LilyPond

El tipografiador de música

Learning Manual

El equipo de desarrollo de LilyPond

(For LilyPond version 2.11.39)

Índice General

Preámbulo

Debió ser en el transcurso de un ensayo de la EJE (Joven Orquesta de Eindhoven), allá por 1995 cuando Jan, uno de los violistas chiflados, le habló a Han-Wen, uno de los trompistas distorsionados, acerca del gran proyecto en que estaba trabajando. Era un sistema automático para imprimir música (para ser exactos se trataba de MPP, un preprocesador para MusiXTeX). Ocurrió entonces que Han-Wen quiso imprimir unas particellas sacadas de una partitura, y así empezó a echarle un vistazo al programa, y en seguida se quedó estancado. Decidieron que MPP era un callejón sin salida. Después de muchísimo filosofar y de montañas de encendidas conversaciones por correo electrónico, Han-Wen inició el proyecto LilyPond en 1996. Esta vez fue Jan quien resultó absorbido por el nuevo proyecto de Han-Wen.

En ciertos aspectos, desarrollar un programa de ordenador es como aprender a tocar un instrumento. Al principio es divertido descubrir cómo funciona, y supone un divertido reto intentar aquello que no eres capaz de hacer. Una vez pasado el entusiasmo inicial, hay que practicar más y más. Las escalas y los estudios pueden llegar a aturdir, y si no está motivado por otras personas (profesores, directores o el público) uno siempre está tentado de abandonarlo. Uno persevera y, poco a poco, tocar se convierte en parte de la vida de uno. Algunos días se acoge de forma natural, y es estupendo, y otros simplemente la cosa no funciona, pero uno sigue tocando día tras día.

Igual que hacer música, trabajar en LilyPond puede ser un trabajo muy duro y hay días en que uno se siente como pisando un hormiguero. A pesar de todo, se ha convertido en parte de nuestra vida y seguimos haciéndolo. Con toda probabilidad la motivación más importante es que nuestro programa realmente hace algo útil por las personas. Cuando navegamos por la red encontramos mucha gente que utiliza LilyPond y produce unas partituras impresionantes con él. De esta observación se desprende una sensación algo irreal, pero muy agradable.

Nuestros usuarios no sólo nos transmiten buenas vibraciones por usar el programa, también muchos de ellos nos ayudan enviando sugerencias e informes de fallo, por ello nos gustaría agradecer a todos los usuarios que nos han enviado estos informes, emitido sugerencias o contribuido a LilyPond de cualquier otra forma.

Tocar e imprimir música es algo más que una bonita analogía. Programar juntos es muy divertido, y ayudar a las personas es algo profundamente gratificante, pero en último término trabajar en LilyPond es una forma de expresar nuestro profundo amor por la música. ¡Ojalá le ayude a elaborar montañas de preciosas partituras!

Han-Wen y Jan

Utrecht/Eindhoven, Holanda, julio de 2002.

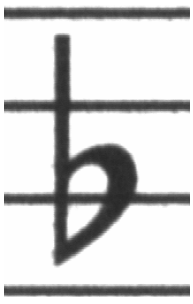
1 Introducción

1.1 Grabado

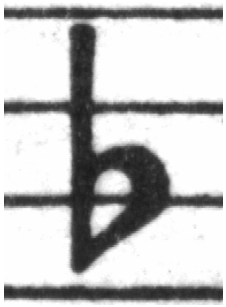
El arte de la tipografía musical se conoce como *grabado (en plancha)*. El término deriva del proceso tradicional de la impresión musical. Hace sólo unas décadas, la música impresa se hacía estampando la música sobre planchas de zinc o estaño de forma invertida como en un espejo. Después la plancha se entintaba y las depresiones causadas por los cortes y estampados retenían la tinta. Al presionar una hoja de papel sobre la plancha, se formaba una imagen. El estampado y cortado se hacía completamente a mano. Cualquier corrección era muy fastidiosa de realizar, si es que era posible hacerla siquiera, así que el grabado tenía que quedar perfecto a la primera. El grabado era una habilidad altamente especializada; un artesano necesitaba unos cinco años de preparación antes de poder ostentar el título de maestro grabador, y se necesitaban otros cinco años de experiencia para ser un auténtico experto.

Hoy en día, toda la música impresa nueva se produce con ordenadores. Esto tiene unas ventajas evidentes: las copias son más baratas de producir y el trabajo editorial se puede repartir por correo electrónico. Desgraciadamente la penetrante utilización de ordenadores también ha hecho disminuir la calidad gráfica de las partituras. Las impresiones de ordenador tienen un aspecto insulso y mecánico, lo que hace que sea desagradable tocar a partir de ellas.

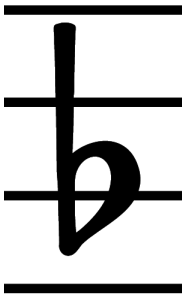
Las imágenes siguientes ilustran la diferencia entre el grabado tradicional y la salida típica de ordenador, y la tercera imagen muestra cómo LilyPond imita el aspecto tradicional. La imagen de la izquierda presenta el dibujo escaneado de un símbolo de bemol sacado de una edición publicada en el año 2000. La del centro es un símbolo procedente de una edición de Bärenreiter grabada a mano de la misma música. La de la izquierda ilustra los típicos puntos débiles de la impresión por ordenador: las líneas del pentagrama son muy delgadas, el peso del símbolo del bemol es también demasiado ligero como las líneas del pentagrama, y tiene una apariencia rectilínea con esquinas afiladas. En contraste, el bemol de Bärenreiter tiene una apariencia redonda, pesada, casi voluptuosa. Nuestro símbolo del bemol se diseñó según éste, entre otros. Es de forma redondeada y su peso está en armonía con el grosor de nuestras líneas de pentagrama, que son asimismo mucho más gruesas que las de la edición por ordenador.



Henle (2000)

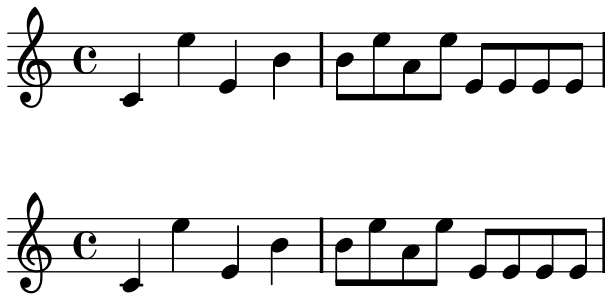


Bärenreiter (1950)



Tipografía Feta de LilyPond (2003)

Tratándose del espaciado, la distribución del espacio debe reflejar las duraciones que hay entre las notas. Sin embargo muchas partituras modernas se atañen a las duraciones con precisión matemática, lo que lleva a unos resultados bastante pobres. En el siguiente ejemplo se muestra un ejemplo dos veces: una utilizando espaciado matemáticamente exacto, y otra con ciertas correcciones. ¿Puede adivinar cuál es cuál?



Cada uno de los dos compases de este fragmento tiene solamente notas de duración constante. El espaciado debería reflejarlo. Desgraciadamente el ojo nos engaña un poco; no solamente percibe la distancia entre las cabezas de las notas, sino que tiene también en cuenta la distancia entre las plicas. Como resultado, las notas de una combinación plica arriba/plica abajo se tendrían que separar más, y las notas de una combinación plica abajo/plica arriba deberían juntarse, todo ello dependiendo de las posiciones combinadas de las notas. Los dos compases de arriba están impresos con esta corrección y los de abajo sin ella, formando grupos de notas pegadas con plica abajo/plica arriba.

Los músicos están normalmente más concentrados en tocar que en estudiar el aspecto de una partitura, y por ello las pequeñeces sobre los detalles tipográficos pueden parecer académicas. Pero no lo son. En las partituras más largas con ritmos monótonos, las correcciones de espaciado llevan a sutiles variaciones en la disposición de cada una de las líneas dándoles una especie de firma visual distintiva. Sin esta firma, todas las líneas parecerían iguales, y se convertirían en un laberinto. Si un músico aparta la mirada o tiene un lapsus de concentración, las líneas podrían perder su lugar sobre el papel.

De forma similar, la fuerza visual de unos símbolos pesados sobre gruesas líneas de pentagrama se sostiene mejor cuando el lector se aleja del papel, por ejemplo cuando está sobre un atril. Una distribución cuidadosa del espacio blanco permite disponer la música muy apretada sin que los símbolos se toquen unos a otros. El resultado reduce a un mínimo las vueltas de página, lo que es una gran ventaja.

Ésta es una característica normal del arte tipográfico. La disposición de la página tiene que ser bonita, no sólo por sí misma, sino sobre todo porque así ayuda al lector en su tarea. Para los materiales destinados a la interpretación, como las partituras, esto es de una importancia doble: los músicos tienen una capacidad de concentración limitada. Cuanta menos atención necesiten para el acto de leer, más se pueden dedicar al acto de tocar la música. Dicho de otra forma: una mejor tipografía se traduce en una mejor interpretación.

Estos ejemplos demuestran que la tipografía musical es un arte sutil y complejo, y que su elaboración requiere una experiencia considerable, que los músicos no suelen tener. LilyPond representa nuestro esfuerzo para llevar la excelencia visual de la música grabada a mano a la era de la informática, y ponerla a disposición de los músicos normales. Hemos ido afinando nuestros algoritmos, diseños de tipografía y preferencias del programa para producir una impresión cuya calidad se equipara con la de las viejas ediciones que tanto nos gusta contemplar y de las que tanto nos gusta tocar.

1.2 Grabado automático

¿Cómo nos las arreglamos para implementar la tipografía? Si un artesano necesita más de diez años para convertirse en un auténtico maestro ¿cómo vamos a poder nosotros, simples «hackers», escribir un programa que les quite el trabajo?

La respuesta es: ¡no podemos! Puesto que la tipografía se fundamenta en el juicio humano sobre la apariencia, nunca se puede sustituir completamente a las personas. Sin embargo, se puede automatizar gran parte del trabajo más duro y repetitivo. Si LilyPond resuelve la mayoría de las situaciones comunes de forma correcta, esto ya será una tremenda mejoría sobre los programas existentes. El resto de los casos se podrán afinar a mano. Con el transcurso de los

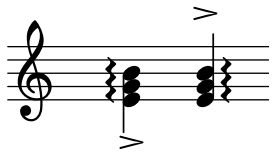
años, el software se puede refinar para que haga un mayor número de cosas de forma automática, de tal forma que los ajustes manuales tienden a ser cada vez menos necesarios.

Cuando empezamos, escribimos el programa LilyPond completamente en el lenguaje C++; la funcionalidad del programa quedaba como esculpida en piedra por los desarrolladores. Este esquema resultó no ser muy satisfactorio por una serie de motivos:

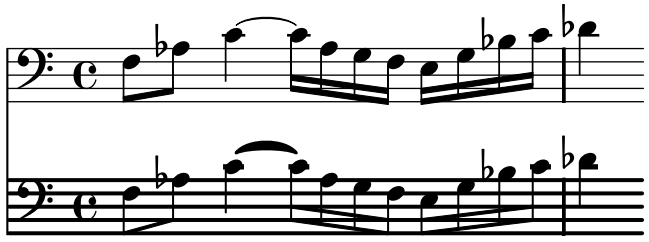
- Cuando LilyPond comete fallos, los usuarios tienen la necesidad de superar las decisiones de formateo. Por ello el usuario debe tener acceso al motor de formateo. De aquí que no podamos dejar establecidas las reglas y valores durante la compilación, sino que los usuarios deben poder acceder a ellos durante la ejecución del programa.
- El grabado de música es cosa de juicio visual y por ello es cuestión de gustos. A pesar de saber tanto como creemos saber, los usuarios pueden no estar de acuerdo con nuestras decisiones personales. Por tanto la definición del estilo tipográfico también debe estar al alcance del usuario.
- Por último, estamos continuamente refinando los algoritmos de formateo y por tanto necesitamos un enfoque flexible para las reglas. El lenguaje C++ fuerza un cierto método para agrupar las reglas que no encaja bien con la manera de funcionar de la notación musical.

