

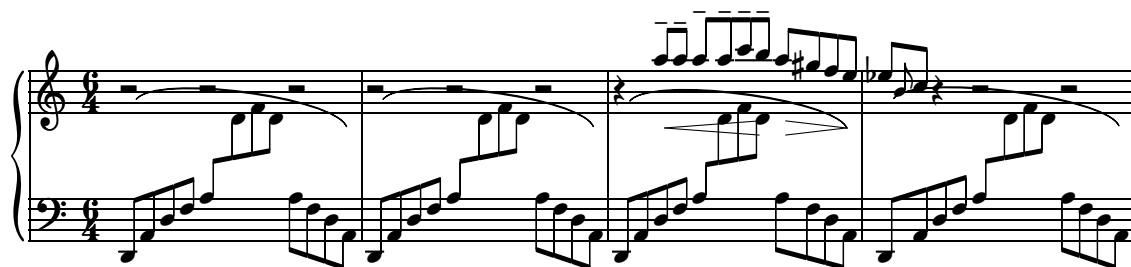
`‘+.ly’:`

0.1 Introduction

This document shows all kinds of tips and tricks, from simple to advanced. Here you may also find dirty tricks, or very the very latest features that have not been documented or fully implemented yet. This document is for LilyPond version 2.1.0.

`‘ac-extra-voice.ly’:`

When using automatic staff changes for the one voice, the other voice must be given a name explicitly.



`‘add-staccato.ly’:`

Using `make-music`, you can add various stuff to notes. Here is an example how to add staccato dots. Note: for this simple case one would not use `scm` constructs. See `separate-staccato.ly` first.



`‘add-text-script.ly’:`

You can add various stuff to notes using `make-music`. Here is an example of how to add an extra fingering.

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



`‘ambitus-mixed.ly’:`

Ambituses can be switched off or translated by using `applyoutput`.

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



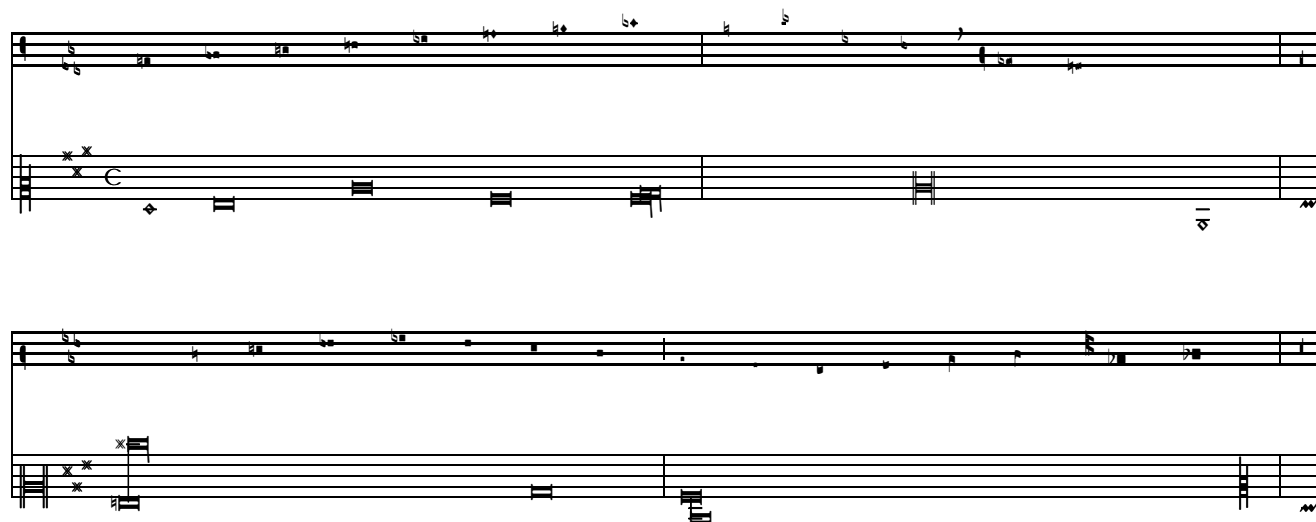
‘ancient-accidentals.ly’:

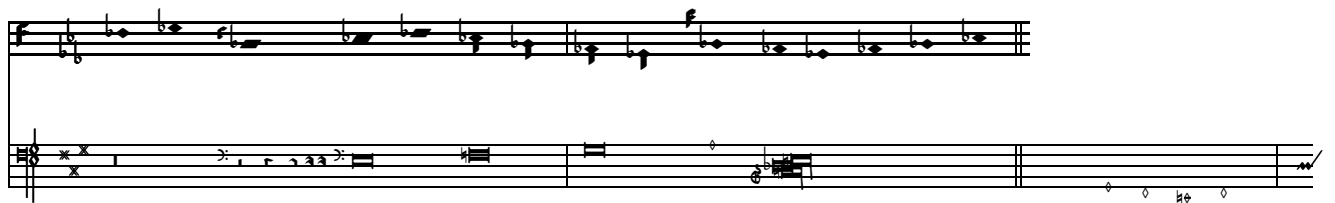
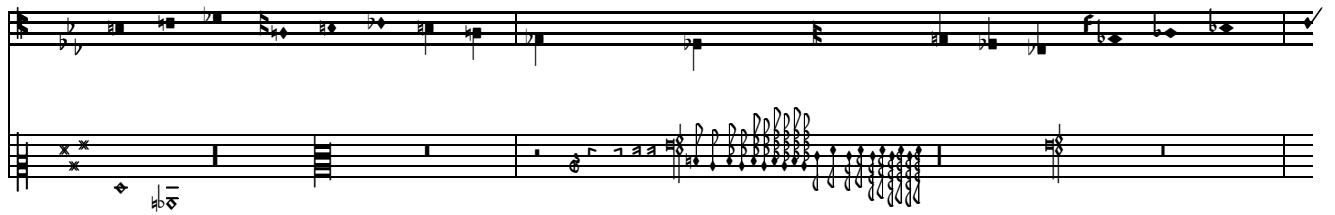
Accidentals are available in different ancient styles. This file lists all of them.



‘ancient-font.ly’:

Here is a display of many (all?) symbols that are included in LilyPond’s support of ancient notation.





‘ancient-time.ly’:

Should use old style.



‘bagpipe.ly’:

Here’s an example of bagpipe music.



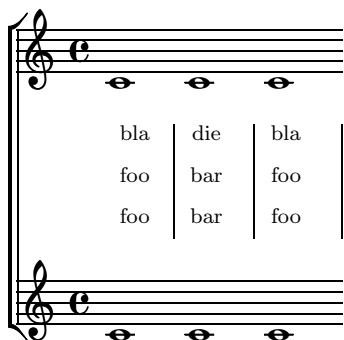
‘bar-always.ly’:

By setting `barAlways` and `defaultBarType`, you can automatically insert barlines everywhere.



`'bar-lines-lyric-only.ly':`

You can move around `Bar_engraver` and `Span_bar_engraver` if you want bar lines on lyrics.



`'bar-lines.ly':`

Different types of bar lines demonstrated.



