ly`

Scripts are put on the utmost head, so they are positioned correctly when there are collisions.



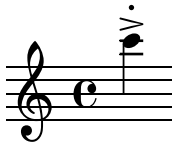
`script-stack-order.ly`

Scripts can be stacked. The order is determined by a priority field, but when objects have the same priority, the input order determines the order. Objects specified first are closest to the note.



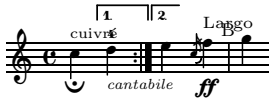
`script-stacked.ly`

Scripts may be stacked.



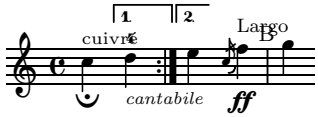
`size11.ly`

Different text styles are used for various purposes.



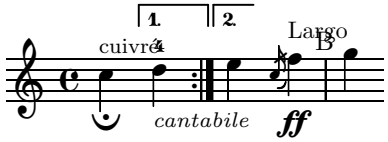
`size13.ly`

Different text styles are used for various purposes.



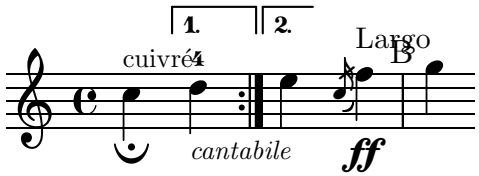
`size16.ly`

Different text styles are used for various purposes.



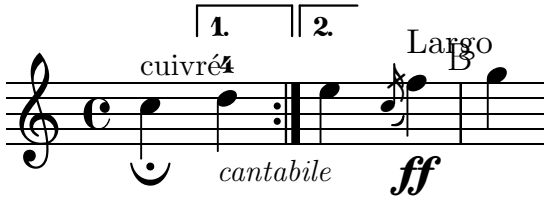
`size20.ly`

Different text styles are used for various purposes.



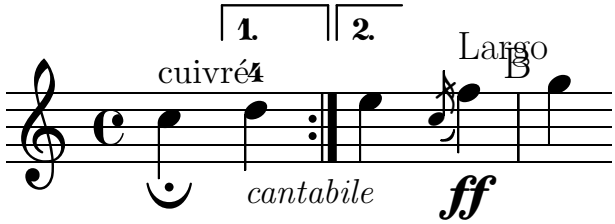
`size23.ly`

Different text styles are used for various purposes.



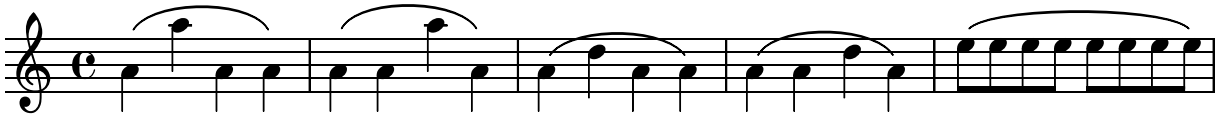
`size26.ly`

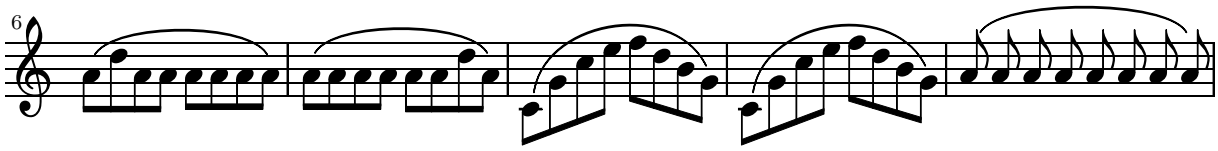
Different text styles are used for various purposes.



`slur-area.ly`

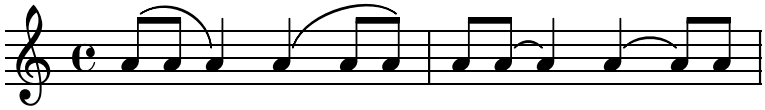
The area underneath an (up) slur is minimised to improve the shape.





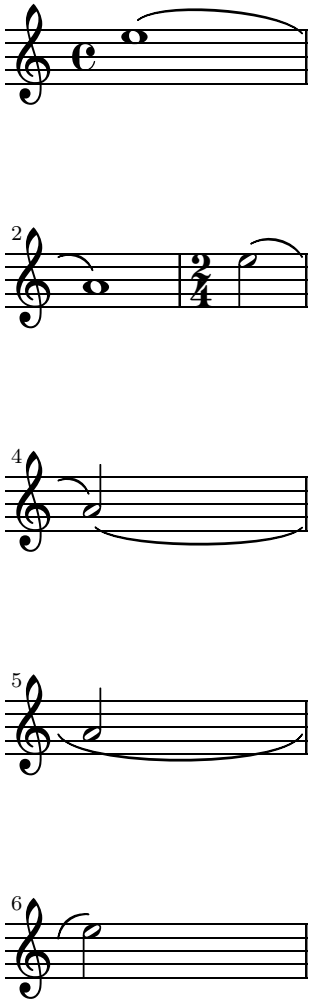
`slur-attachment.ly`

Slurs should be attached to note heads, except when they would collide with beams.