Estos problemas se han solucionado integrando un intérprete del lenguaje Scheme y reescribiendo parte del código de LilyPond en Scheme. La actual arquitectura de formateo se construye alrededor del concepto de objetos gráficos, descrita por variables y funciones de Scheme. Esta arquitectura puede tratar al mismo tiempo con las reglas de formateo, el estilo tipográfico y las decisiones de formateo individuales. El usuario tiene acceso directo a la mayor parte de estos controles.

Las variables de Scheme controlan las decisiones de formateo. Por ejemplo, muchos objetos gráficos tienen una variable de dirección que codifica la elección entre arriba y abajo (o izquierda y derecha). Aquí puede ver dos acordes con acentos y signos de arpegio. En el primer acorde los objetos gráficos tienen todas sus direcciones hacia abajo (o hacia la izquierda). El segundo acorde tiene todas las direcciones hacia arriba (o hacia la derecha).

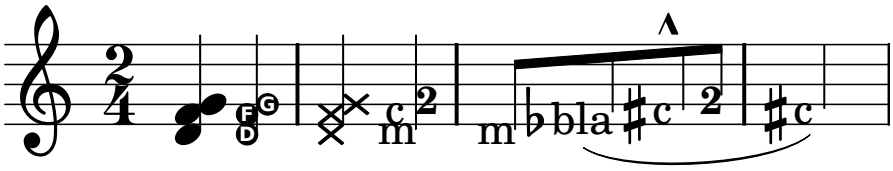


El proceso de formatear una partitura consiste en leer y escribir las variables de los objetos gráficos. Ciertas variables tienen un valor predefinido. Por ejemplo, el grosor de muchas líneas (una característica del estilo tipográfico) son variables con un valor preestablecido. Podemos alterar este valor libremente dando así a nuestra partitura una impresión tipográfica distinta.



Las reglas de formateo también son variables que están predefinidas: cada objeto tiene unas variables que contienen procedimientos. Estos procedimientos realizan el trabajo real de formateo y sustituyéndolos por otros podemos alterar el aspecto de los objetos. En el siguiente

ejemplo, la regla que define cómo se dibuja la cabeza de una nota se altera durante el transcurso del fragmento musical.



1.3 ¿Qué símbolos grabar?

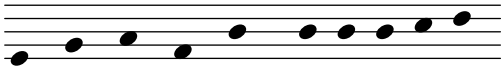
El proceso de formateo toma las decisiones sobre dónde colocar los símbolos. Sin embargo esto sólo se puede hacer una vez que se ha decidido *qué* símbolos han de imprimirse, o dicho de otro modo: qué notación utilizar.

La notación musical común es un sistema de registro de música que ha venido evolucionando desde hace mil años. La forma que se usa en nuestros días data de los primeros tiempos del Renacimiento. Aunque la forma básica (es decir: puntos sobre una pauta de cinco líneas) no ha cambiado, los detalles continúan evolucionando para expresar todas las innovaciones de la notación contemporánea. Por tanto abarca unos quinientos años de música. Sus aplicaciones se extienden sobre un amplio rango que abarca desde melodías monofónicas hasta monstruosos contrapuntos para gran orquesta.

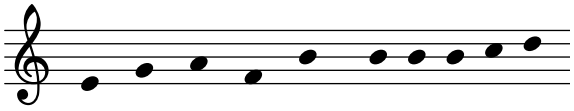
¿Cómo podemos tratar con una bestia de tantas cabezas, y obligarla a que se encierre dentro de los límites de un programa de ordenador? Nuestra solución es trocear el problema de la notación (por oposición al grabado, esto es, a la tipografía) en fragmentos digeribles y más fáciles de programar: cada tipo de símbolo se maneja por un módulo separado que recibe el nombre de «plug-in». Cada «plug-in» es completamente modular e independiente, de forma que puede desarrollarse y mejorarse por separado. Estos «plug-ins» se llaman `engraver`s (grabadores), por analogía con los artesanos que traducen las ideas musicales a símbolos gráficos.

En el siguiente ejemplo vemos cómo comenzamos con un plug-in para las cabezas de las notas, el `Note_heads_engraver`.

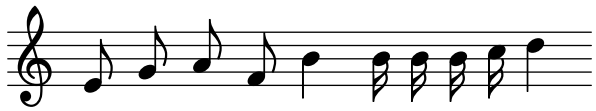
A continuación un `Staff_symbol_engraver` (grabador del pentagrama) añade las líneas de la pauta.



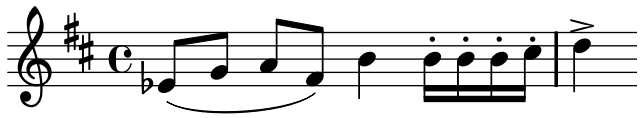
El `Clef_engraver` (grabador de la clave) define un punto de referencia para la altura de las notas en el pentagrama.



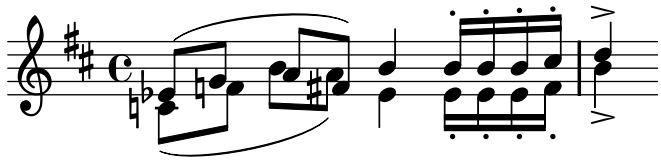
y el `Stem_engraver` (grabador de las plicas) añade las plicas.



El `Stem_engraver` (grabador de plicas) recibe una notificación cuando llega una cabeza. Cada vez que se ve una cabeza (o más, si es un acorde), se crea un objeto plica y se conecta a la cabeza. Añadiendo grabadores para las barras, ligaduras, acentos, alteraciones, líneas divisorias, indicación de compás y armadura conseguimos una notación completa.



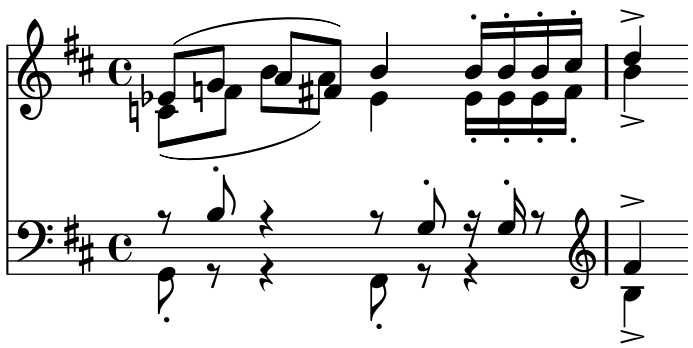
Este sistema funciona bien para la música monofónica, pero ¿y con la polifonía? En notación polifónica muchas voces pueden compartir el mismo pentagrama.



En esta situación, las alteraciones y la pauta se comparten, pero las plicas, ligaduras, barras, etc. son propias de cada voz. Por tanto los grabadores han de agruparse. Los grabadores de cabezas, plicas, ligaduras, etc. se unen en un grupo llamado 'Contexto de voz', mientras que los grabadores de la armadura, alteraciones, compás, etc. van a un grupo que se llama 'Contexto de la pauta'. En el caso de la polifonía, un único Contexto de pauta contiene más de un Contexto de voz. De forma semejante, varios Contextos de pauta pueden agruparse en un único Contexto de partitura. El Contexto de partitura es el contexto de notación de más alto nivel.

Véase también

Referencia del programa: `Contexts`.



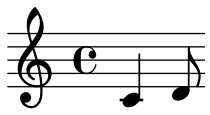
1.4 Representación musical

Idealmente el formato de entrada para cualquier sistema de formateo de alto nivel es una descripción abstracta del contenido. En este caso, eso constituiría la propia música, lo que plantea un tremendo problema: ¿cómo podemos definir qué es realmente la música? En lugar de intentar hallar una respuesta, le hemos dado la vuelta a la pregunta. Escribimos un programa capaz de producir partituras y ajustamos el formato para que sea tan escueto como sea posible. Cuando el formato ya no puede reducirse más, por definición nos habremos quedado con el contenido musical propiamente dicho. Nuestro programa sirve como definición formal de un documento musical.

La sintaxis también es el interfaz de usuario de LilyPond, así que es fácil teclear

```
c'4 d'8
```

un Do1 (Do central) negra, y un Re1 (el Re por encima del Do central) corchea.

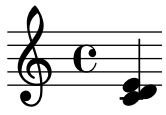


A una escala microscópica, dicha sintaxis es fácil de utilizar. A una escala mayor, la sintaxis también requiere una estructura. ¿De qué otra forma podríamos introducir piezas complejas como sinfonías u óperas? La estructura se forma mediante el concepto de expresiones musicales: al combinar pequeños fragmentos de música dentro de otros mayores, se pueden expresar ideas musicales más complejas. Por ejemplo

```
c4
```

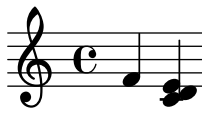
Los acordes se pueden construir encerrando las notas entre << y >>

```
<<c4 d4 e4>>
```



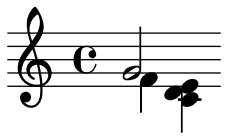
Esta expresión se coloca en secuencia encerrándola dentro de llaves `{ ... }`

```
{ f4 <<c4 d4 e4>> }
```



Lo anterior, a su vez también es una expresión, y por ello se puede combinar de nuevo con otra expresión simultánea (una blanca) usando <<, \\, y >>

```
<< g2 \\ { f4 <<c4 d4 e4>> } >>
```

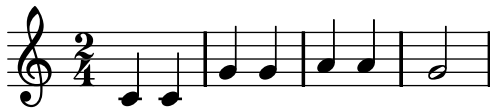


Las mencionadas estructuras recursivas se pueden especificar de forma nítida y formal dentro de una gramática independiente del contexto. El código de análisis también se genera a partir de esta gramática. En otras palabras, la sintaxis de LilyPond está definida de una forma clara y sin ambigüedades.

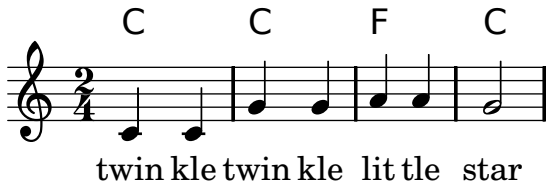
Los interfaces de usuario y la sintaxis son lo que la persona ve y con lo que trata principalmente. En parte, son fruto de preferencias personales y como tales están sujetas a mucha discusión. Aunque las discusiones sobre el gusto tienen su mérito, no son demasiado productivas. Dentro de la escena global de LilyPond, la sintaxis de la entrada tiene una importancia relativamente pequeña: inventarse una sintaxis elegante es fácil, pero escribir un código de formateo decente es mucho más difícil. Esto también queda ilustrado por la cantidad de líneas de código de los componentes respectivos: el análisis y la representación se llevan menos del 10% del código fuente.

1.5 Aplicaciones de ejemplo

Escribimos LilyPond como un experimento de cómo condensar el arte del grabado de música dentro de un programa de ordenador. Gracias a todo este duro trabajo, el programa ahora se puede usar para hacer trabajos útiles. La aplicación más sencilla es imprimir notas.



Añadiendo los nombres de acordes y la letra, obtenemos una hoja guía de acordes (lead sheet).