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

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



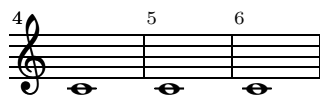
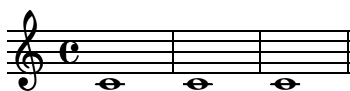
`'bar-number-regular-interval.ly':`

Bar numbers can also be printed at regular intervals.



`'bar-number-show-all.ly':`

Second line has bar numbers on start of every measure.



`'beam-auto-4-8.ly':`

You can override LilyPond's automatic beaming.



`'beam-auto-override.ly':`

This shows how auto-beaming settings can be overridden.

The auto-beamer will only engrave beams that end when:

- a rest is encountered
- another beam (entered manually) is encountered
- there's a 'gap' in the beam note's durations

The beam will be ended also when now `% beamAutoEnd = 0`.





`'beam-control.ly':`

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



`'beam-count.ly':`

You can alter the number of stems in a beam. Here we see two sets of four 32nds joined as if they were 8th notes.



`'beam-dir-functions.ly':`

There are several ways to calculate the direction of a beam:

- `majority` number count of up or down notes
- `mean` mean center distance of all notes
- `median` mean centre distance weighted per note

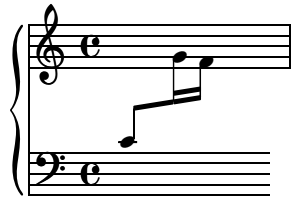
You can spot the differences of these settings from these simple examples:

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



`'beam-isknee.ly':`

LilyPond can beam across a Piano Staff.



`'beam-neutral-direction.ly':`

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



`'beam-rest.ly':`

Beams over rests.



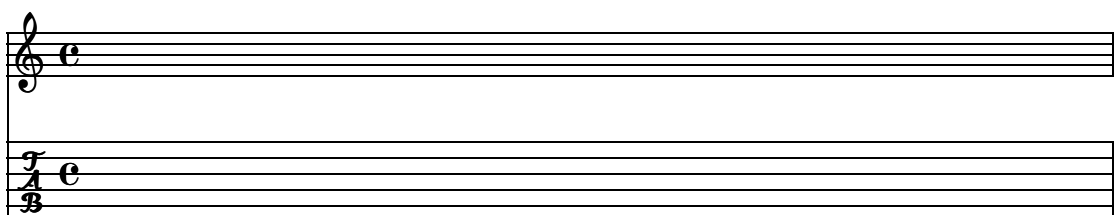
`'blank-notes.ly':`

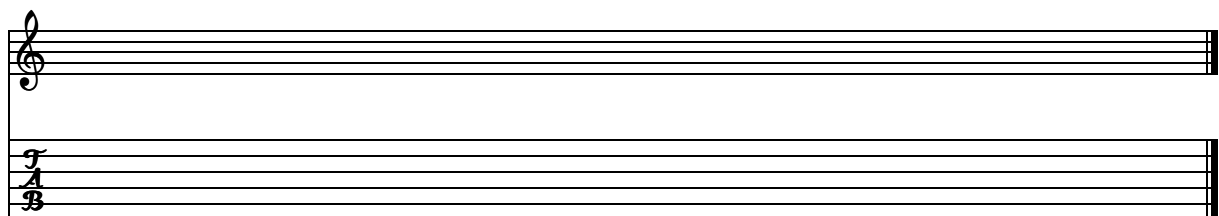
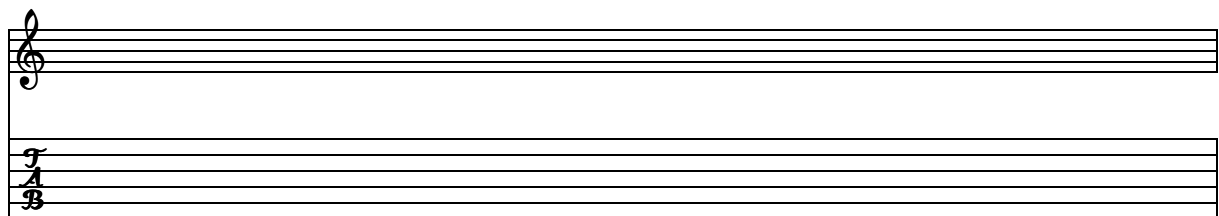
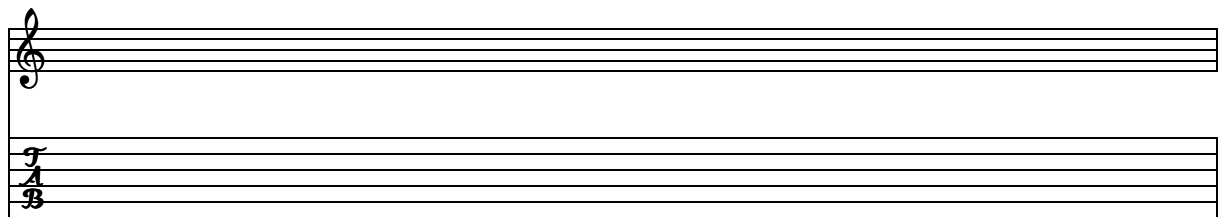
You can suppress printing of LilyPond output. This example shows you how to print invisible (or blank) notes. This can be very useful when you want to do wierd tricks with LilyPond (especially with slurs, since you can't attach a slur to a rest or spacer rest).



`'blank-paper-tab.ly':`

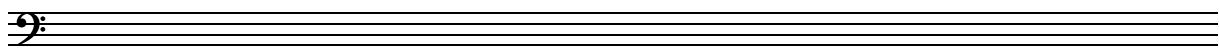
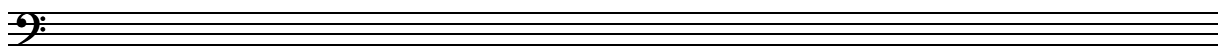
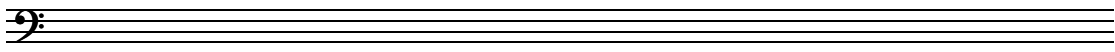
Blank music paper, another example: empty staves and a tablature staff.





`'blank-paper.ly':`

Blank music paper with clefs. Change the repeat count to get more staves.



`'boxed-molecule.ly':`

You can override the molecule callback to draw a box around arbitrary grobs.



`'caps.ly':`

You can set the font to use small caps.



what is THE MA-TRIX?

'cautionaries.ly':

LilyPond can display cautionary accidentals in different ways.



'chord-names-german.ly':

By setting `ChordNames.chordRootNamer`, the root of the chord may be named with a different function.

Setting `\germanChords` gives true german chord-names, `\semiGermanChords` gives semi-german chord-names - - with Bb and keeping the english names.