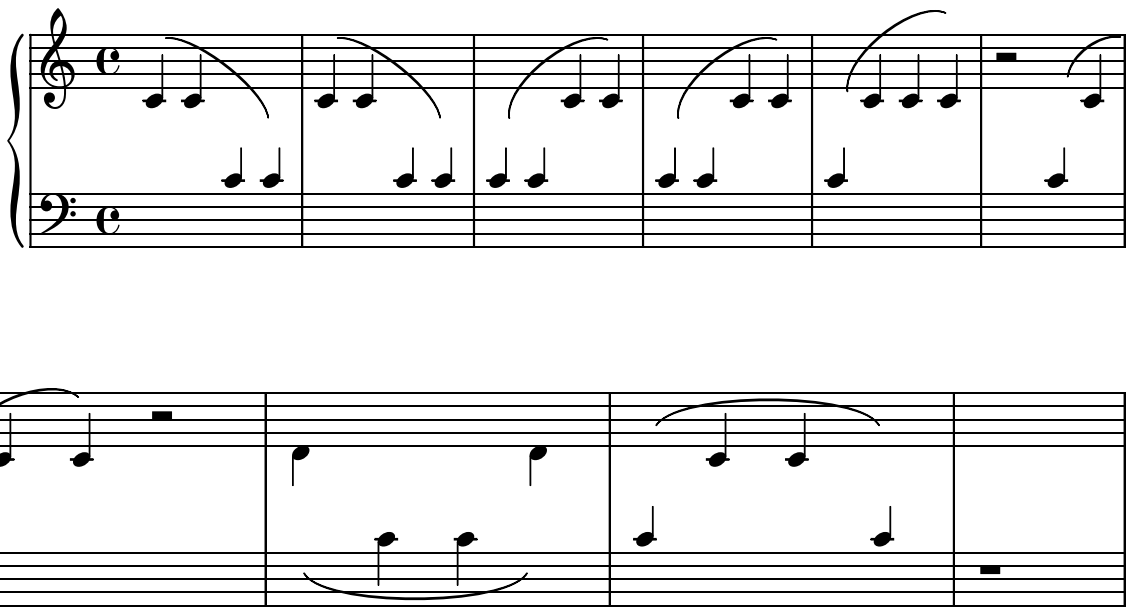
`slur-broken-trend.ly`

Across line breaks, slurs behave nicely. On the left, they extend to just after the preferatory matter, and on the right to the end of the staff. A slur should follow the same vertical direction it would have in unbroken state.



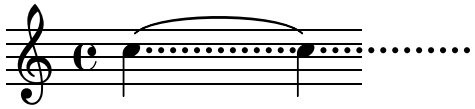
`slur-cross-staff.ly`

Slurs behave decently when broken across a linebreak.



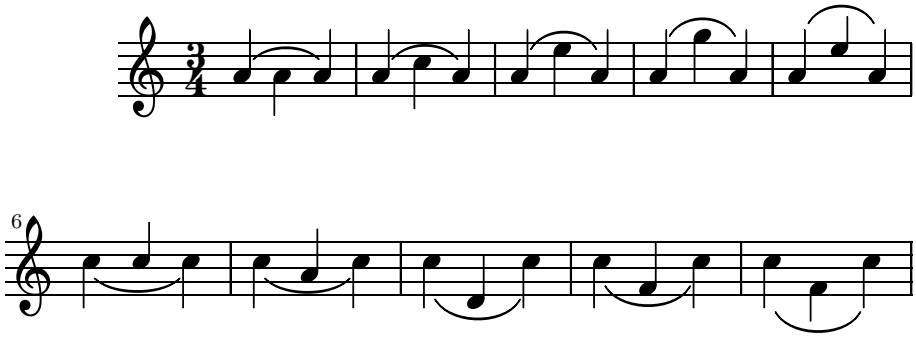
`slur-dots.ly`

Slurs should not get confused by augmentation dots. With a lot of dots, the problems becomes more visible.



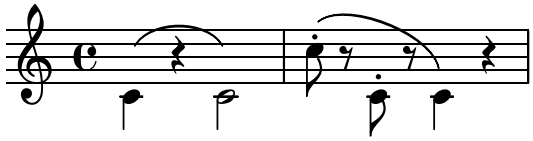
`slur-nice.ly`

Slurs should look nice and symmetric. The curvature may increase only to avoid noteheads, and as little as possible. Slurs never run through noteheads or stems.



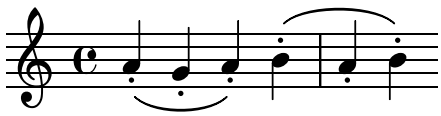
`slur-rest.ly`

Slurs may be placed over rest. The slur will avoid colliding with the rest.



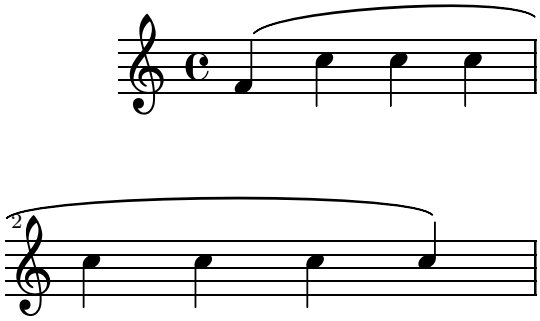
`slur-staccato.ly`

An extra offset may be added between a slur and staccato(s).



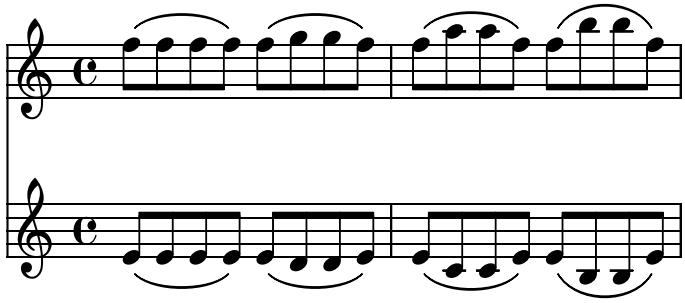
`slur-stem-broken.ly`

Trend of broken slur with user-overridden stem attachment should also follow the same vertical direction it would have had in unbroken state.



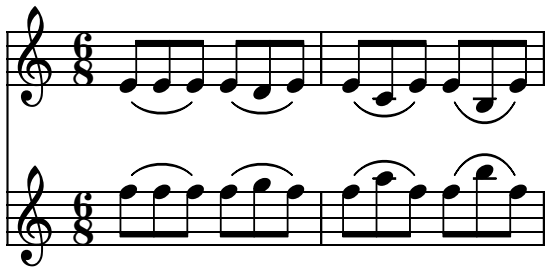
`slur-symmetry-1.ly`

Symmetric figures should lead to symmetric slurs.