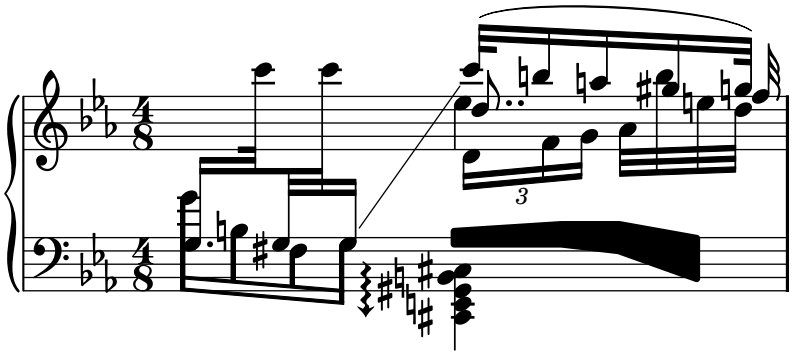


También se pueden imprimir notación polifónica y música para piano. El ejemplo siguiente combina algunas otras construcciones exóticas.

Screech and boink

Random complex notation

Han-Wen Nienhuys



Todos los fragmentos mostrados se han escrito a mano, pero esto no es necesariamente así. Puesto que el motor de formateo es casi completamente automático, puede servir como medio de salida para otros programas que manipulan música. Por ejemplo, se puede usar también para convertir bases de datos de fragmentos musicales en imágenes con destino a páginas web y presentaciones multimedia.

Este manual también es un ejemplo de aplicación: el formato de entrada es texto sencillo, y por ello se puede empotrar fácilmente dentro de otros formatos basados en texto, como LaTeX, HTML, o en el caso concreto de este manual, Texinfo. A través de un programa especial, los fragmentos de entrada se pueden sustituir por imágenes musicales dentro de los archivos de salida PDF o HTML resultantes. Esto convierte la tarea de mezclar música y texto dentro de los documentos, en algo muy sencillo.

1.6 Sobre el presente manual

Hay dos manuales sobre LilyPond: el *manual del usuario* (este mismo documento) y el de *utilización del programa*.

Manual del usuario

El manual se divide en tres libros.

Learning manual

Este libro explica cómo empezar el aprendizaje de LilyPond, así como algunos conceptos clave en términos sencillos. Se recomienda leer estos capítulos secuencialmente.

- *Capítulo 2 [Tutorial], página 11* da una amable introducción a la tipografía musical. Los usuarios que se acercan por primera vez deben comenzar por aquí.
- *Capítulo 3 [Juntándolo todo], página 35* explica algunos conceptos generales sobre el formato de los archivos de entrada de LilyPond. Si no está seguro de dónde colocar un comando ¡lea este capítulo!
- *Capítulo 4 [Trabajar en proyectos de LilyPond], página 43* trata los usos prácticos de LilyPond y cómo evitar ciertos problemas bastante comunes.
- *Capítulo 5 [Trucar la salida], página 52* muestra la manera de cambiar el grabado por omisión que produce LilyPond.

Notation reference

Este libro explica todas las instrucciones de LilyPond que producen notación impresa. Da por supuesto que el lector está familiarizado con los conceptos del manual de aprendizaje.

Más información

Existen otras fuentes de información que pueden resultar muy útiles.

- El glosario explica los términos musicales, e incluye traducciones a varios idiomas. Es un documento aparte, disponible en HTML y en PDF. Si no está familiarizado con la notación o la terminología musicales (sobre todo si no es un anglófono nativo), le recomendamos que consulte el glosario.
- Los fragmentos de código (snippets) son una enorme colección de pequeños ejemplos que ejemplifican toda clase de consejos, trucos y funcionalidades especiales de LilyPond. La mayor parte de estos fragmentos de código también se pueden encontrar en el Archivo de fragmentos de código (snippets) de LilyPond. Este sitio web también dispone de un manual de LilyPond en el que se pueden realizar búsquedas.
- La Referencia del programa es un conjunto de páginas HTML con una tupida red de enlaces cruzados, que documentan al detalle el meollo de todas y cada una de las clases, objetos

y funciones de LilyPond. Se produce directamente a partir de las definiciones de formateo que se utilizan.

Casi toda la funcionalidad de formateo que se emplea internamente, se encuentra disponible para el usuario de forma directa. Por ejemplo, todas las variables que controlan los valores de grosor, distancias, etc., se pueden cambiar dentro de los archivos de entrada. Hay un enorme número de opciones de formateo, y todas ellas se describen en este documento. Cada sección del manual de notación tiene una subsección **Véase también**, que hace referencia a la documentación generada. En el documento HTML, estas subsecciones llevan enlaces que se pueden pulsar.

Cuando ya sea un usuario con experiencia podrá usar el manual como referencia: hay un índice muy completo[1], pero el documento también está disponible en una sola página HTML, en la que es fácil buscar cualquier cosa utilizando la función de búsqueda de su navegador de web.

En todos los documentos HTML que tienen fragmentos de música incrustados, la entrada de LilyPond que se utilizó para producir dicha imagen se puede ver pulsando con el ratón sobre la imagen.

La localización exacta de los archivos de documentación que hemos mencionado puede variar de un sistema a otro. En ocasiones este manual hace referencia a archivos de inicialización y de ejemplo. A lo largo del manual, nos referimos a archivos de entrada por su ruta relativa respecto de directorio de nivel más alto de los archivos de código fuente. Por ejemplo, '`input/lsr/dirname/bla.ly`' puede referirse al archivo '`lilypond2.x.y/input/lsr/dirname/bla.ly`'. En los paquetes binarios para la plataforma Unix, normalmente la documentación y los ejemplos se encuentran en algún lugar dentro de '`/usr/share/doc/lilypond/`'. Los archivos de inicialización, como por ejemplo '`scm/lily.scm`', o '`ly/engraver-init.ly`', se encuentran normalmente en el directorio '`/usr/share/lilypond/`'.

Por último, este y el resto de los manuales están disponibles en línea tanto como archivos PDF como HTML en el sitio web, que encontrará en http://www.lilypond.org/.

[1] Si está buscando algo y no lo encuentra en el manual, eso se considera un bug (fallo). En este caso le rogamos que envíe un informe de fallo.

2 Tutorial

Este tutorial comienza con una introducción al lenguaje musical LilyPond y cómo producir música impresa. Después de este primer contacto, explicaremos cómo crear notación musical usual.

2.1 Primeros pasos

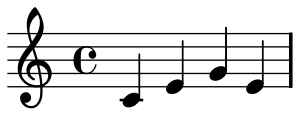
Esta sección le ofrece una introducción básica al trabajo con LilyPond.

2.1.1 Compilar un archivo

El primer ejemplo demuestra cómo empezar a trabajar con LilyPond. Para crear una partitura, escribimos un archivo de texto que especifica la notación. Por ejemplo, si escribimos

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

el resultado tiene este aspecto



Advertencia: Todo el código que se da como entrada a LilyPond tiene que ir entre `{` **llaves** `}`. Las llaves deberían también estar rodeadas por espacios a no ser que se encuentren al principio o al final de una línea, para evitar ambigüedades. Es posible que se omitan en algunos ejemplos del presente manual ¡pero no las omita en su propia música!

Además la entrada de LilyPond es **sensible a las mayúsculas**. `{ c d e }` es una entrada válida; `{ C D E }` producirá un mensaje de error.

Introducir música y ver la salida

En esta sección vamos a explicar qué órdenes hay que ejecutar y cómo, para ver o imprimir el resultado.

MacOS X

Si hace doble clic sobre `LilyPond.app`, se abrirá con un archivo de ejemplo. Guárdelo, por ejemplo, como '`prueba.ly`' en el Escritorio, y a continuación procéselo con la orden de menú '`Compilar > Componer archivo`'. El PDF resultante se mostrará en la pantalla.

Advierta que la primera vez que ejecute LilyPond, tardará un minuto o dos porque todas las tipografías del sistema han de ser analizadas previamente.

Para posteriores usos de LilyPond, debería comenzar eligiendo 'Nuevo' o 'Abrir'. Tiene que grabar el archivo antes de componerlo tipográficamente. Si se produce algún error durante el proceso, observe la ventana del registro.

Windows

En Windows, si hace doble clic sobre el icono de LilyPond que está en el escritorio, se abrirá un sencillo editor de texto con un archivo de ejemplo. Guárdelo, por ejemplo, con el nombre '`prueba.ly`' en el escritorio y después haga doble clic sobre el icono del archivo para procesarlo (el icono tiene la forma de una nota). Transcurridos unos segundos, obtendrá un archivo '`prueba.pdf`' en el escritorio. Haga doble clic sobre este archivo PDF para ver la partitura

compuesta tipográficamente. Un método alternativo para procesar el archivo '`prueba.ly`' es arrastrarlo y soltarlo sobre el icono de LilyPond utilizando el ratón.

Para editar un archivo '`.ly`' existente, haga clic sobre él con el botón derecho del ratón y elija "Edit source" (editar el archivo fuente). Para empezar con un archivo vacío, arranque el editor como se describe más arriba y elija "New" (nuevo) del menú "File" (archivo).

Al hacer doble clic sobre el archivo no sólo se obtiene como resultado un archivo PDF, sino también un archivo '`.log`' que contiene cierta información acerca de lo que LilyPond ha hecho con el archivo. Si se produce algún error, examine este archivo de registro.

Tenga en cuenta que hay disponibles varios editores de texto alternativos con un mejor apoyo para la realización de documentos de LilyPond, consulte el manual de utilización del programa, `Apoyo respecto de los editores de texto` para ver más información.

Unix

Comience abriendo una ventana de terminal e iniciando un editor de texto. Por ejemplo, puede abrir un xterm y ejecutar `joe`[1]. En su editor de texto, introduzca la siguiente entrada y guarde el archivo como '`prueba.ly`'

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

Para procesar '`prueba.ly`', haga lo siguiente:

```
lilypond prueba.ly
```

Verá algo parecido a:

```
lilypond prueba.ly
GNU LilyPond 2.10.0
Procesando `prueba.ly'
Analizando...
Interpretando la música... [1]
Preprocesando los objetos gráficos...
Calculando los cortes de línea... [2]
Escribiendo la página de salida en `prueba.ps'...
Convirtiendo a `prueba.pdf'...
```

El resultado es el archivo '`prueba.pdf`' que podrá imprimir o ver con las utilidades estándar de su sistema operativo.[2]

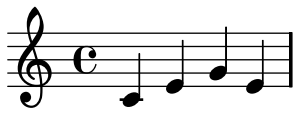
2.1.2 Notación sencilla

LilyPond añadirá ciertos elementos de notación de manera automática. En el siguiente ejemplo hemos especificado solamente cuatro alturas, pero LilyPond ha añadido la clave, el compás y las duraciones.

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

[1] Existen macros para los adictos a VIM, y hay un `LilyPond-mode` (modo de LilyPond) para los adictos a Emacs. Si no han sido instalados aún, consulte el archivo '`INSTALL.txt`'. El entorno de edición más sencillo es '`LilyPondTool`'. Consulte el manual de utilización del programa, `Apoyo respecto de los editores de texto` para más información.

[2] Si su sistema no dispone de dichas herramientas, puede probar Ghostscript, un programa de libre disposición para ver e imprimir archivos PDF y PostScript.

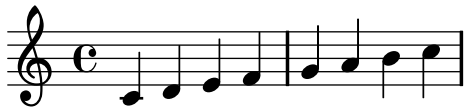


Este comportamiento se puede modificar, pero en general estos valores automáticos son adecuados.