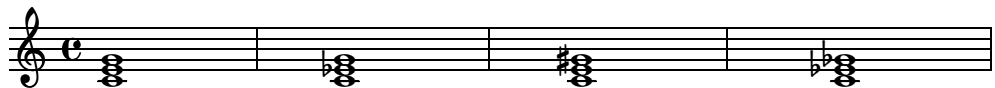
	C/C	C#/C#	B/B	B#/B#	Bb/Bb
german	C/c	C#/cis	H/h	H#/his	B ^d /b
semi-german	C/c	C#/cis	H/h	H#/his	B ^b /b



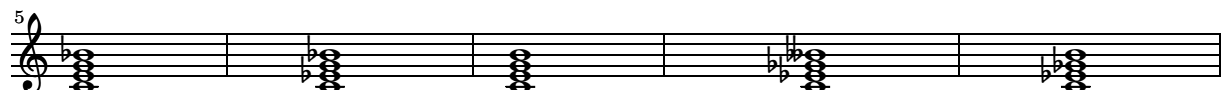
'chord-names-jazz.ly':

Chord names are generated from a list pitches. The functions constructing the names are customisable. This file shows Jazz chords, following Ignatzek (1995), page 17 and 18, Banter chords, and an alternative Jazz chord notation.

Ignatzek (default)	C	Cm	C+	C°
Alternative	C	C ^{b3}	C ^{#5}	C ^{b3 b5}



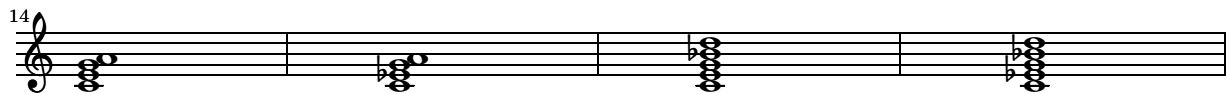
Def	C ⁷	Cm ⁷	C ^Δ	C ^{o7}	Cm ^{Δ/b5}
Alt	C ⁷	C ^{7 b3}	C ^{#7}	C ^{b3 b5 b7}	C ^{b3 b5 #7}



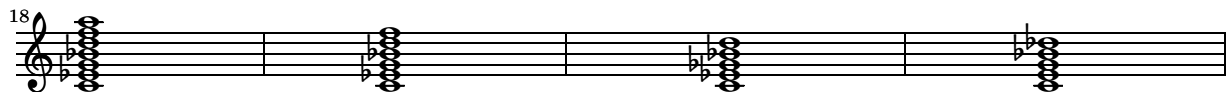
Def	C ^{7/#5}	Cm ^Δ	C ^{Δ/#5}	C ^φ
Alt	C ^{7 #5}	C ^{b3 #7}	C ^{#5 #7}	C ^{7 b3 b5}



Def	C ⁶	Cm ⁶	C ⁹	Cm ⁹
Alt	C ⁶	C ^{b3 6}	C ⁹	C ^{9 b3}



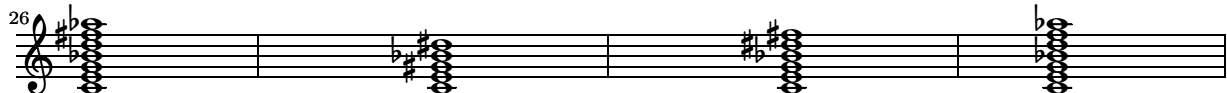
Def	Cm ¹³	Cm ¹¹	Cm ^{7/b5/9}	C ^{7/b9}
Alt	C ^{13 b3}	C ^{11 b3}	C ^{9 b3 b5}	C ^{7 b9}



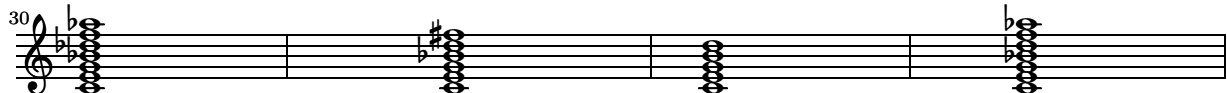
Def	C ^{7/#9}	C ¹¹	C ^{7/#11}	C ¹³
Alt	C ^{7 #9}	C ¹¹	C ^{9 #11}	C ¹³



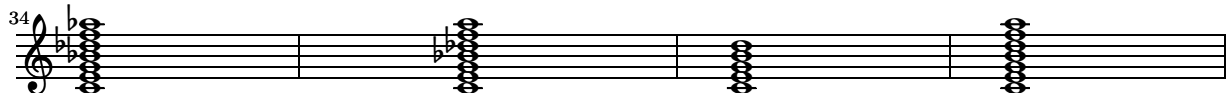
Def	C ^{7/#11/b13}	C ^{7/#5/#9}	C ^{7/#9/#11}	C ^{7/b13}
Alt	C ^{9 #11 b13}	C ^{7 #5 #9}	C ^{7 #9 #11}	C ^{11 b13}



Def	C ^{7/b9/b13}	C ^{7/#11}	C ^{Δ/9}	C ^{7/b13}
Alt	C ^{11 b9 b13}	C ^{9 #11}	C ^{9 #7}	C ^{11 b13}



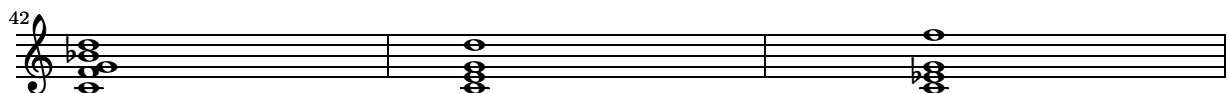
Def	C ^{7/b9/b13}	C ^{7/b9/13}	C ^{Δ/9}	C ^{Δ/13}
Alt	C ^{11 b9 b13}	C ^{13 b9}	C ^{9 #7}	C ^{13 #7}



Def	C ^{Δ/#11}	C ^{7/b9/13}	C ^{sus4}	C ^{7/sus4}
Alt	C ^{9 #7 #11}	C ^{13 b9}	C ^{add4 5}	C ^{add4 5 7}

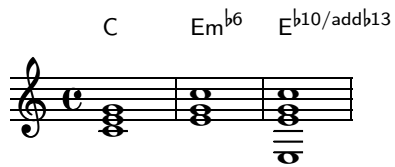


Def	C ^{9/sus4}	C ^{add9}	Cm ^{add11}
Alt	C ^{add4 5 7 9}	C ^{add9}	C ^{b3 add11}



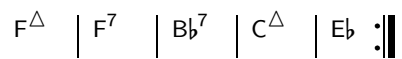
`'chord-names-no-inversions.ly':`

Chord names don't attempt to find inversions and bass notes.



`'chords-without-melody.ly':`

Jazz chords can also be used without notes.



`'clef-8-syntax.ly':`

Appending `_8` or `^8` to a clef name will add an octavation sign to the clef, although the clef name needs to be in quotes (such as `"treble^8"`).



`'clef-end-of-line.ly':`

Scales, but with clef and key signature at the end of the line.



`'clef-manual-control.ly':`

You can use the clef engraver by using `\property` directly. `\clef` is merely a front-end to this. All the notes in this example are central C.



`'coriolan-margin.ly':`

Demonstration of how to set up an orchestral score (Beethoven's Coriolan overture).

2 Flauti

2 Oboi

Clarineti
in B \flat

2 Fagotti

Corni
in E \flat

2 Trombe
(C)

Timpani
(C-G)

Violino I

Violino II

Viola

Violoncello
e
Contrabasso

`'count-systems.ly':`

Display the number of systems, or the system number of a Grob. This can be most useful to ascertain that a piece uses a specified number of lines.