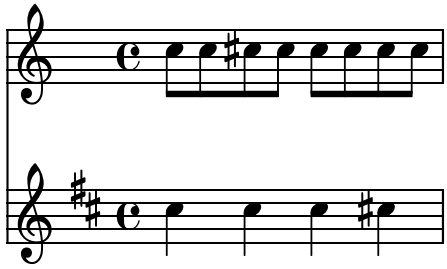
`slur-symmetry.ly`

Symmetric figures should lead to symmetric slurs.



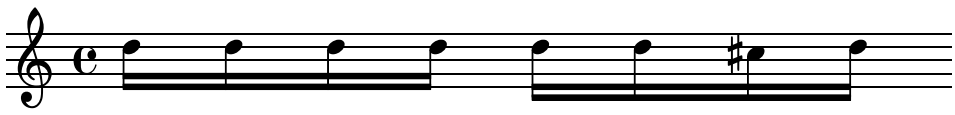
`spacing-accidental-staffs.ly`

Accidentals in different staves do not affect the spacing of the quarter notes here.



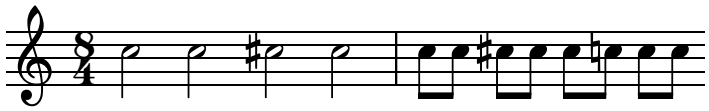
`spacing-accidental-stretch.ly`

Accidentals do not influence the amount of stretchable space. The accidental does add a little non-stretchable space.



`spacing-accidental.ly`

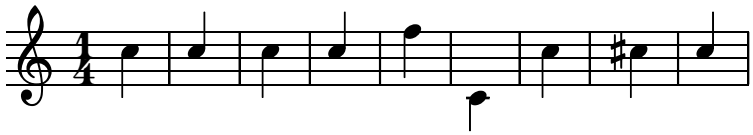
Accidentals sticking out to the left of a note will take a little more space, but only if the spacing is tight.



`spacing-bar-stem.ly`

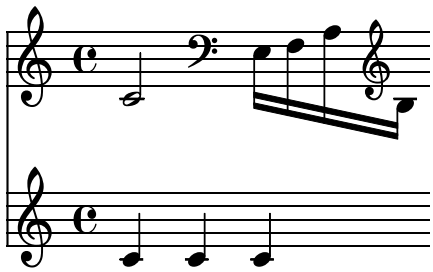
Downstem notes following a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.

Accidentals after the barline get some space as well.



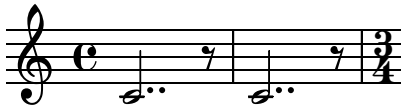
`spacing-clef-first-note.ly`

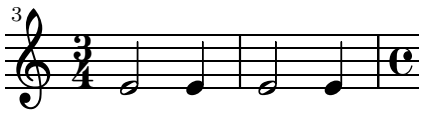
Clef changes at the start of a line get much more space than clef changes halfway the line.



`spacing-end-of-line.ly`

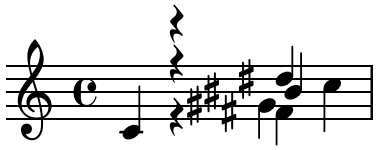
Broken engraving of a bar at the end of a line does not upset the space following rests and notes.





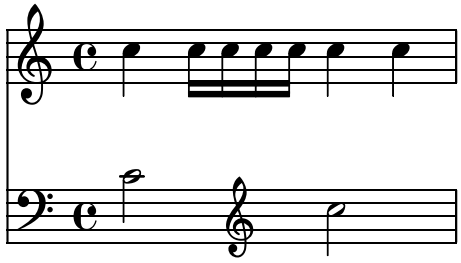
`spacing-ended-voice.ly`

A voicelet (a very short voice to get polyphonic chords correct) should not confuse the spacing engine.



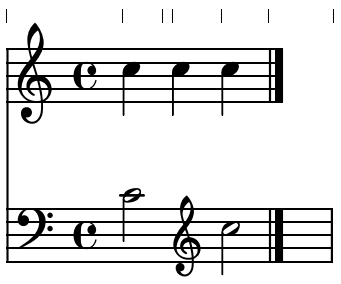
`spacing-folded-clef.ly`

A clef can be folded below notes in a different staff, if this does not disrupt the flow of the notes.



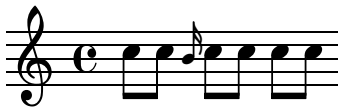
`spacing-folded-clef2.ly`

A clef can be folded below notes in a different staff, if there is space enough. With `Paper_column` stencil callbacks we can show where columns are in the score.



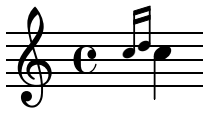
`spacing-grace-duration.ly`

Spacing uses the duration of the notes, but disregards grace notes for this. In this example, the 8ths around the grace are spaced exactly as the other 8th notes.



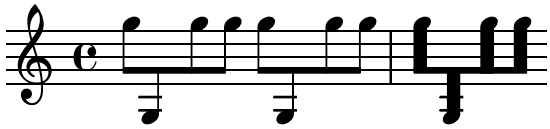
`spacing-grace.ly`

Grace note spacing. Should it be tuned?



`spacing-knee.ly`

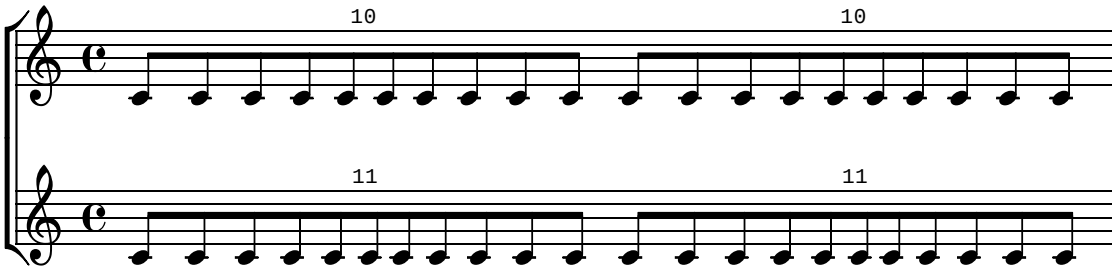
For knees, the spacing correction is such that the stems are put at regular distances. This effect takes into account the width of the note heads and the thickness of the stem.