Alturas

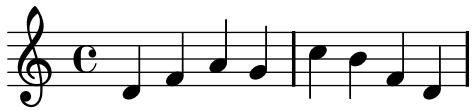
La manera más sencilla de introducir las notas es mediante la utilización del modo `\relative`. En este modo se supone que el `intervalo` entre la nota anterior y la actual se encuentra dentro de una `cuarta`. Comenzamos introduciendo la pieza musical más elemental, una `escala`.

```
\relative c' {
  c d e f
  g a b c
}
```



La nota inicial es `Do central`. Cada nota sucesiva se encuentra dentro de una cuarta de la nota previa (en otras palabras: la primera '`c`' es el Do más cercano al Do central; a éste le sigue el Re más cercano a la nota previa, y así sucesivamente). Podemos crear melodías con intervalos mayores:

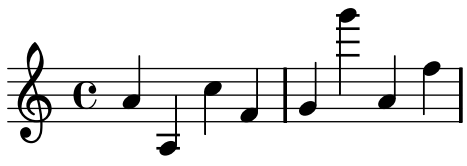
```
\relative c' {
  d f a g
  c b f d
}
```



Como puede observar, este ejemplo no comienza en el Do central. La primera nota (la '`d`') es el Re más cercano al Do central.

Para poner intervalos mayores de una cuarta, podemos elevar la octava añadiendo una comilla simple `'` (o apóstrofe) al nombre de la nota. También podemos bajar la octava adjuntando una coma `,` al nombre de la nota.

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

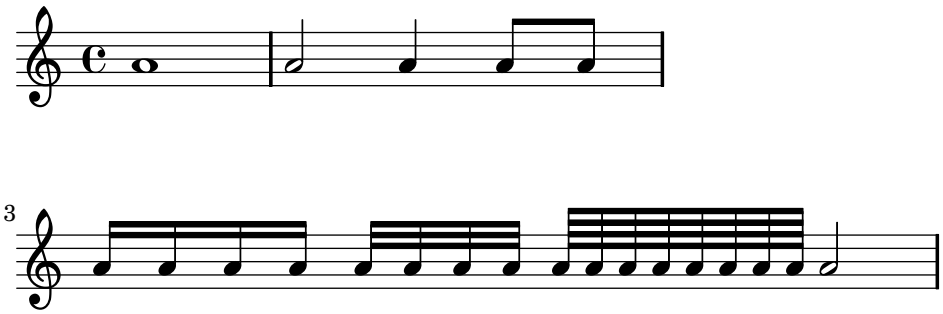


Para subir o bajar una nota en dos (¡o más!) octavas, utilizamos varias `''` ó `,,` (pero tenga cuidado de utilizar dos comillas simples `''` ¡y no una comilla doble `"` !) El valor inicial de `\relative c'` también puede modificarse de esta forma.

Duraciones (valores rítmicos)

La `duración` de una nota se especifica mediante un número después del nombre de la nota. '`1`' significa `redonda`, '`2`' significa `blanca`, '`4`' significa `negra` y así sucesivamente. Las barras de corchea se añaden automáticamente.

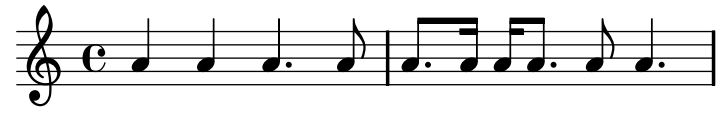
```
\relative c'' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a a2
}
```



Si no especifica una duración, se utiliza la duración previa para la nota siguiente. La figura por omisión de la primera nota es una negra.

Para crear notas con puntillo (véase `figura con puntillo`), añada un punto '`.`' al número de la duración.

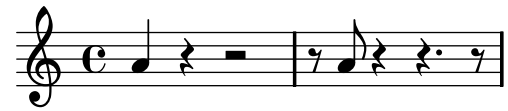
```
\relative c'' {
  a a a4. a8
  a8. a16 a a8. a8 a4.
}
```



Silencios

Un `silencio` se introduce igual que una nota, con el nombre '`r`':

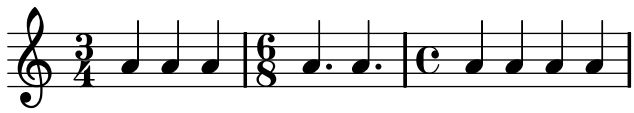
```
\relative c'' {
  a r r2
  r8 a r4 r4. r8
}
```



Indicación de compás

La `indicación de compás`) se puede establecer con la orden `\time` :

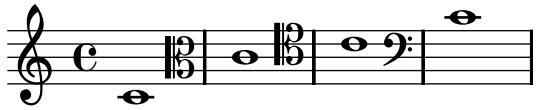
```
\relative c'' {
  \time 3/4
  a4 a a
  \time 6/8
  a4. a
  \time 4/4
  a4 a a a
}
```



Clave

La `clave` se puede establecer utilizando la orden `\clef` :

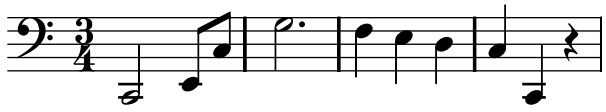
```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



Todo junto

He aquí un pequeño ejemplo que muestra todos los elementos anteriores juntos:

```
\relative c, {
  \time 3/4
  \clef bass
  c2 e8 c' g'2.
  f4 e d c4 c, r4
}
```



Véase también

Introducir alturas y duraciones
: véase el manual del usuario, `Alturas` y el manual del usuario, `Duraciones`.

Silencios
: véase el manual del usuario, `Silencios`.

Indicaciones de compás y otras instrucciones de tiempo
: véase el manual del usuario, `Indicación de compás`.

Claves
: véase el manual del usuario, `Clave`.

2.1.3 Trabajar sobre archivos de texto

Los archivos de entrada de LilyPond son como los archivos fuente de muchos lenguajes de programación corrientes. Son sensibles a las mayúsculas e insensibles al número de espacios Las expresiones se forman con llaves `{ }` y los comentarios se denotan por un signo de porcentaje (`%`) o por `%{ ... %}`.

Si la frase anterior no tiene sentido para usted ¡no se preocupe! A continuación explicaremos el significado de todos estos términos:

- **Sensible a las mayúsculas**: tiene importancia el hecho de que introduzca una letra en minúsculas (p.ej. `a, b, s, t`) o en mayúsculas (p.ej. `A, B, S, T`). Las notas son minúsculas: `{ c d e }` es una entrada válida; `{ C D E }` produciría un mensaje de error.
- **Insensible al número de espacios**: no importa cuántos espacios (o saltos de línea) añada. `{ c d e }` significa lo mismo que `{ c  d e }` y que

```
{
c                       d
  e }
```

  Por supuesto, el ejemplo anterior es difícil de leer. Una regla práctica es sangrar los bloques de código con un carácter de tabulación, o bien con dos espacios:

```
{
  c d e
}
```

- **Expresiones:** Todo fragmento de código de entrada para LilyPond ha de llevar **`{ llaves }`** antes y después de la entrada. Estas llaves le dicen a LilyPond que la entrada es una expresión musical unitaria, igual que los paréntesis '`()`' de las matemáticas. Las llaves deben ir rodeadas de un espacio a no ser que se encuentren al comienzo o al final de una línea, para evitar cualquier ambigüedad.

  Una función (como por ejemplo `\relative { }`) también es una expresión musical unitaria.
- **Comentarios**: Un comentario es una nota para el lector humano de la entrada musical; se ignora cuando esta entrada se analiza, de manera que no tiene ningún efecto sobre la salida impresa. Existen dos tipos de comentarios. El símbolo de porcentaje '`%`' introduce un comentario de línea; todo lo que se encuentra después de '`%`' en esa línea se ignora. Un comentario de bloque marca una sección entera de entrada musical como comentario. Todo lo que está encerrado dentro de `%{` y `%}` se ignora (pero los comentarios no pueden incluir otros comentarios). El siguiente fragmento muestra algunos posibles usos para los comentarios:

```
% a continuación van las notas de campanitas del lugar
  c4 c g' g a a g2

%{
    Esta línea y las notas que aparecen más abajo
```

```
        se ignoran, por estar dentro de un
        comentario de bloque.

        g g f f e e d d c2
    %}
```

Hay más trucos para elaborar archivos de entrada en el manual del usuario, `Sugerencias para escribir archivos de LilyPond`.

2.1.4 Cómo leer el tutorial

Como vimos en el manual del usuario, `Trabajar sobre archivos de texto`, la entrada de LilyPond debe estar rodeada de llaves `{ }` o de `\relative c'' { ... }`. Durante el resto del presente manual, la mayor parte de los ejemplos omitirán las llaves.

Si está leyendo la documentación HTML y quiere ver el código de LilyPond exacto que se utilizó para crear el ejemplo, sencillamente haga clic sobre la imagen. Si no está leyendo la versión en HTML, podría copiar y pegar la entrada que se muestra, pero **deberá** añadir `\relative c'' { }` de la siguiente manera:

```
\relative c'' {
  ... aquí va el ejemplo...
}
```

¿Por qué omitir las llaves? Casi todos los ejemplos del presente manual se pueden insertar en medio de un fragmento mayor de música. Para estos ejemplos no tiene ningún sentido añadir `\relative c'' { }` (no debería poner un `\relative` dentro de otro `\relative`), de forma que usted no podría copiar un ejemplo pequeño procedente de la documentación y pegarlo dentro de su propia pieza.

2.2 Notación en un solo pentagrama

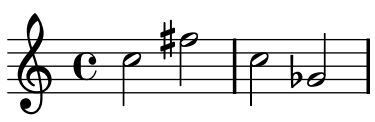
Esta sección es una introducción a la notación corriente que se utiliza para una voz o un pentagrama.

2.2.1 Nombres de nota relativos

Como hemos visto en el manual del usuario, `Notación sencilla`, LilyPond calcula la altura de cada nota de forma relativa respecto de la nota anterior[3]. Si no se añade ninguna marca de octava (`'` y `,`), se supone que cada altura se encuentra dentro de una cuarta desde la nota anterior.

LilyPond examina las alturas basándose en los nombres de las notas (dicho de otra forma: una cuarta aumentada *no* es lo mismo que una quinta disminuida. Si empezamos en un Do, un Fa sostenido situado a continuación se colocará por encima del Do, mientras que un Sol bemol se colocará por debajo de este mismo Do.