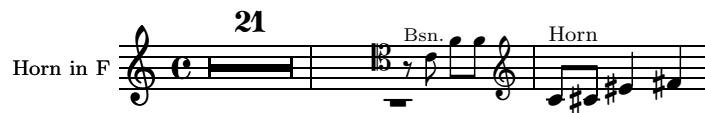
`'crescendi.ly':`

LilyPond can print crescendi in a number of different ways.



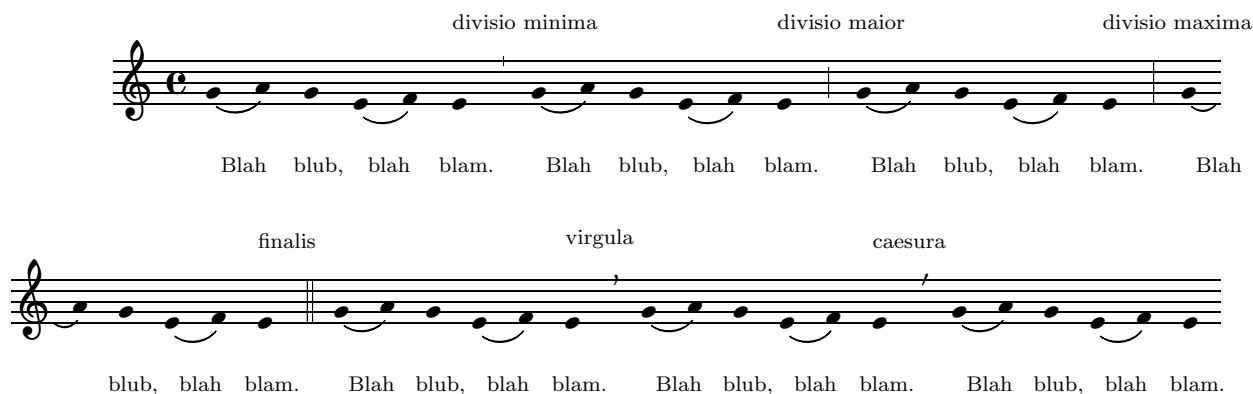
‘cue-notes.ly’:

Cue notes should be set in smaller type.



‘divisiones.ly’:

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



‘drarn-slurs.ly’:

Slurs can be forced to always attach to note heads.



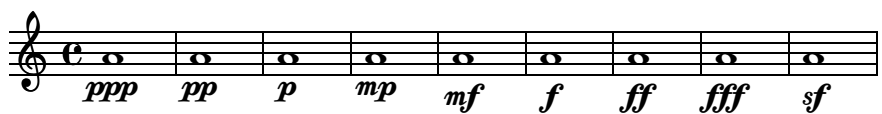
‘drarn.ly’:

You can attach slurs and ties to noteheads.



‘dynamic-absolute-volume.ly’:

Absolute dynamics have effect in MIDI files.



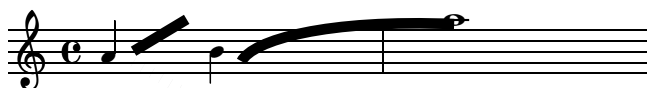
‘dynamic-extra.ly’:

Additional tricks for dynamics. Pi‘u forte dynamic script.



‘embedded-postscript.ly’:

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



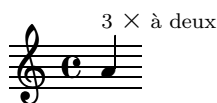
‘embedded-scm.ly’:

You can embed scm functions in your scores.



‘embedded-tex.ly’:

You can embed Tex commands in your score.



‘engraver-contexts.ly’:

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



‘engraver-one-by-one.ly’:

The notation problem, *what* symbols to create, is handled by plugins. Each plugin is called Engraver. In this example, we switch on engravers one by one, in the following order

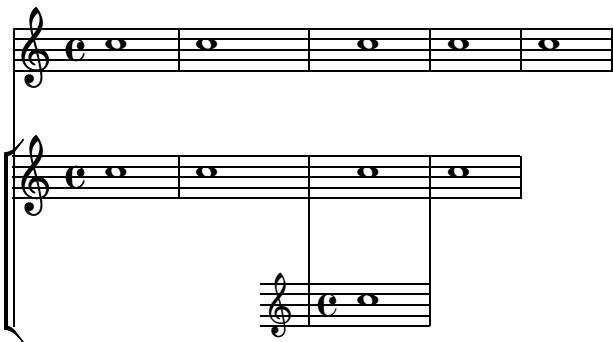
- Note heads
- Staff symbol
- Clef
- Stem
- Beams, slurs, accents
- Accidentals, bar lines, time signature, and key signature.

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

• • • • • • • • • •

`‘extra-staff.ly’:`

You can add an extra staff after the beginning of a piece.



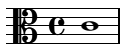
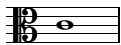
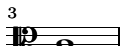




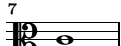
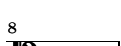
`‘figured-bass-alternate.ly’:`

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

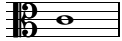


`‘fill-a4.ly’:`

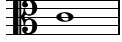
This should fill a4 paper.


2

3

4

5

6

7

8

9


10



11



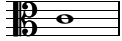
12



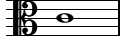
13



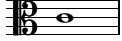
14



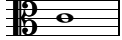
15



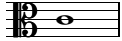
16



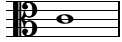
17



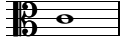
18



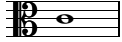
19



20



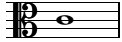
21



22



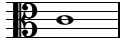
23



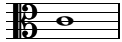
24



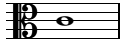
25



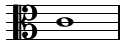
26

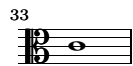
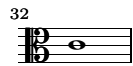
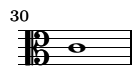


27



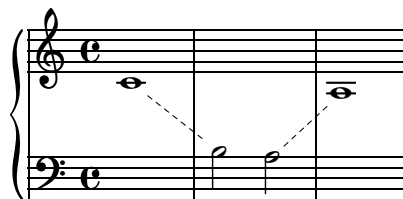
28





`'follow-thread.ly':`

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



`'force-hshift.ly':`

Force `hshift` to override collisions.



`'gourlay.ly':`

This is taken from Gourlay's paper on breaking lines.



`'gregorian-scripts.ly':`

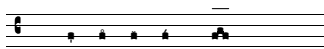
Gregorian Scripts:

ictus, circulus, semicirculus, accentus, episem.

TODO: augmentum. Syntax: either as bracket (`\augmentumInitium`, `\augmentumFinis`), or as head prefix with subsequently collecting all dots and putting them behind the ligature in a vertical row. Counterexample to the second approach: Graduale Triplex, tempus per annum, hebdomada septima, alleluia (page 280).

FIXME: horizontal spacing (ragged right mode).

FIXME: padding/minimum-distance is fragile.



`'harmonic.ly':`

For stringed instruments, artificial harmonics are notated with two different notehead styles on the same stem.



`'header-iffalse.ly':`

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

This example puts the current version in the tagline via Scheme. Since the tagline isn't used in creating the webpage, this example doesn't output anything unusual in the collated snippets.