`spacing-multi-tuplet.ly`

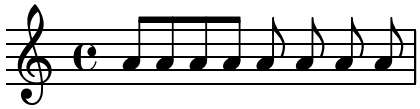
Concurrent tuplets should be equidistant on all staffs.

Note that this only spaces correctly (exactly) when `raggedright` is enabled. For a non-`raggedright` case, it still shows a bug: uneven spacing.



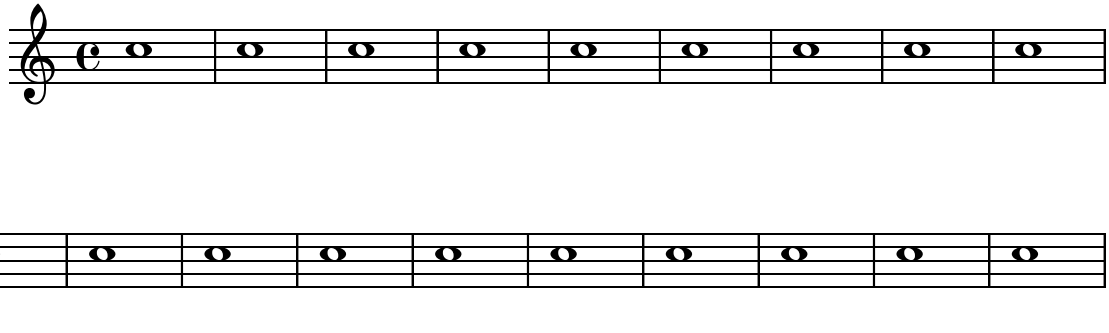
`spacing-note-flags.ly`

The flags of 8th notes take some space, but not too much: the space following a flag is less than the space following a beamed 8th head.



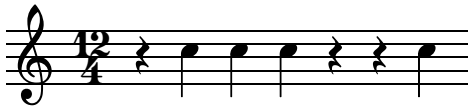
`spacing-ragged-last.ly`

If `raggedlast` is set, the systems are broken similar to paragraph formatting in text: the last line is justified.



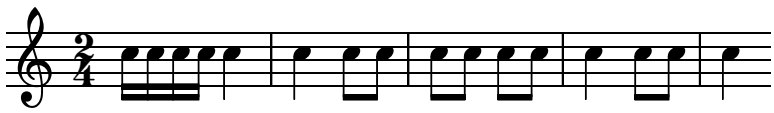
`spacing-rest.ly`

Rests get a little less space, since they are narrower. However, the quarter rest in feta font is relatively wide, causing this effect to be very small.



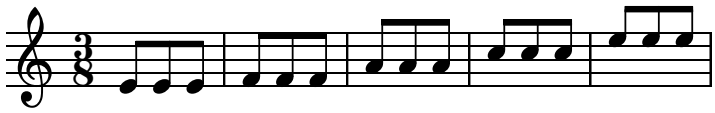
`spacing-short-notes.ly`

Notes that are shorter than the common shortest note get a space (i.e. without the space needed for the note) proportional to their duration. So, the 16th notes get 1/2 of the space of an eigth note. The total distance for a 16th (which includes note head) is 3/4 of the eighth note.



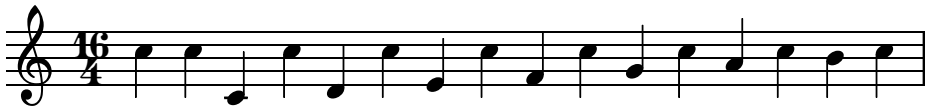
`spacing-stem-bar.ly`

Upstem notes before a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.



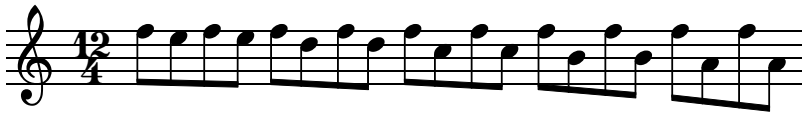
`spacing-stem-direction.ly`

There are optical corrections to the spacing of stems. The overlap between two adjacent stems of different direction is used as a measure for how much to correct.



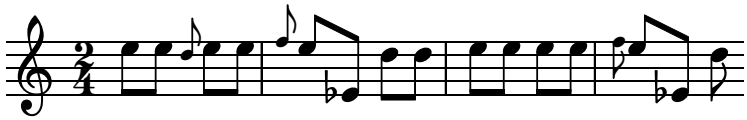
`spacing-stem-same-direction.ly`

For juxtaposed chords with the same direction, a slight optical correction is used. It is constant, and works only if two chords have no common head-positions range.



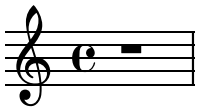
`spacing-to-grace.ly`

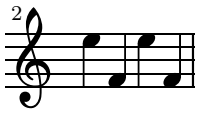
Space from a normal note (or barline) to a grace note is smaller than to a normal note.



`spacing-very-tight.ly`

When tightly spaced, the spaces between elements (hinterfleisch?) may approach zero. In that case, stems may touch the bar lines and opposite stems may touch eachother. In these situations, a mininum of about a note-width/interline space is needed, so that all vertical lines are approximately equally spaced in tightly spaced music.