```
c2 fis
c2 ges
```



[3] Existe otra forma de introducir las alturas, el manual del usuario, `Nombres de nota absolutos`, pero en la práctica el modo relativo es mucho más cómodo y seguro de utilizar.

Véase también

Octavas relativas
: ver el manual del usuario, `Relative octaves`.

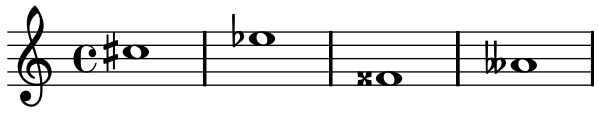
Comprobación de octava
: ver el manual del usuario, `Comprobación de la octava`.

2.2.2 Alteraciones accidentales y armaduras

Alteraciones accidentales

Una nota `sostenido` se hace añadiendo '`is`' al nombre, y una nota `bemol` añadiendo '`es`'. Como ha podido adivinar, un `doble sostenido` o `doble bemol` se hace añadiendo '`isis`' o '`eses`'[4]

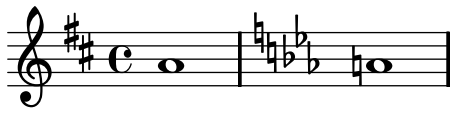
```
cis1 ees fisis, aeses
```



Armaduras

La armadura de la tonalidad se establece mediante la instrucción `\key` seguido de una nota y `\major` o `\minor`.

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



Advertencia: armaduras y alturas

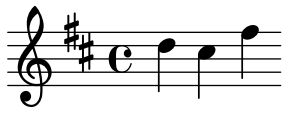
Para determinar si hay que imprimir una alteración accidental, LilyPond examina las notas y la armadura de la tonalidad. La armadura solamente afecta a las alteraciones *impresas*, ¡no a las propias notas! Ésta es una funcionalidad que frecuentemente confunde a los que se inician con el programa, por ello permítanos explicarla en detalle.

LilyPond hace una clara distinción entre el contenido musical y la presentación. La alteración (bemol, becuadro o sostenido) de una nota es parte de la altura, y por tanto es contenido musical. Si una alteración (un signo *impreso* de bemol, becuadro o sostenido) se imprime o no delante de la nota correspondiente, es una cuestión de presentación. La presentación es algo que sigue unas reglas, así que las alteraciones accidentales se imprimen automáticamente según dichas reglas. Las alturas de las notas en su música son obras de arte, por tanto no se añadirán automáticamente, y usted deberá introducir aquello que quiera oír.

En el siguiente ejemplo

[4] Esta sintaxis derivó de las convenciones de las lenguas nórdicas y germánicas para nombrar las notas como el alemán y el holandés. Para utilizar otros nombres para las alteraciones accidentales, vea el manual del usuario, `Nombres de las notas en otros idiomas`.

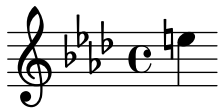
```
\key d \major
d cis fis
```



ninguna nota lleva una alteración impresa, pero de todas formas usted debe añadir la '`is`' a `cis` y a `fis`.

El texto '`e`' no significa "imprimir una bolita negra en la primera línea del pentagrama." Más bien significa: "hay una nota Mi natural." En la tonalidad de La bemol mayor, *lleva* una alteración accidental:

```
\key aes \major
e
```



Poner todas las alteraciones de forma explícita puede que requiera algo más de trabajo al teclear, pero la ventaja es que la transposición es más fácil, y las alteraciones se pueden imprimir siguiendo varias convenciones distintas. Consulte el manual del usuario, `Alteraciones accidentales automáticas` para ver ejemplos de cómo se pueden imprimir las alteraciones de acuerdo a reglas diferentes.

Véase también

Alteraciones accidentales
: ver el manual del usuario, `Alteraciones accidentales` y el manual del usuario, `Alteraciones accidentales automáticas`.

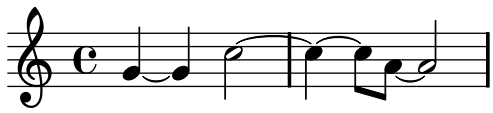
Armadura de la tonalidad
: ver el manual del usuario, `Armadura de la tonalidad`.

2.2.3 Ligaduras de unión y de expresión

Ligaduras de unión

Una `ligadura de unión` se crea adjuntando un carácter tilde '`~`' a la primera nota ligada

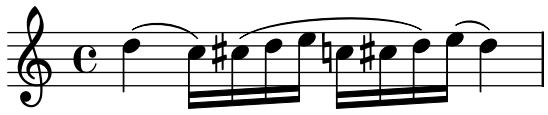
```
g4~ g c2~
c4 ~ c8 a8 ~ a2
```



Ligaduras de expresión

Una `ligadura de expresión` es una curva que se traza abarcando varias notas. Las notas inicial y final se marcan mediante '`(`' y '`)`' respectivamente.

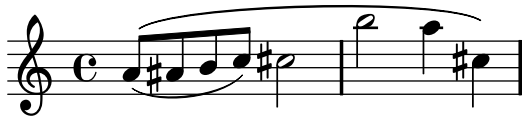
```
d4( c16) cis( d e c cis d) e( d4)
```



Ligaduras de fraseo

Las ligaduras que se utilizan para indicar fraseos más largos se pueden introducir mediante `\(` y `\)`. Puede haber al mismo tiempo ligaduras de legato y ligaduras de fraseo, pero no es posible tener legatos simultáneos o ligaduras de expresión simultáneas.

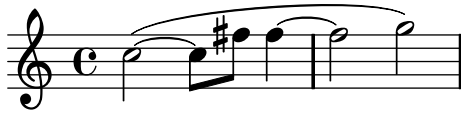
```
a8(\( ais b c) cis2 b'2 a4 cis,\)
```



Advertencias: ligaduras de expresión frente a ligaduras de unión

Una ligadura de expresión parece una `ligadura de unión`, pero tiene un significado distinto. Una ligadura (de unión) sencillamente hace que la primera nota sea más larga, y sólo se puede utilizar sobre parejas de notas iguales. Las ligaduras de expresión indican las articulaciones de las notas, y se pueden utilizar sobre grupos mayores de notas. Las ligaduras de unión y de expresión se pueden anidar unas dentro de otras.

```
c2~( c8 fis fis4 ~ fis2 g2)
```



Véase también

Ligaduras de unión
: ver el manual del usuario, `Ligaduras de unión`.

Ligaduras de expresión
: ver el manual del usuario, `Ligaduras de expresión`.

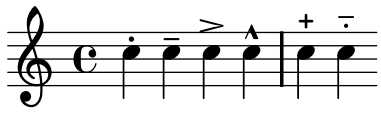
Ligaduras de fraseo
: ver el manual del usuario, `Ligaduras de fraseo`.

2.2.4 Articulaciones y matices dinámicos

Articulaciones

Las articulaciones (véase `articulación`) más corrientes se pueden añadir a las notas utilizando un guión '`-`' seguido de un carácter único:

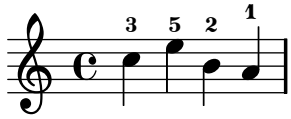
```
c-. c-- c-> c-^ c-+ c-_
```



Digitaciones

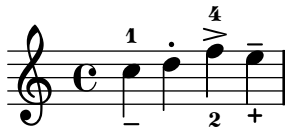
De manera similar, las indicaciones de digitación se pueden añadir a una nota utilizando un guión (‘-’) seguido del dígito deseado:

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



Las articulaciones y digitaciones normalmente se colocan de forma automática, pero puede especificar una dirección mediante ‘^’ (encima) o ‘_’ (debajo). También puede usar varias articulaciones sobre la misma nota. Sin embargo, casi siempre es mejor dejar que LilyPond determine la dirección de las articulaciones.

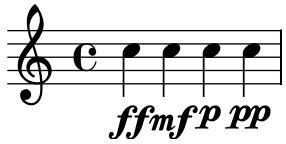
```
c_-^1 d^. f^4_2-> e^-_+
```



Matices dinámicos

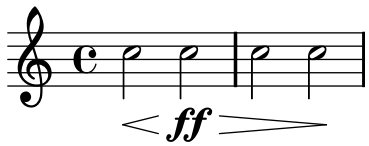
Las expresiones de matiz o signos dinámicos se hacen añadiendo las marcas (con una barra invertida) a la nota:

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



Los crescendi y decrescendi comienzan con las órdenes `\<` y `\>`. Una expresión terminal de matiz, como por ejemplo `\f`, dará por terminado el (de)crescendo, o bien se puede usar la orden `\!`:

```
c2\< c2\ff\> c2 c2\!
```



Véase también

Articulaciones
: ver el manual del usuario, `Articulaciones`.

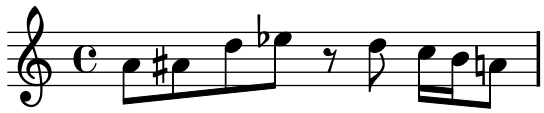
Digitaciones
: ver el manual del usuario, `Indicaciones de digitación`.

Matices
: ver el manual del usuario, `Matices dinámicos`.

2.2.5 Barras automáticas y manuales

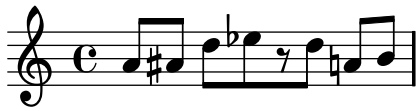
Todas las barras se dibujan automáticamente:

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



Si no le gustan las barras automáticas, pueden forzarse manualmente. Marque la primera nota que comprende la barra con '[' y la última con ']'.

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



Véase también

Barras automáticas
: ver el manual del usuario, Barras automáticas.

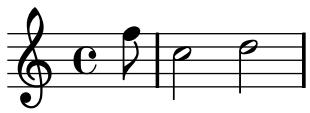
Barras manuales
: ver el manual del usuario, Barras manuales.

2.2.6 Comandos rítmicos avanzados

Compás parcial

Una anacrusa (o anacrusa) se introduce con la palabra clave \partial. Va seguida de una duración: \partial 4 es una anacrusa de negra y \partial 8 de corchea.

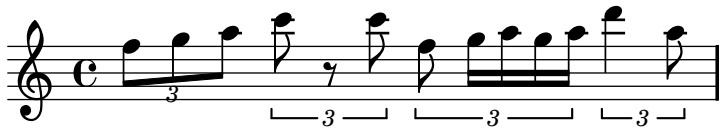
```
\partial 8
f8 c2 d
```



Grupos especiales

Los grupos especiales como los tresillos se hacen con la palabra clave \times. Requiere dos argumentos: una fracción y un fragmento de música. La duración del fragmento de música se multiplica por la fracción. Los tresillos hacen que las notas ocupen 2/3 de su duración expresa, por tanto un tresillo lleva una fracción de 2/3

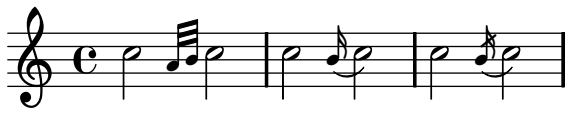
```
\times 2/3 { f8 g a }
\times 2/3 { c r c }
\times 2/3 { f,8 g16[ a g a] }
\times 2/3 { d4 a8 }
```



Notas de adorno

Las notas de adorno se crean con la instrucción `\grace`, aunque también se pueden conseguir precediendo una expresión musical por la palabra clave `\appoggiatura` o `\acciaccatura`

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



Véase también

Notas de adorno
: ver el manual del usuario, `Notas de adorno`,

Grupos especiales
: ver el manual del usuario, `Grupos especiales`,

Anacrusas ver el manual del usuario, `Partial measures`.

2.3 Varias notas a la vez

Esta sección es una introducción a las notas simultáneas: varios instrumentos, varios pentagramas para un solo instrumento (p.ej. piano) y acordes.

La palabra 'polifonía' en música hace referencia al hecho de tener más de una voz en un momento determinado dentro de una pieza musical. La palabra 'polifonía' en LilyPond se refiere al hecho de tener más de una voz en el mismo pentagrama.

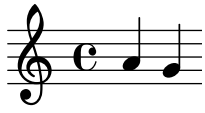
2.3.1 Explicación de las expresiones musicales

En los archivos de entrada de LilyPond, la música se representa mediante *expresiones musicales*. Una sola nota es una expresión musical, aunque no es una entrada válida por sí sola.