`'hshift.ly':`

You can manually shift notes horizontally.



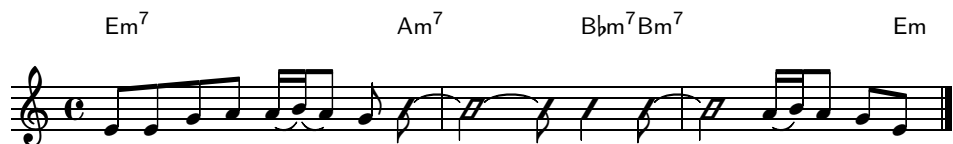
`'hymn.ly':`

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



`'improv.ly':`

Noteheads for improvisation have a different shape.



`'incipit.ly':`

This shows how to make an “incipit” to indicate scordatura tuning of a violin part, using the `clefStyle` property. The two first bars of Biber’s Rosary sonata III.



`'instrument-name-grandstaff.ly':`

You can name the whole grandstaff in addition to individual staves.



`'ly2dvi-testpage.ly':`

This file tests ly2dvi titling. It should be processed with ly2dvi.



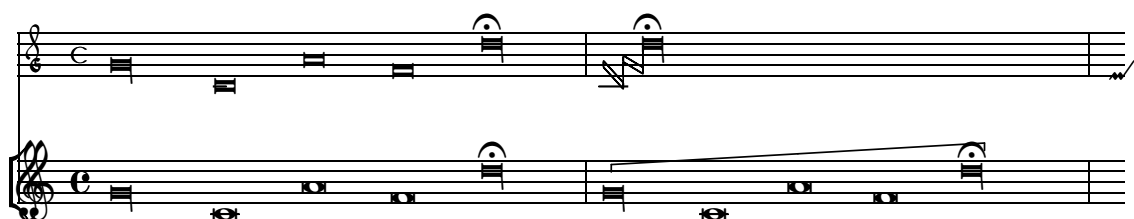
`'maximum-rest-count.ly':`

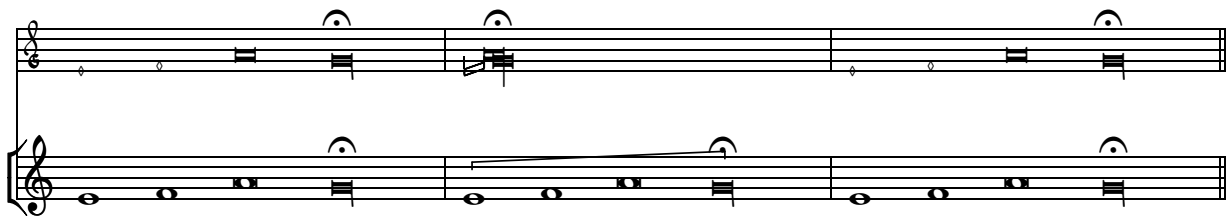
Control the number of rests in a collision with `maximum-rest-count`.



`'mensural-ligatures.ly':`

LilyPond can print mensural ligatures.





`'midi-scales.ly':`

MIDI and midi2ly test file. Diff between this and midi2ly.py should be minimal.





`'move-accidentals.ly':`

Positions of accidentals may be manually set. This involves some scheme code.



`'move-specific-text.ly':`

You can move objects around with scheme. This example shows how to move text around.



`'music-box.ly':`

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



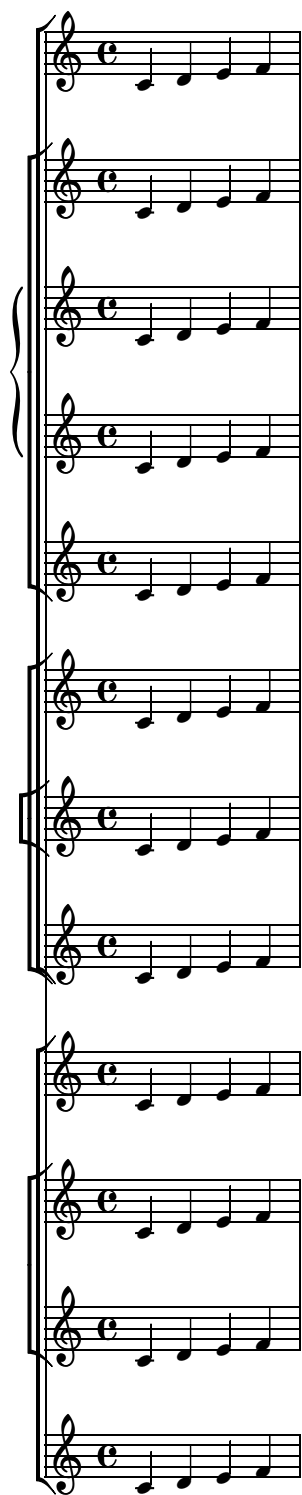
`'music-creation.ly':`

You can create music expressions from Scheme. The mechanism for this is rather clumsy to use, so avoid it if possible.



`'nested-groups.ly':`

LilyPond can print nested groups of staves.



`'no-bar-lines.ly':`

You can stop LilyPond from printing bar lines by removing the engraver.



`'no-staff.ly':`

You can stop LilyPond from printing the staff by removing the engraver.



`'octave-duplicate.ly':`

Octave doubling parts of music.



`'ossia.ly':`

Ossias present alternatives for a piece. They are not really supported, but can be somewhat faked in lily.



`'part-combine-moments.ly':`

When you combine two voices with the same notes, you should only have one stem.

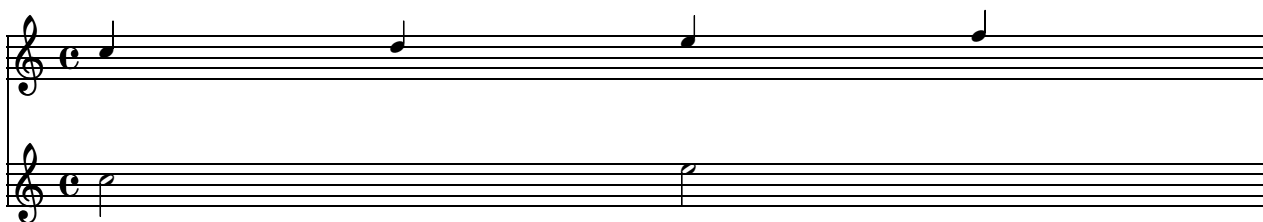


2

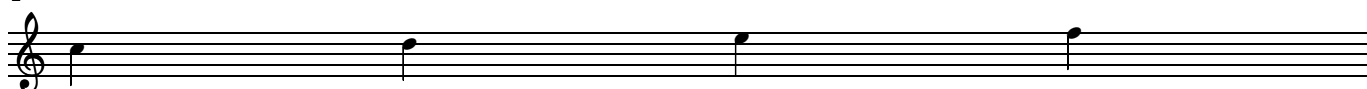


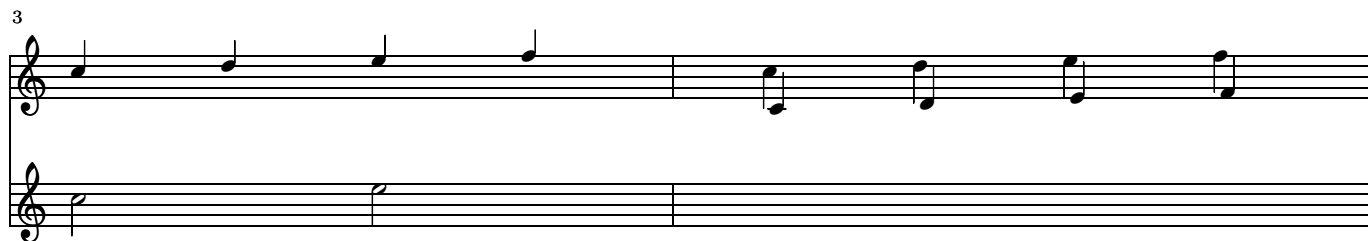
`'part-combine-staff.ly':`

You can combine parts on two staves, as well as two voices.