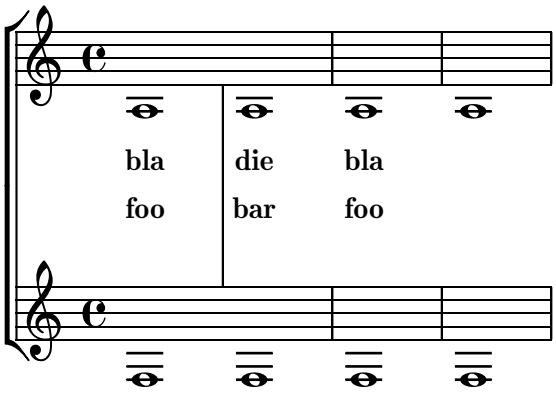




`span-bar.ly`

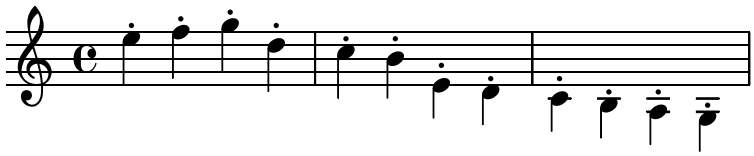
Span bars are drawn only between staff bar lines. By setting bar lines to transparent, they are shown only between systems.

Setting `SpanBar` transparent does the removes the barlines between systems.



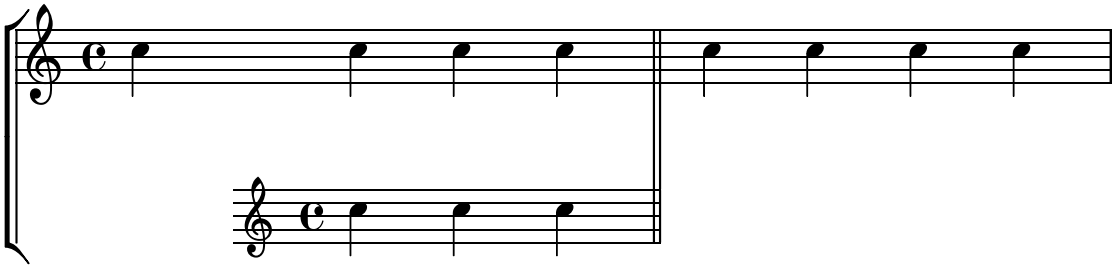
`staccato-pos.ly`

The staccato dot (and all scripts with follow-into-staff set) must not be on staff lines. The staccato dot is close to the notehead. If the head is in a space, then the dot is in the space next to it.



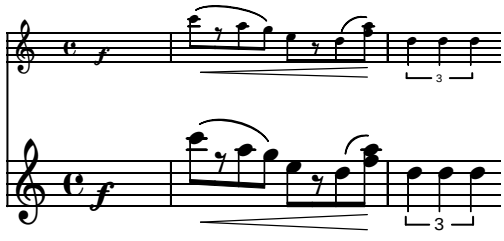
`staff-halfway.ly`

Staves starting and ending halfway include clefs and bar lines.



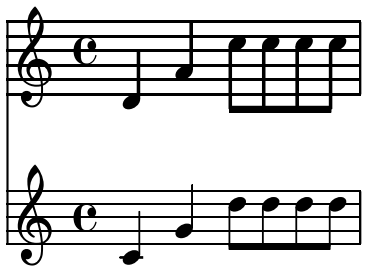
`staff-mixed-size.ly`

Staves may be present in several sizes within a score. This is achieved with an internal scaling factor. If the scaling factor is forgotten in some places, objects generally become too thick or too large on smaller staves.



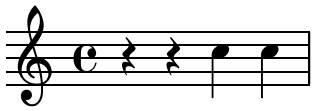
`staff-tweak.ly`

The staff is a grob (graphical object) which may be adjusted as well, for example, to have 6 thick lines and a slightly large `staff-space`. However, beams remain correctly quantized.



`stanza-number.ly`

Stanza numbers are put left of their lyric. Theyr are aligned in a column.



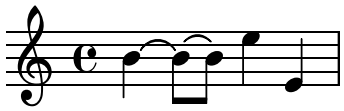
1. Foo

2. FFFooooo

`stem-direction.ly`

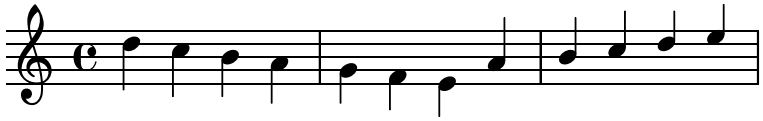
Beams, stems and noteheads often have communication troubles, since the two systems for y dimensions (1 unit = staffspace, 1 unit = 1 point) are mixed.

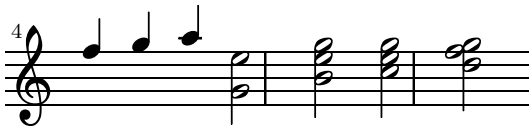
Stems, beams, ties and slurs should behave similarly, when placed on the middle staff line. Of course stem-direction is down for high notes, and up for low notes.



`stem-shorten.ly`

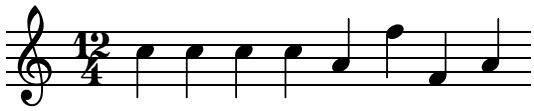
If note head is 'over' the center line, the stem is shortened. This happens with forced stem directions, and with some chord configurations.





`stem-spacing.ly`

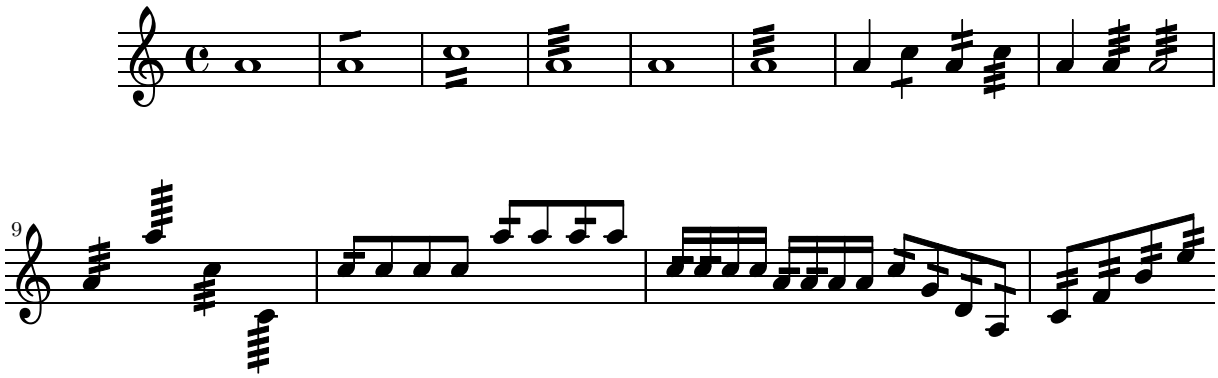
In a limited number of cases, the optical spacing effects are corrected. In this example, space for opposite pointed stems is adjusted.