```
a4
```

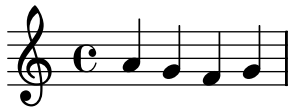
Al encerrar un grupo de notas dentro de llaves creamos una expresión musical:

```
{ a4 g4 }
```



Si colocamos un grupo de expresiones musicales (p.ej.: notas) dentro de llaves, eso significa que se encuentran en secuencia (es decir, cada una sigue a la anterior). El resultado es otra expresión musical:

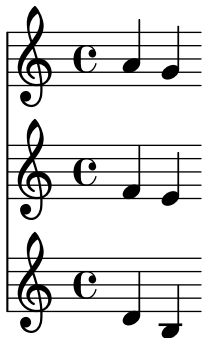
```
{ { a4 g } f g }
```



Expresiones musicales simultáneas: varios pentagramas

Esta técnica es muy útil para la música polifónica. Para introducir música con más voces o con más pentagramas, lo que hacemos es combinar varias expresiones en paralelo. Para indicar que dos voces se deben interpretar al mismo tiempo, sencillamente introduzca una combinación simultánea de expresiones musicales. Una expresión musical 'simultánea' se forma encerrando las expresiones dentro de `<<` y `>>`. En el ejemplo que sigue, tres secuencias (cada una de las cuales contiene dos notas diferentes) se combinan de forma simultánea:

```
\relative c'' {
  <<
    { a4 g }
    { f e }
    { d b }
  >>
}
```



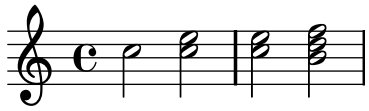
Tenga en cuenta que hemos sangrado cada nivel jerárquico de la entrada con un margen distinto. A LilyPond no le importa cuánto (o cuán poco) espacio haya al comienzo de una línea, pero el establecimiento de márgenes distintos dentro del código de LilyPond, de esta forma, lo hace mucho más fácil de leer por nosotros los seres humanos.

Advertencia: cada nota se entiende relativa a la nota anterior de la entrada, no relativa a la `c''` dentro de la instrucción inicial `\relative`.

Expresiones musicales simultáneas: un solo pentagrama

Para determinar el número de pentagramas en una pieza, LilyPond examina la primera expresión. Si ésta consiste en una sola nota, hay un solo pentagrama; si hay una expresión simultánea, hay más de un pentagrama.

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



Analogía: expresiones matemáticas

Este mecanismo es semejante a las fórmulas matemáticas: una fórmula grande se construye combinando fórmulas pequeñas. Dichas fórmulas se llaman expresiones, y su definición es recursiva de tal forma que se pueden construir expresiones de un tamaño y complejidad arbitrarios. Por ejemplo:

```
1

1 + 2

(1 + 2) * 3

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

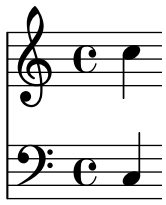
Ésta es una secuencia de expresiones donde cada expresión se encuentra contenida dentro de la siguiente, más grande. Las expresiones más simples son números, y las mayores se hacen combinando expresiones mediante operadores (como '`+`', '`*`' y '`/`') y paréntesis. Del mismo modo que las expresiones matemáticas, las expresiones musicales se pueden anidar a una profundidad arbitraria, lo que se hace necesario para músicas complejas como las partituras polifónicas.

2.3.2 Varios pentagramas

Como ya hemos visto en el manual del usuario, `Explicación de las expresiones musicales`, los archivos de entrada para LilyPond se construyen a base de expresiones musicales. Si la partitura comienza con expresiones musicales simultáneas, LilyPond crea varios pentagramas. Sin embargo es más fácil ver lo que ocurre si creamos cada uno de los pentagramas de forma explícita.

Para imprimir más de un pentagrama, cada fragmento de música que constituye un pentagrama se marca escribiendo `\new Staff` antes de él. Estos elementos `Staff` se combinan después en paralelo con `<<` y `>>`:

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



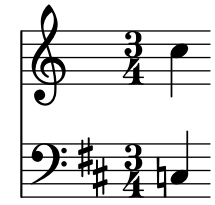
La instrucción `\new` inaugura un 'contexto de notación'. Un contexto de notación es un entorno dentro del que se interpretan los acontecimientos musicales (como las notas o las instrucciones `\clef`). Para piezas sencillas, tales contextos de notación se crean automáticamente. Para piezas más complicadas, es mejor marcar los contextos de forma explícita.

Existen varias clases de contextos. `Score`, `Staff` y `Voice` manejan la notación melódica, mientras que `Lyrics` se ocupa de los textos cantados y `ChordNames` imprime los nombres de los acordes.

En términos de sintaxis, la anteposición de `\new` a una expresión musical crea una expresión musical mayor. Es semejante al signo menos de las matemáticas. La fórmula $(4 + 5)$ es una expresión, por tanto $-(4 + 5)$ es una expresión más amplia.

Las indicaciones de compás escritas en un pentagrama afectan al resto de ellos[5]. En cambio, la armadura de la tonalidad de un pentagrama *no* afecta a los otros pentagramas.

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



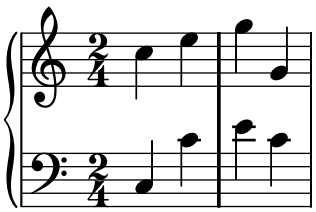
2.3.3 Sistemas de piano

La música para piano se compone tipográficamente en forma de dos pentagramas unidos mediante una llave. El aspecto impreso de este sistema de pentagramas se parece al ejemplo polifónico que aparece en el manual del usuario, `Varios pentagramas`, pero en esta ocasión la expresión completa se coloca dentro de un `PianoStaff`:

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

He aquí un pequeño ejemplo:

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



Véase también

Ver el manual del usuario, `Música de piano`.

[5] Este comportamiento se puede cambiar si uno lo desea; consulte el manual del usuario, `Notación polimétrica` para ver más detalles.

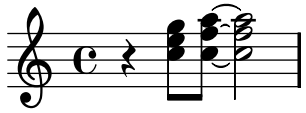
2.3.4 Combinar notas para formar acordes

Los acordes se hacen rodeando las notas con paréntesis en ángulo simples. Los paréntesis en ángulo son los símbolos '<' (menor que) y '>' (mayor que).

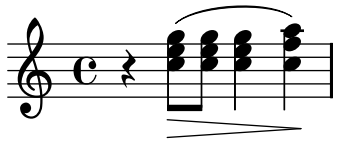
```
r4 <c e g>4 <c f a>2
```

Se pueden combinar las marcas como barras y ligaduras con acordes. Se deben colocar por fuera de los paréntesis en ángulo:

```
r4 <c e g>8[ <c f a>]~ <c f a>2
```



```
r4 <c e g>8\>( <c e g> <c e g>4 <c f a>\!)
```

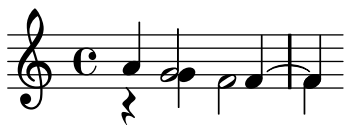


2.3.5 Polifonía en un solo pentagrama

Cuando distintas líneas melódicas se combinan sobre un solo pentagrama, se imprimen como voces polifónicas; cada voz lleva sus propias plicas, ligaduras y barras de corchea, y la voz superior tiene las plicas hacia arriba mientras que la voz inferior las tiene hacia abajo.

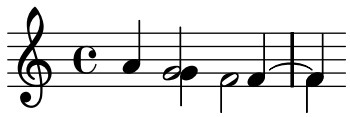
La introducción de estas partes se hace escribiendo cada voz en forma de secuencia (con `{...}`) y combinando éstas de forma simultánea, separando las voces con `\\`

```
<<
  { a4 g2 f4~ f4 } \\
  { r4 g4 f2 f4 }
>>
```



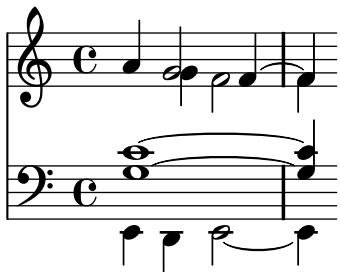
Para el tipografiado de música polifónica, puede ser conveniente la utilización de silencios separadores, o sea, silencios que no aparecen impresos. Son muy útiles para rellenar voces que temporalmente no están cantando. He aquí el mismo ejemplo con un silencio separador ('`s`') en vez de un silencio normal ('`r`'):

```
<<
  { a4 g2 f4~ f4 } \\
  { s4 g4 f2 f4 }
>>
```



Una vez más, las expresiones de este tipo se pueden anidar de forma arbitraria.

```
<<
  \new Staff <<
    { a4 g2 f4~ f4 } \\
    { s4 g4 f2 f4 }
  >>
  \new Staff <<
    \clef bass
    { <c g>1 ~ <c g>4 } \\
    { e,,4 d e2 ~ e4}
  >>
>>
```



Véase también

Ver el manual del usuario, `Polifonía básica`.

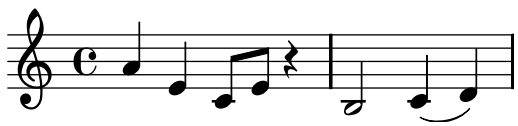
2.4 Canciones

Esta sección es una introducción a la música vocal y a las partituras de canciones sencillas.

2.4.1 Printing lyrics

Consideremos una melodía sencilla:

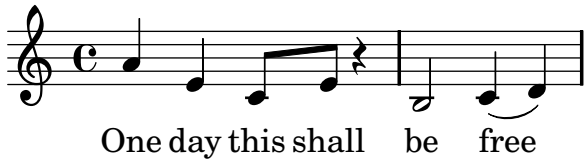
```
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
```



La letra se puede asignar a esas notas, combinando ambas con la palabra clave `\addlyrics`. La letra se escribe separando cada sílaba mediante un espacio.

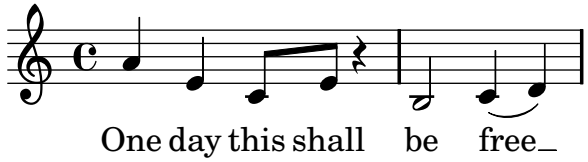
```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
```

```
  }
  \addlyrics { One day this shall be free }
>>
```



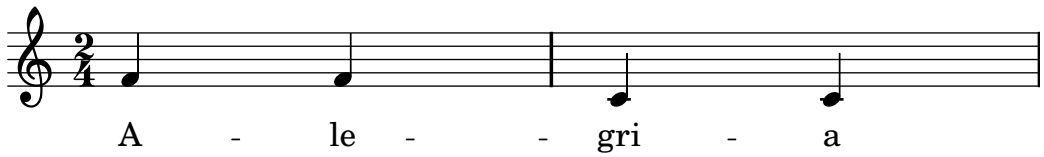
Esta melodía acaba en un `melisma`, una sílaba única ('suelta') que se canta sobre más de una nota. Esto se indica mediante una *línea de extensión*. Se escribe como dos guiones bajos `__`:

```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



De forma parecida, los guiones entre las palabras se pueden escribir como dos guiones, dando como resultado unas líneas cortas entre cada dos sílabas:

```
<<
  \relative c' {
    \time 2/4
    f4 f c c
  }
  \addlyrics { A -- le -- gri -- a }
>>
```



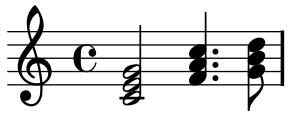
Véase también

Otras posibilidades como poner varios versos debajo de una melodía, se discuten en el manual del usuario, `Música vocal`.

2.4.2 Hojas guía de acordes

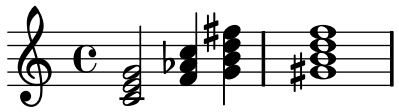
En la música pop es muy común denotar el acompañamiento con los nombres de los acordes. Estos acordes se pueden escribir como notas:

```
\chordmode { c2 f4. g8 }
```



Ahora, cada altura se lee como la fundamental de un acorde en vez de como una nota. Este modo se activa con `\chordmode`. Otros acordes pueden construirse añadiendo modificadores después de un signo de dos puntos. El ejemplo que sigue muestra algunos modificadores comunes:

```
\chordmode { c2 f4:m g4:maj7 gis1:dim7 }
```

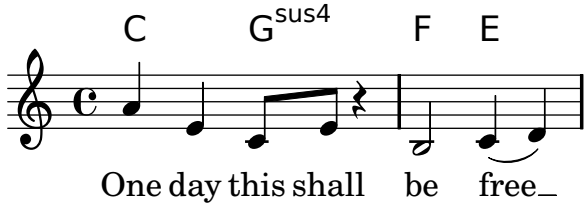


Para canciones con acordes, éstos no se imprimen sobre pentagramas, sino como nombres en su propia línea. Esto se consigue utilizando `\chords` en sustitución de `\chordmode`. Utiliza la misma sintaxis que `\chordmode`, pero pinta las notas dentro de un contexto de `ChordNames`, con el siguiente resultado:

```
\chords { c2 f4.:m g4.:maj7 gis8:dim7 }
```

Cuando se combinan, los nombres de acorde, la letra y la melodía forman una hoja de canción con acordes (‘lead sheet’):

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Véase también

Puede ver una lista completa de los modificadores y otras opciones de presentación en el manual del usuario, `Acordes`.

2.5 Retoques finales

Éste es el último apartado del tutorial; muestra la forma de dar los toques finales a piezas sencillas, y ofrece una introducción al resto del manual.

2.5.1 Número de la versión

La indicación `\version` deja registrado para qué versión de LilyPond se escribió el archivo:

```
\version "2.11.38"
```

por convenio se sitúa al principio del archivo de partitura de LilyPond.

Estas anotaciones hacen menos problemáticas las subsiguientes actualizaciones de LilyPond. Los cambios en la sintaxis se tratan mediante un programa especial, '`convert-ly`' (ver el manual de utilización del programa, `Actualizar ficheros con convert-ly`), y utiliza `\version` para determinar qué reglas hay que aplicar.

2.5.2 Añadir títulos

La información sobre el título, autor, número de Opus y similares se escriben en el bloque `\header`. Éste se encuentra fuera de la expresión musical principal; el bloque `\header` normalmente se sitúa por debajo del el manual del usuario, `Número de la versión`.