2





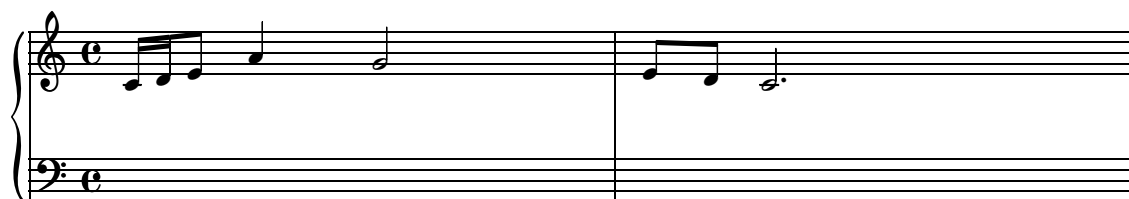
`'part-combine.ly':`

In orchestral scores and hymns, voices are traditionally combined onto one staff. LilyPond has a part combiner, that combines or separates two voices according to actual rhythm and pitch. User-defined texts such as “solo” and “a2” are typeset automatically, as appropriate.



`'partial-blank.ly':`

When entering half music (i.e. for students to complete by hand) you need the spacing to correspond to the timing – all measures same length, etc. This thing implements it by adding invisible staff with lots of fast notes.



`'pedal.ly':`

Piano pedal symbols merge stop and start. The strings are configurable. Text style, bracket style, and a mixture of both are supported.



`'phrasing-slur-height.ly':`

Make PhrasingSlur higher, to avoid collision from other slurs.



`'polymetric-differing-notes.ly':`

You can have multiple time signatures occurring at the same time, with different durations aligned. This is done by 1. compressing one of the lines, analogous to imes, but without the bracket, and 2. manually setting timeSignatureFraction to the desired fraction.

This example puts 3/4, 9/8 and 10/8 in parallel. The last staff shows what happens on the inside: a 3/4 time signature is combined with a 3/5 tuplet yielding the equivalent of a 10/8.



`'polymetric.ly':`

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

This is done by moving the timing engraver to staff context. Also, Staff should be given the alias `Timing` to make `ime` command work correctly. Barlines distort the regular spacing, though.



`'preset-extent.ly':`

Grob extents may be hard coded using grob properties. This requires `Grob::preset_extent ()` function.

The lyrics in this example have extent (-10,10) which is why they are spaced so widely.

foo- bar baz

`'repeat-manual.ly':`

You can manually control repeat signs and numbers to produce unusual output.



`'repeat-shorter-bracket.ly':`

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



intro	chorus	one	verse	five
	chorus	two	verse	
	chorus	three	verse	
	chorus	four	verse	

`'repeat.ly':`

You can use alternate lyrics as well as alternate notes for repeats.

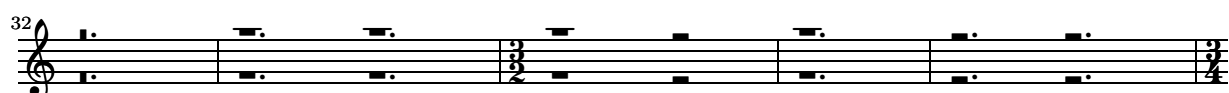
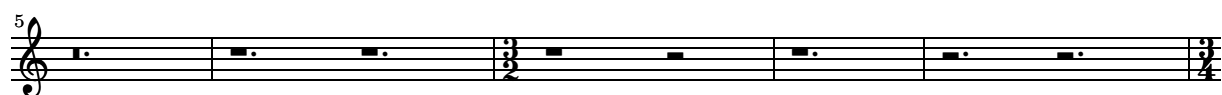


De eer- ste maat en dan twee keer en dan nog dit er ach- ter aan
een koe- plet

`'rest-dot-positions.ly':`

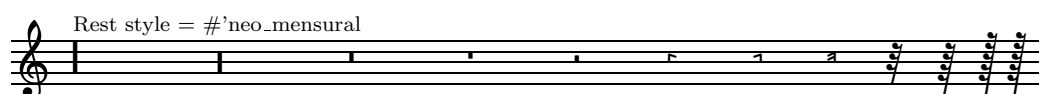
This file tests dotted rests.





'rests.ly':

Rests in various styles.





'reverse-music.ly':

Simple customised music apply.



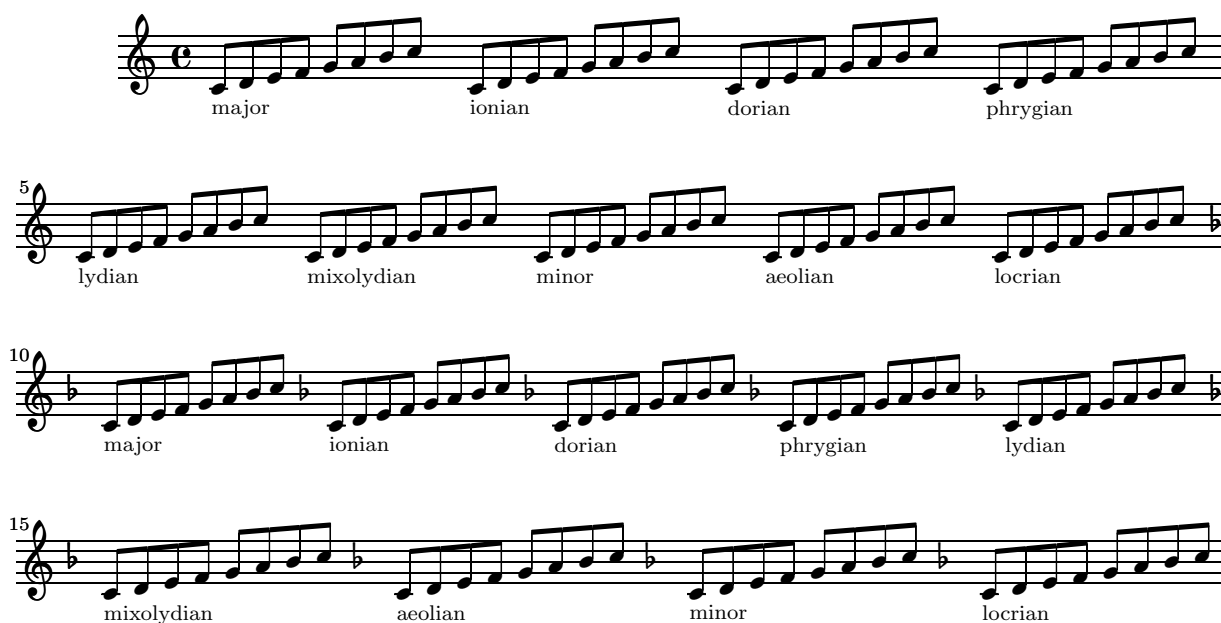
'rhythm-exercise.ly':

This example shows a way to generate rhythm exercises with LilyPond (e.g. no staff but retaining the barlines).