`stem-tremolo.ly`

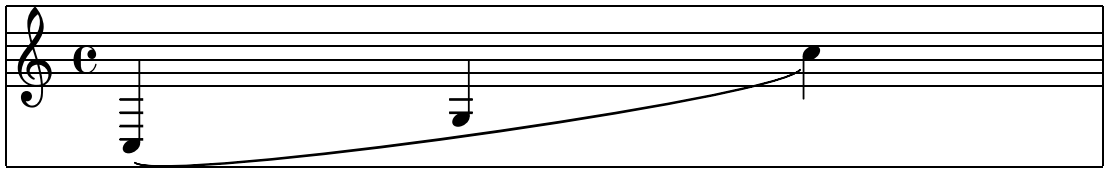
Stem tremolos or rolls are tremolo signs that look like beam segments crossing stems. If the stem is in a beam, the tremolo must be parallel to the beam. If the stem is invisible (e.g. on a whole note), the tremolo must be centered on the note.

:4 :8 :16 :32 x :



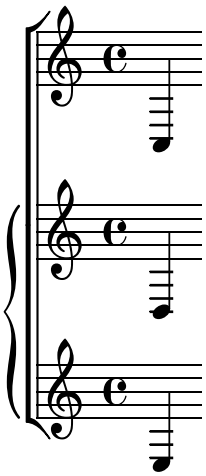
`system-extents.ly`

The size of every system is correctly determined; this includes postscript constructs such as slurs.



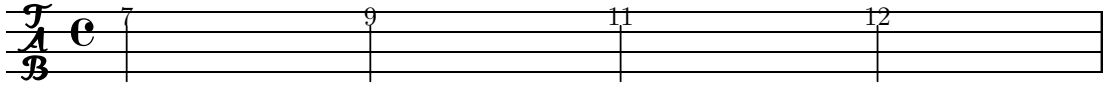
`system-start-bracket.ly`

The piano brace should be shifted horizontally if it is enclosed in a bracket.



`tablature-string-tunings.ly`

For other tunings, it is sufficient to set `stringTunings`. The number of staff lines is adjusted accordingly.

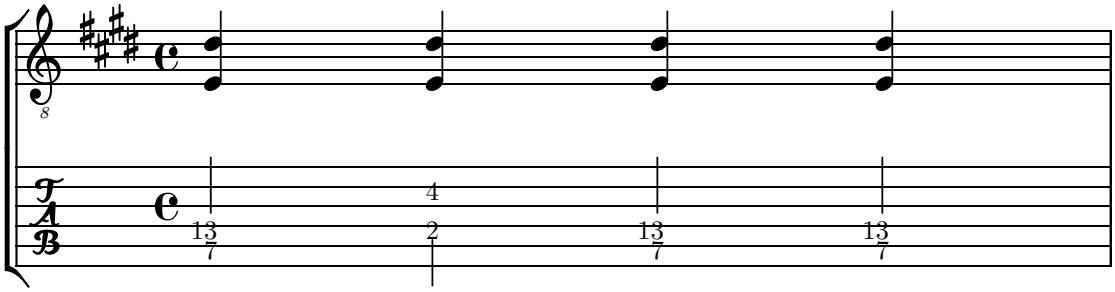


`tablature.ly`

A sample tablature, with both normal staff and tab.

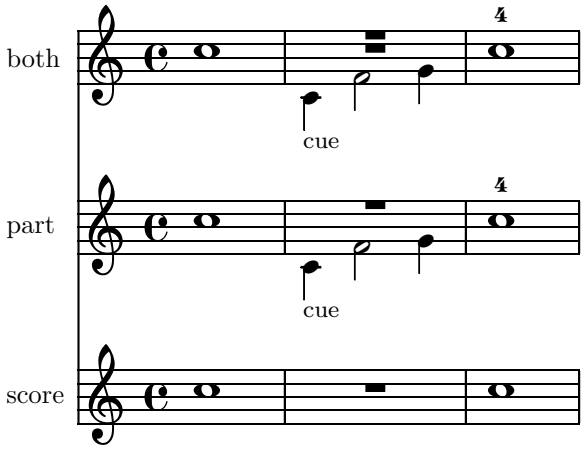
Tablature is done by overriding the note-head formatting function, and putting it on a 6-line staff. A special engraver takes care of going from string-number + pitch to number.

String numbers can be entered as note articulations (inside a chord) and chord articulations (outside a chord)



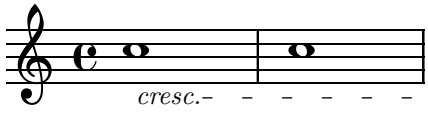
`tag-filter.ly`

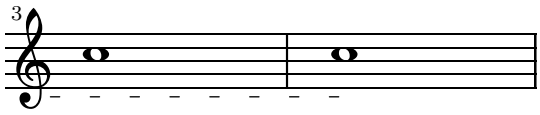
The `\tag` command marks music expressions with a name. These tagged expressions can be filtered out later. This mechanism can be used to make different versions of the same music. In this example, the top stave displays the music expression with all tags included. The bottom two staves are filtered: the part has cue notes and fingerings, but the score has not.