```
\version "2.11.38"
\header {
  title = "Symphony"
  composer = "Me"
  opus = "Op. 9"
}

{
  ... music ...
}
```

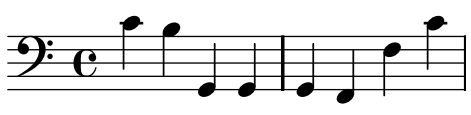
Cuando se procesa el archivo, el título y el autor se imprimen por encima de la música. Puede obtener más información sobre los títulos en el manual del usuario, `Crear títulos`.

2.5.3 Nombres de nota absolutos

Hasta el momento siempre hemos utilizado `\relative` para definir las alturas. Ésta es la forma más sencilla de escribir la mayor parte de la música, pero existe otra forma de definir las alturas: el modo absoluto.

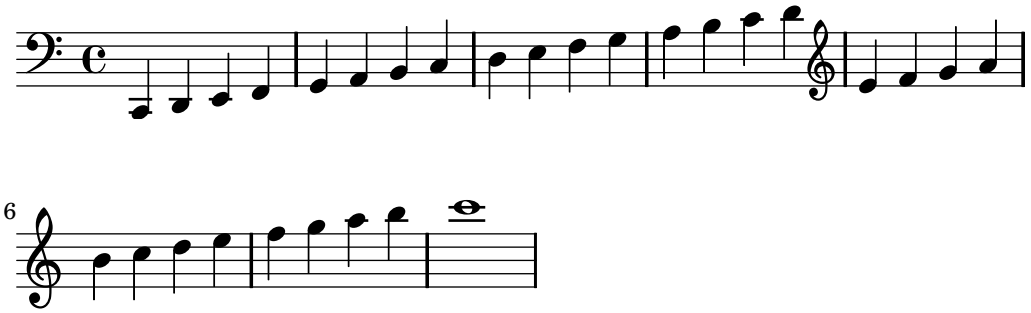
Si omite el `\relative`, LilyPond tratará todas las alturas como valores absolutos. Una `c'` significará siembre un Do central, una `b` significará siempre la nota inmediatamente por debajo del Do central, y una `g,` significará siempre la nota que se coloca en la primera línea del pentagrama en clave de Fa.

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



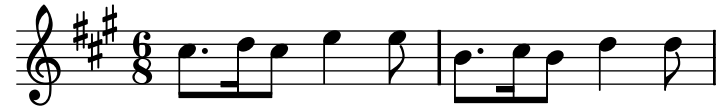
He aquí una escala que abarca cuatro octavas:

```
{
  \clef bass
  c, d, e, f,
  g, a, b, c
  d e f g
  a b c' d'
  \clef treble
  e' f' g' a'
  b' c'' d'' e''
  f'' g'' a'' b''
  c'''1
}
```



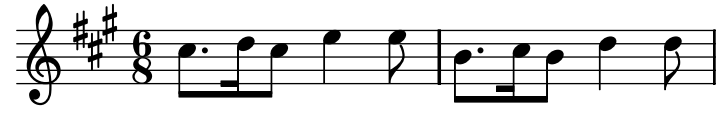
Como puede ver, escribir una melodía en clave de Sol implica escribir gran cantidad de apóstrofes ' . Consideremos este fragmento de Mozart:

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



Todos estos apóstrofes hacen casi ilegible el código de entrada y será origen de numerosos errores. Con `\relative`, el ejemplo anterior es mucho más fácil de leer:

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



Si comete un error con una marca de octava (' o ,) mientras trabaja en el modo `\relative`, será muy obvio (muchas notas estarán en la octava equivocada). Mientras trabaja en el modo absoluto, un solo fallo no será tan visible, y tampoco será tan fácil de localizar.

Sin embargo, el modo absoluto es útil para escribir música que contenga intervalos grandes, y será extremadamente útil para hacer archivos de LilyPond generados por ordenador.

2.5.4 Organizing pieces with identifiers

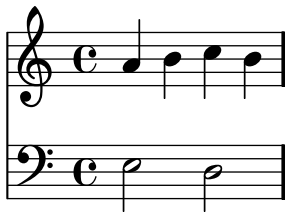
Cuando los elementos que hemos discutido anteriormente se combinan para producir archivos mayores, las expresiones musicales se hacen enormes. En música polifónica con muchos pentagramas, los archivos de entrada pueden volverse muy propensos a la confusión. Podemos reducir esta confusión utilizando los *identificadores*.

Con los identificadores (también conocidos como variables o macros), podemos trocear las expresiones musicales complejas. Un identificador se asigna de la manera siguiente:

```
musicaConNombre = { ... }
```

El contenido de la expresión musical `musicaConNombre` se puede usar posteriormente colocando una barra invertida delante del nombre (`\musicaConNombre`, igual que una orden normal de LilyPond). Los identificadores se deben definir *antes* de la expresión musical principal.

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



El nombre de un identificador debe consistir enteramente en caracteres alfabéticos, es decir sin números, guiones ni guiones bajos.

Es posible utilizar variables para otras muchas clases de objetos en el código de entrada. Por ejemplo:

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

Dependiendo de su contenido, el identificador se puede usar en distintos lugares. El siguiente ejemplo utiliza las variables anteriores:

```
\paper {
  \aFivePaper
```

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

2.5.5 Más allá del tutorial

Después de terminar el tutorial, quizá debería probar a escribir una o dos piezas. Comience con una de las el manual del usuario, `Plantillas` y añada algunas notas. Si necesita un tipo de notación que no ha sido tratada en el tutorial, eche un vistazo a la Referencia de Notación, empezando por el manual del usuario, `Basic notation`. Si quiere escribir música para un conjunto instrumental que no está cubierto por ninguna plantilla, consulte el manual del usuario, `Extender las plantillas`.

Una vez que ha escrito algunas piezas cortas, lea el resto del Manual de aprendizaje (capítulos 3 al 5). ¡Por supuesto, no pasa nada por leerlo ahora mismo! Sin embargo, el resto del Manual de Aprendizaje da por sentado que está familiarizado con la entrada de LilyPond. Puede saltarse estos capítulos ahora y volver a ellos cuando haya adquirido más experiencia.

2.5.6 Cómo leer el manual

Como ya vimos en el manual del usuario, `Cómo leer el tutorial`, muchos ejemplos del tutorial omitían el `\relative c'' { ... }` alrededor del ejemplo mostrado.

Durante el resto del manual seremos mucho menos estrictos respecto de los ejemplos impresos: a veces pueden omitir el `\relative c'' { ... }`, pero otras veces se puede usar una nota inicial diferente (como `c'` o `c,,`), y en ocasiones ¡todo el ejemplo estará en el modo de notas absoluto! Sin embargo, las ambigüedades de esta clase existen solamente allí donde las alturas no son lo más importante. En aquellos ejemplos en los que las alturas tienen importancia, hemos escrito explícitamente `\relative` o bien el `{ }` en modo absoluto.

Si aún tiene dudas respecto de la entrada de LilyPond exacta que se ha utilizado en un ejemplo, lea la versión HTML (si no lo está haciendo ya) y pulse con el ratón sobre la imagen de la música. De esta forma se mostrará la entrada exacta que LilyPond usó para generar este manual.

Para informarse sobre la estructura que tiene el resto del manual, consulte el manual del usuario, `Sobre el presente manual`.

3 Juntándolo todo

Este capítulo trata de los conceptos generales de LilyPond y de cómo elaborar bloques `\score`.

3.1 Extender las plantillas

Ha leído el tutorial y ahora sabe escribir música. Pero ¿cómo puede poner los pentagramas que quiere? Las plantillas están muy bien, pero ¿qué ocurre si quiere algo que no está en una de ellas?

Para empezar, tome la plantilla que le parezca más parecida a aquello que quiere conseguir. Digamos que quiere escribir algo para soprano y cello. En este caso comenzaríamos con la plantilla 'Notas y letra' (para la parte de soprano).

```
\version "2.11.38"
melodia = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

texto = \lyricmode {
  Aaa Bee Cee Dee
}

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

Ahora queremos añadir una parte de violoncello. Veamos el ejemplo 'Sólo notas':

```
\version "2.11.38"
melodia = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

\score {
\new Staff \melody
\layout { }
\midi { }
}
```

No necesitamos dos comandos `\version`. Vamos a necesitar la sección `melodia`. No queremos dos secciones `\score` (si tuviésemos dos `\score`s, acabaríamos con las dos particellas por separado. Queremos las dos juntas, como un dúo. Dentro de la sección `\score`, no nos hacen falta dos `\layout` ni dos `\midi`.

Si nos limitásemos a copiar y pegar la sección `melodia`, acabaríamos con dos secciones `melodia` separadas, así que vamos a cambiarles el nombre. Llamaremos `musicaSoprano` a la sección de la soprano y `musicaCello` a la sección del violoncello. Al mismo tiempo cambiaremos el nombre de `texto` a `letraSoprano`. Recuerde cambiar el nombre a las dos apariciones de todos estos nombres – tanto la definición inicial (la parte `melodia = relative c' {`) – como el uso de ese nombre (en la sección `\score`).

También aprovecharemos para cambiar el pentagrama de la parte del cello (los violoncellos se escriben normalmente en clave de Fa). Asimismo, cambiaremos algunas notas del cello.

```
\version "2.11.38"
musicaSoprano = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

letraSoprano = \lyricmode {
  Aaa Bee Cee Dee
}

musicaCello = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    \new Voice = "uno" {
      \autoBeamOff
      \sopranoMusic
    }
    \new Lyrics \lyricsto "uno" \letraSoprano
  >>
  \layout { }
  \midi { }
}
```

Esto tiene una pinta prometedora, pero la parte del cello no sale en la partitura (no la hemos puesto en la sección `\score`). Si queremos que la parte del cello aparezca debajo de la de soprano, tenemos que añadir

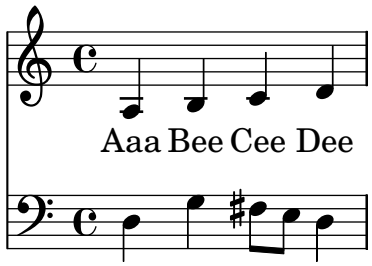
```
\new Staff \musicaCello
```

justo debajo de todo lo de la soprano. También tenemos que poner `<<` y `>>` antes y después de la música – lo que indica a LilyPond que hay más de una cosa (en este caso, `Staff`) sucediendo al mismo tiempo – . La `\score` se parecerá ahora a esto

```
\score{
  <<
    <<
      \new Voice = "uno" {
        \autoBeamOff
        \sopranoMusic
      }
      \new Lyrics \lyricsto "uno" \letraSoprano
    >>
    \new Staff \musicaCello
  >>
  \layout { }
  \midi { }
}
```

Esto parece un poco enrevesado; los márgenes están descuadrados. Esto tiene fácil solución. Presentamos aquí la plantilla completa para soprano y cello.

```
\version "2.11.38"
sopranoMusic = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

sopranoLyrics = \lyricmode {
  Aaa Bee Cee Dee
}

celloMusic = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \sopranoMusic
      }
      \new Lyrics \lyricsto "one" \sopranoLyrics
    >>
    \new Staff \celloMusic
  >>
  \layout { }
  \midi { }
}
```



3.2 Cómo funcionan los archivos de LilyPond

El formato de entrada de LilyPond es bastante libre en su forma y concede a los usuarios con experiencia mucha flexibilidad para estructurar sus archivos de la forma que deseen. Sin embargo, toda esta flexibilidad puede hacer que las cosas se vuelvan confusas para los nuevos usuarios. Esta sección le va a explicar parte de esta estructura, pero puede obviar ciertos detalles en aras de la simplicidad. Para ver una descripción completa del formato de entrada, consulte el manual del usuario, `Estructura del archivo`.