'scales-greek.ly':

Show different scales.



'scheme-interactions.ly':

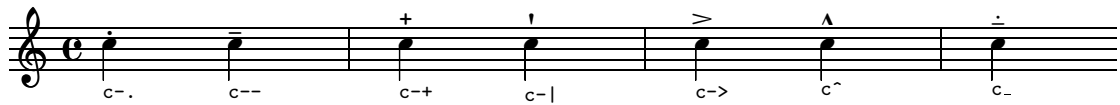
With `ly:export`, you can pass of the result of Scheme expressions as lilypond input. Within a Scheme expression, you can use, define or change lilypond variables.

In this example, the E-s and D-s are generated using scheme functions, and woven together with manually entered C-s.



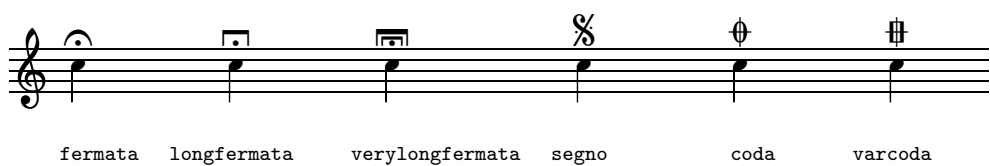
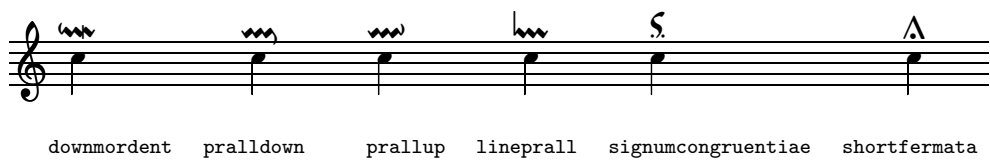
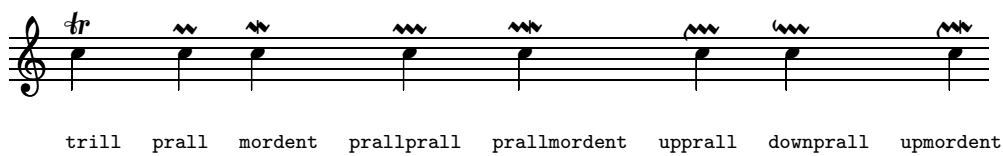
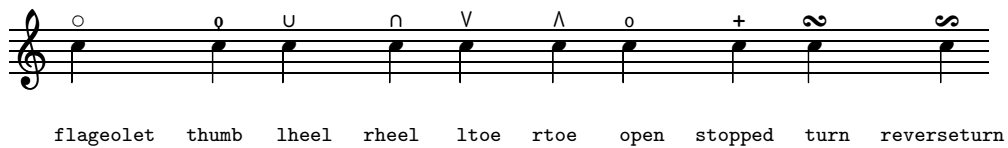
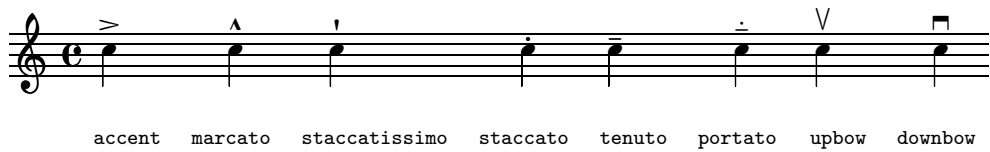
`'script-abbreviations.ly':`

Some scripts may be entered using an abbreviation.



‘script-chart.ly’:

A chart showing all feta scripts.



‘script-priority.ly’:

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



‘script-stack.ly’:

You can stack text and articulations.



'separate-staccato.ly':

You can enter notes and articulations separately, and merge them into one thread. Here is an example to add repeated staccato dots.



'slur-attachment-override.ly':

In some cases you may want to set slur attachments by hand.



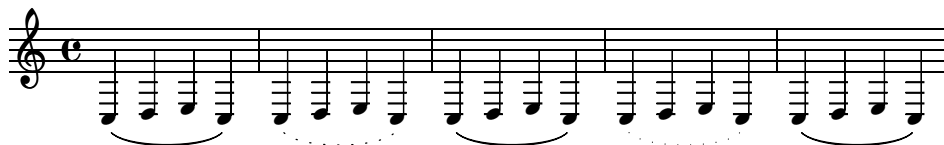
'slur-beautiful.ly':

Similarly, the curvature of a slur is adjusted to stay clear of note heads and stems. When that would increase the curvature too much, the slur is reverted to its default shape. The threshold for this decision is in Slur's object property `beautiful`. It is loosely related to the enclosed area between the slur and the notes. Usually, the default setting works well, but in some cases you may prefer a curved slur when LilyPond decides for a vertically moved one. You can indicate this preference by increasing the `beautiful` value.



'slur-dash.ly':

You can print different kinds of slurs (dotted, dashed, etc).



'slur-manual.ly':

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



'slur-minimum-length.ly':

You can set the minimum length of a slur.



`'slur-shape.ly':`

Slurs become flatter as they grow longer.



`'slur-ugly.ly':`

You can get ugly slurs, if you want.



`'smart-transpose.ly':`

Here's a copy of my feature request :

Your task, if you accept it is to implement a `\smarttranspose` command>> that would translate such oddities into more natural notations. Double accidentals should be removed, as well as E-sharp (-> F), bC (-> B), bF (-> E), B-sharp (-> C).

You mean like this. (Sorry 'bout the nuked indentation.)

Modified to use the standard transpose mechanism. The question is how useful these enharmonic modifications are. Mats B.



`'spacing-2.ly':`

1. Upon stretching every note should stretch according to duration.
2. 8th notes should be spaced equidistantly.



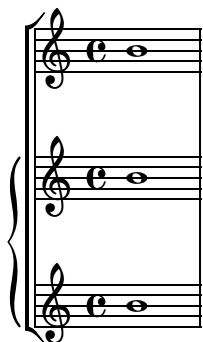
`'spanner-after-break-tweak.ly':`

To selectively tweak spanners after the linebreaking process, Scheme code must be used. In this simple example, the tie after the line break is moved around.



`'staff-bracket.ly':`

Here's an example of staff brackets.