`text-spanner.ly`

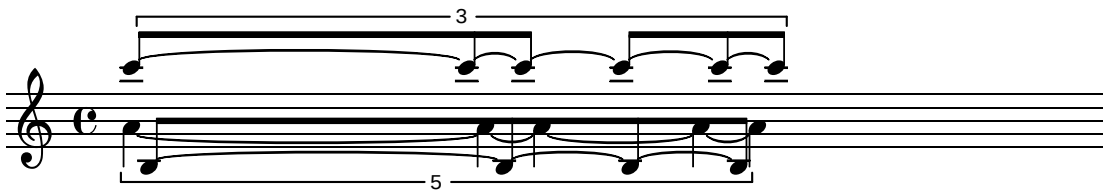
Text spanners should not repeat start text when broken.





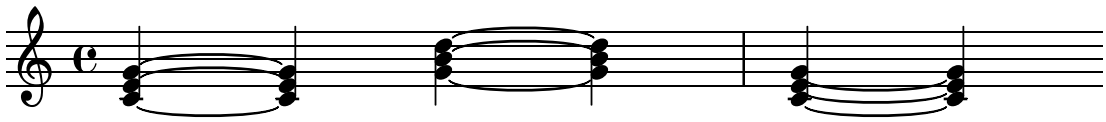
`tie-busy-grobs.ly`

Tie engraver uses `busyGrobs` to keep track of note heads. By throwing many mixed tuplets on the queue, one may have collisions between ties and beams.



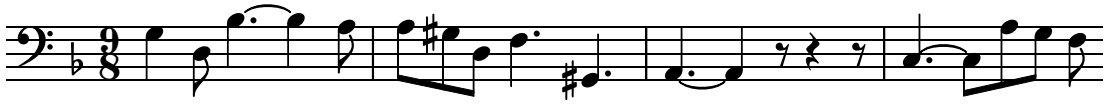
`tie-chord.ly`

When tying chords, the outer slurs point outwards, the inner slurs point away from the center of the staff. The behavior can be overridden by setting explicitly the `direction` of a `TieColumn`.



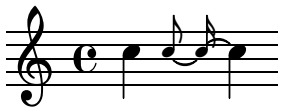
`tie-dots.ly`

Ties should not collide with dots.



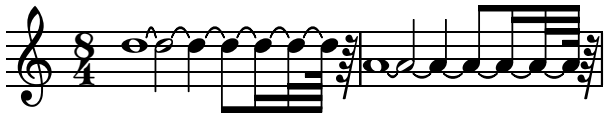
`tie-grace.ly`

Tieing a grace to the to a following grace or main note works.



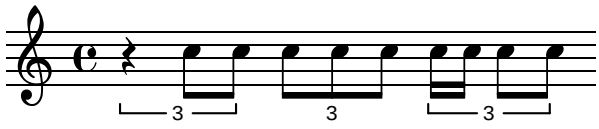
`tie.ly`

Ties are strictly horizontal. They are placed in between note heads. The horizontal middle should not overlap with a staffline.



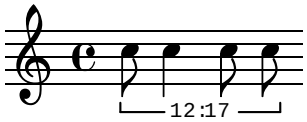
`tuplet-beam.ly`

In combination with a beam, the bracket of the tuplet bracket is removed. This only happens if there is one beam, as long as the bracket.



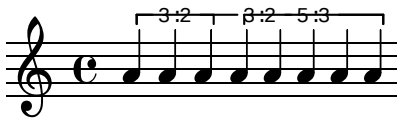
`tuplet-gap.ly`

The size of the tuplet bracket gap is adjusted to the width of the text.



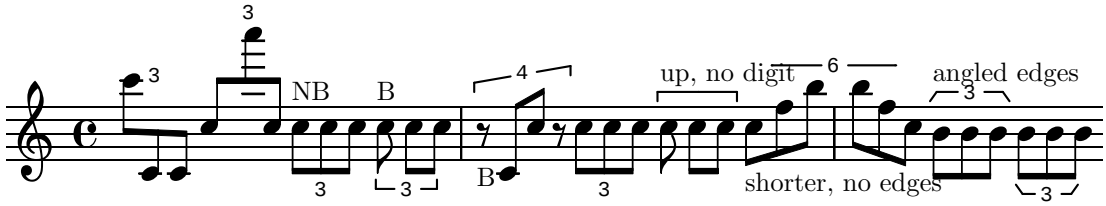
`tuplet-nest.ly`

By a manual hack for nested tuplets, an outer tuplet can be moved up.



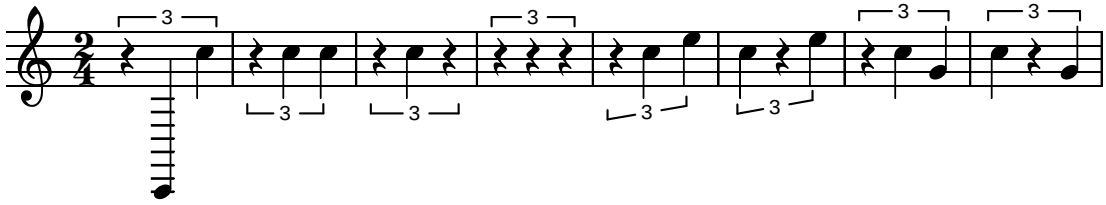
`tuplet-properties.ly`

Tuplet bracket formatting supports numerous options, for instance, bracketed (B) and non-bracketed (NB).



`tuplet-rest.ly`

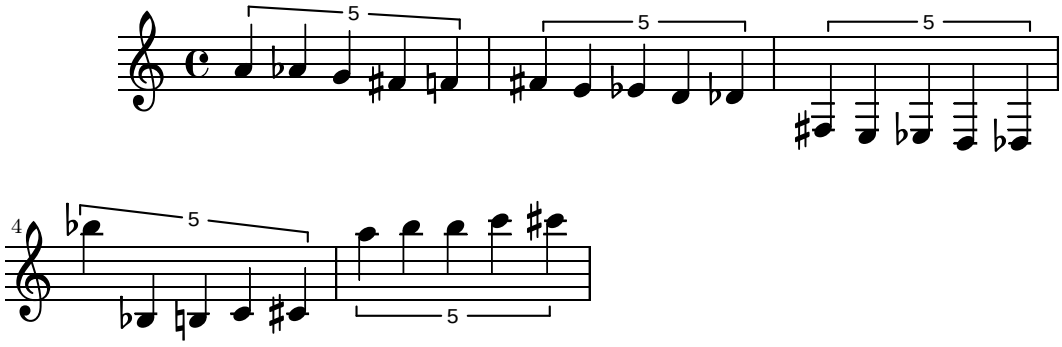
Tuplets may contain rests.