`'staff-container.ly':`

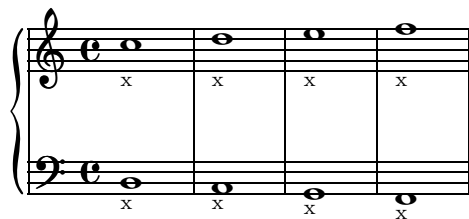
Container By splitting the grouping (`Axis_group-engraver`) and creation functionality into separate contexts, you can override interesting things.

Notation like this is used in modern scores. Note that LilyPond is not especially prepared for it: the clefs and time-signatures don't do what you would expect.



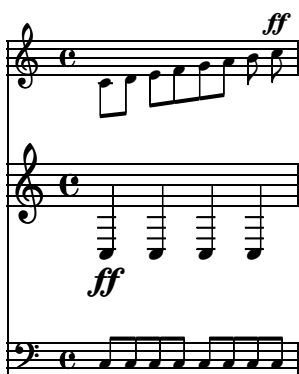
`'staff-lines.ly':`

Staff symbol property set workaround.



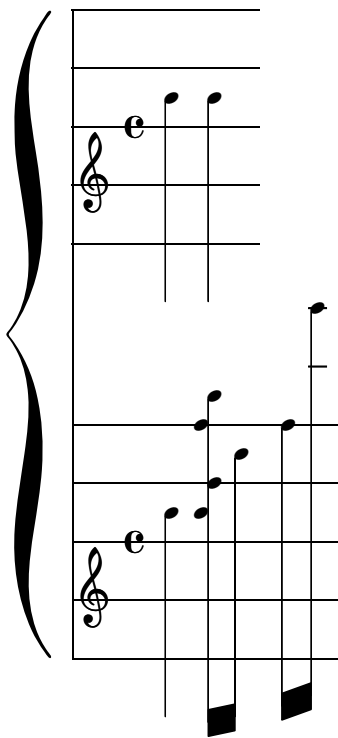
`'staff-size.ly':`

Setting staff sizes is a little clumsy. There are two options: using `StaffContainer` and `override/revert`, or `\applyoutput`. Both methods are shown in this example.



`'staff-space.ly':`

Setting staff space on a staff.



`'stem-centered.ly':`

Mensural note heads.



`'stem-cross-staff.ly':`

There is no support for putting chords across staves. You can get this result by increasing the length of the stem in the lower stave so it reaches the stem in the upper stave, or vice versa.



`'stem-extend.ly':`

You can stop LilyPond from extending stems to the center line.



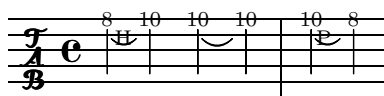
`'stem-length.ly':`

You can alter the length of stems.



`'tablature-hammer.ly':`

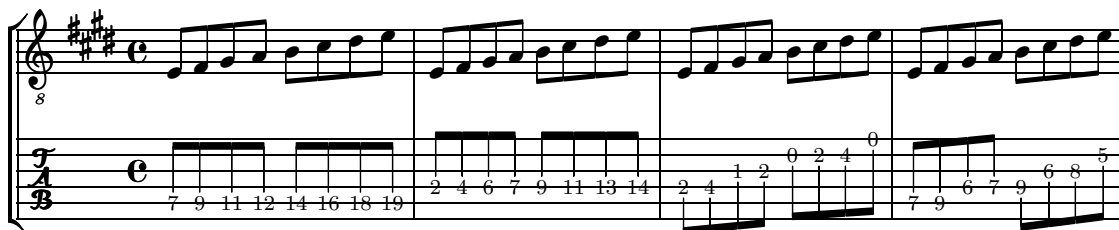
You can fake a hammer in tablature with slurs.



`'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.



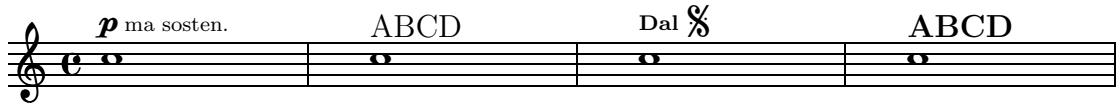
`'text-spanner.ly':`

You can make LilyPond print text spanners.



`'textscript.ly':`

Test font selection and scm text markup.



`'tie-cross-voice.ly':`

Cross voice ties can be faked by blanking noteheads.



`'tie-sparse.ly':`

Setting sparseTies causes only one tie to be generated per chord pair.



`'time-signature-double.ly':`

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



`'time.ly':`

Old time signatures. For further information, consult the file.





'timing.ly':

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





'to-xml.ly':

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

```

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

```

```

<duration
  log="2"
  dots="0"
  numer="1"
  denom="1">
</duration>
<pitch
  octave="0"
  notename="4"
  alteration="0">
</pitch>
</music>
</music>
</music>
</music>
</music>

```



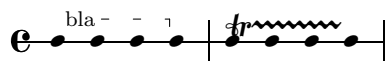
‘transposition.ly’:

Transposition test file.



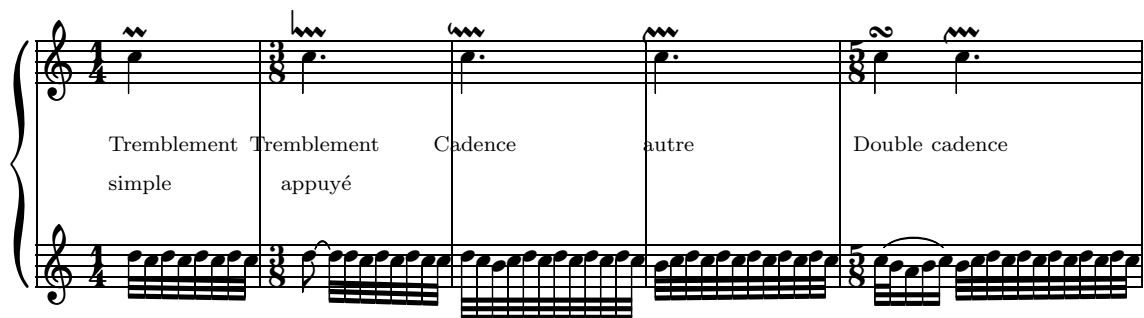
‘trill.ly’:

Show trill line type.



‘trills.ly’:

Document trills, pralls and turns.



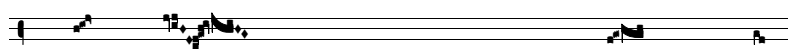
`'unfold-all-repeats.ly':`

The standard function `unfold-repeats` will recursively unfold all repeats for correct MIDI output. Thanks to Rune Zedeler.



`'vaticana.ly':`

Ancient Vaticana Vaticana ligature test.



Al- le- lu- ia.

`'version-output.ly':`

By putting the output of `lilypond-version` into a lyric, we can print the version number in a score, or a lilypond-book document.

Processed with LilyPond version 2.1.0

`'vertical-extent.ly':`

Vertical extents may be overridden by `minimumVerticalExtent`, `extraVerticalExtent`, and `verticalExtent`. These are normal property values, and are written into the grob when the associated context finishes, so using it in `\property` works.