`tuplet-slope.ly`

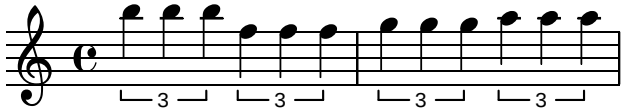
Tuplet brackets stay clear of the staff. The slope is determined by the graphical characteristic of the notes, but if the musical pattern does not follow graphical slope, then the bracket is horizontal

The bracket direction is determined by the dominating stem direction.



`tuplet-staffline-collision.ly`

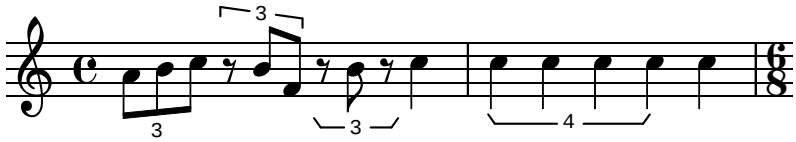
Horizontal tuplet brackets are shifted vertically to avoid staff line collisions.

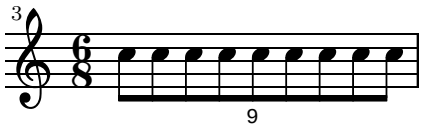


`tuplets.ly`

Tuplets are indicated by a bracket with a number. There should be no bracket if there is a beam exactly matching the length of the tuplet. The bracket does not interfere with the stafflines, and the number is centered in the gap in the bracket.

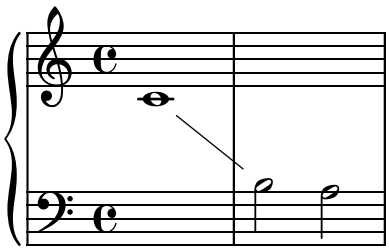
The bracket stops at the end of the stems, if the stems have the same direction as the bracket. The endings can be adjusted with `bracket-flare`.





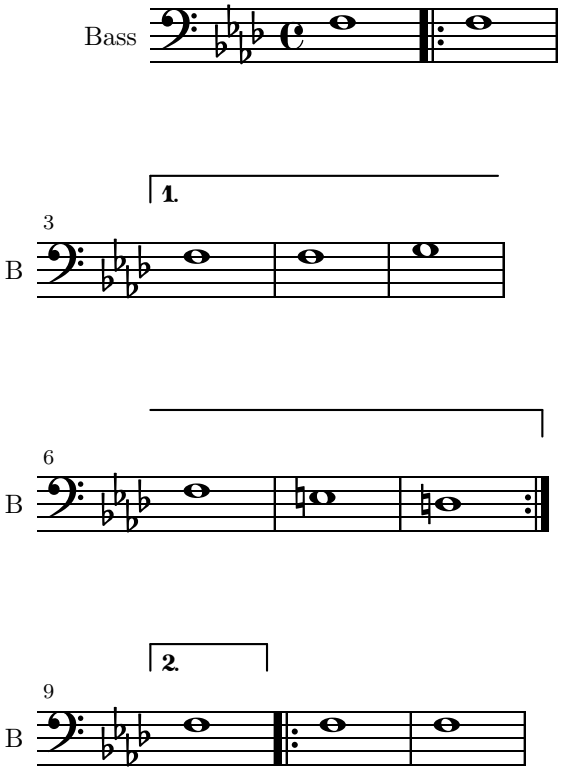
`voice-follower.ly`

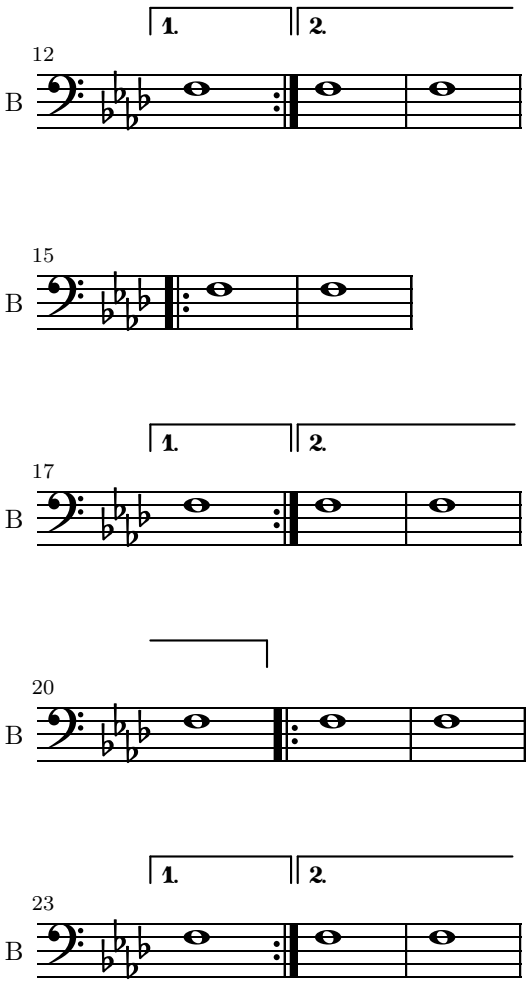
Whenever a voice switches to another staff a line connecting the notes can be printed automatically. This is enabled if the property `followVoice` is set to true.



`volta-broken-left-edge.ly`

Broken volta spanners behave correctly at their left edge in all cases.





`volta-multi-staff.ly`

By setting `voltaOnThisStaff`, repeats can be put on more staves in a score.