Casi todos los ejemplos del presente manual con pequeños fragmentos de código. Por ejemplo

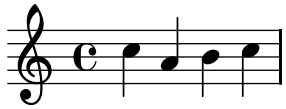
```
c4 a b c
```

Como (esperamos) ya se habrá dado cuenta, esto no se puede compilar tal cual está. Estos ejemplos son sólo resúmenes de los ejemplos completos. Todos ellos necesitan, cuando menos, un par de llaves antes y después para que se puedan compilar

```
{
  c4 a b c
}
```

Casi todos los ejemplos también usan el comando `\relative c'` (o `c''`). Esto no es necesario para conseguir que los ejemplos simplemente se puedan compilar, pero casi siempre la salida tendrá un aspecto muy extraño si omite el `\relative c'`.

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



Ahora nos encontramos con el único tropiezo de verdad: la entrada de LilyPond en esta forma es realmente *otra* abreviatura. Aunque se puede compilar y presenta la salida correcta, es una abreviatura de

```
\score {
  \relative c'' {
    c4 a b c
  }
}
```

Un `\score` debe comenzar con una única expresión musical. Recuerde que una expresión musical podía ser cualquier cosa desde una sola nota hasta todo un

```
{
  \new GrandStaff <<
    introduzca aquí la partitura completa de una ópera de Wagner
  >>
}
```

Ya que todo está dentro de `{ ... }`, cuenta como una sola expresión musical.

La `\score` puede contener otras cosas como

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

Hay personas que ponen algunos de estos comandos fuera del bloque `\score` (por ejemplo, `\header` se sitúa con frecuencia por encima del `\score`. Es tan sólo otra abreviatura que LilyPond acepta.

Otro atajo genial es la posibilidad de definir variables. Todas las plantillas emplean lo siguiente:

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

\score {
  { \melodia }
}
```

Cuando LilyPond examina este archivo, toma el valor de `melodia` (todo lo que está después del signo igual) y lo inserta dondequiera que ve `\melodia`. No se requiere un cuidado especial con los nombres – puede ser `melodia`, `global`, `manoderechadelpiano`, o `fulanomengano` –. Puede usar el nombre de variable que desee. Para ver más detalles, consulte el manual del usuario, `Saving typing with identifiers and functions`.

Para ver una definición completa del formato de entrada, consulte el manual del usuario, `Estructura del archivo`.

3.3 Score is a single musical expression

En la sección anterior, el manual del usuario, `Cómo funcionan los archivos de LilyPond`, pudimos ver la organización general de los archivos de entrada de LilyPond. Pero parece que nos saltamos la parte más importante: ¿cómo averiguamos qué escribir después de `\score`?

No nos hemos saltado nada en absoluto. El gran misterio es, sencillamente, que no hay *ningún* misterio. La siguiente línea lo explica todo:

Una `\score` *debe comenzar con una única expresión musical.*

Quizá encuentre útil dar un repaso a el manual del usuario, `Explicación de las expresiones musicales`. En esta sección, vimos cómo elaborar grandes expresiones musicales a partir de pequeñas piezas – comenzábamos con notas, luego acordes, etc. –. Ahora partiremos de una gran expresión musical y recorreremos el camino inverso hacia abajo.

```
\score {
  {   % esta llave da inicio a la expresión musical completa
    \new GrandStaff <<
      introduzca aquí la partitura completa de una ópera de Wagner
    >>
  }   % esta llave da por terminada la expresión musical completa
  \layout { }
}
```

Una ópera de Wagner completa puede ser fácilmente el doble de larga que este manual, por tanto vamos a hacer sólo un cantante y un piano. No necesitamos un `GrandStaff` para este conjunto, así que lo retiramos. Sin embargo, sí que *necesitamos* un cantante y un piano.

```
\score {
  {
    <<
      \new Staff = "cantante" <<
      >>
      \new PianoStaff = "piano" <<
      >>
    >>
  }
  \layout { }
}
```

Recuerde que usamos << y >> para presentar música simultánea. Y desde luego ¡queremos presentar las partes vocal y del piano al mismo tiempo!

```
\score {
  {
    <<
      \new Staff = "cantante" <<
        \new Voice = "vocal" { }
      >>
      \new Lyrics \lyricsto vocal \new Lyrics { }
      \new PianoStaff = "piano" <<
        \new Staff = "superior" { }
        \new Staff = "inferior" { }
      >>
    >>
  }
  \layout { }
}
```

Ahora tenemos muchos más detalles. Tenemos la pauta del cantante: contiene una `Voice` o voz (en LilyPond, este término hace referencia a un conjunto de notas, no necesariamente notas vocales – por ejemplo, un violín generalmente toca una voz –) y el texto de la canción. También tenemos una pauta de piano: contiene un pentagrama superior (mano derecha) y un pentagrama inferior (mano izquierda).

En este momento podríamos comenzar a meter las notas. Dentro de las llaves que siguen a `\new Voice = vocal`, podríamos empezar escribiendo

```
\relative c'' {
  a4 b c d
}
```

Pero si lo hiciéramos, la sección `\score` se haría bastante larga y sería más difícil comprender lo que ocurre. En lugar de esto utilizaremos identificadores o variables.

```
melodia = { }
texto = { }
superior = { }
inferior = { }
\score {
  {
    <<
      \new Staff = "cantante" <<
        \new Voice = "vocal" { \melodia }
      >>
```

```
        \new Lyrics \lyricsto vocal \new Lyrics { \texto }
        \new PianoStaff = "piano" <<
          \new Staff = "superior" { \superior }
          \new Staff = "inferior" { \inferior }
        >>
      >>
    }
    \layout { }
  }
```

Recuerde que puede usar casi cualquier nombre que se le antoje. Las limitaciones impuestas a los nombres de identificador se detallan en el manual del usuario, `Estructura del archivo`.

Cuando escriba una sección `\score` o cuando la esté leyendo, hágalo despacio y con cuidado. Comience por la capa exterior y luego trabaje sobre cada una de las capas interiores. Esto también sirve para ser estricto con los márgenes – ¡asegúrese de que en su editor de texto cada elemento de la misma capa comienza en la misma posición horizontal! –.

3.4 Una particella orquestal

En música orquestal, todas las notas se imprimen dos veces. Una vez en las particellas para los músicos, y otra para la partitura del director. Los identificadores se pueden usar para evitar la duplicación del trabajo. La música se escribe una vez y se almacena en una variable. El contenido de dicha variable se usa después para generar tanto la particella como la partitura del director.

Es muy conveniente definir las notas en un archivo especial. Por ejemplo, supongamos que el archivo '`trompa.ly`' contiene la siguiente parte de un dúo para trompa y fagot:

```
notasTrompa = \relative c {
  \time 2/4
  r4 f8 a cis4 f e d
}
```

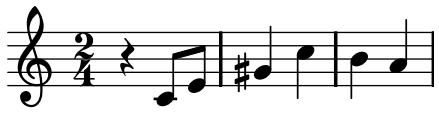
Luego se hace una particella escribiendo en un archivo lo siguiente

```
\include "trompa.ly"
\header {
  instrument = "Trompa en Fa"
}

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

La línea

```
\include "trompa.ly"
```

sustituye el contenido de '`trompa.ly`' en esta posición dentro del archivo, así que `notasTrompa` se define con posterioridad. El comando `\transpose f c'` indica que el argumento constituido por `\notasTrompa` se debe transponer una quinta hacia arriba. Lo que suena como '`f`' se escribe como `c'`, lo que corresponde con el tono de afinación de una trompa normal en Fa. La transposición se puede ver en la siguiente salida



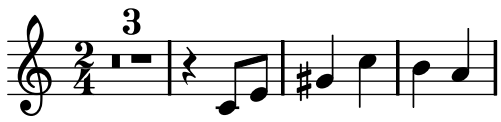
En piezas para conjunto, con frecuencia una de las voces no suena durante muchos compases. Esto queda denotado por un silencio especial, el silencio multicompás. Se introduce con una 'R' mayúscula seguida de una duración (1 en el caso de la redonda, 2 en el caso de una blanca, etc.). Multiplicando la duración se pueden construir silencios más largos. Por ejemplo, este silencio ocupa 3 compases de 2/4

```
R2*3
```

Cuando se imprime la particella tienen que comprimirse los silencios multicompás. Esto se hace estableciendo una variable en tiempo de ejecución

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

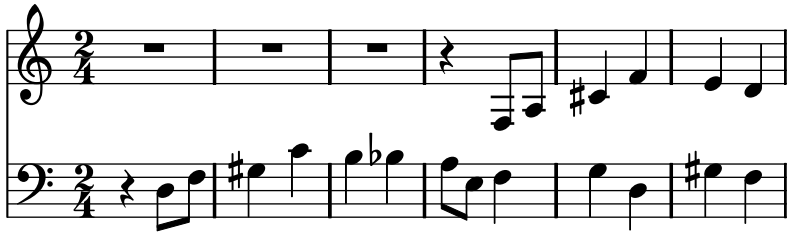
Este comando establece el valor de la propiedad `skipBars` en el contexto de `Score` a verdadero (`##t`). Anteponiendo el silencio y esta opción a la música anterior, llegamos al siguiente resultado



Esta partitura se hace combinando toda la música junta. Suponiendo que la otra voz se encuentra dentro de `notasFagot` en el archivo '`fagot.ly`', la partitura se hace con

```
\include "fagot.ly"
\include "trompa.ly"

<<
  \new Staff \notasTrompa
  \new Staff \notasFagot
>>
```

lo que nos lleva a



Se puede encontrar una información más profunda sobre cómo preparar particellas y partituras en el manual de notación; consulte el manual del usuario, `Orchestral music`.

El establecimiento de variables en tiempo de ejecución ('propiedades') se trata en el manual del usuario, `Cambiar las propiedades de un contexto al vuelo`.

4 Trabajar en proyectos de LilyPond

Esta sección explica cómo resolver o evitar ciertos problemas comunes. Si tiene experiencia en programación muchos de estos consejos pueden parecer obvios, pero aún así le recomendamos que lea este capítulo.

4.1 Sugerencias para escribir archivos de LilyPond

En este momento está preparado para comenzar a escribir archivos de LilyPond más grandes – no sólo los pequeños ejemplos que aparecen en el tutorial, sino piezas completas –. Pero ¿cómo debe proceder para hacerlo?

En la medida en que LilyPond entienda sus archivos y produzca la salida que usted pretendía, realmente no importa mucho qué aspecto tengan sus archivos. Sin embargo existen algunas otras cosas a tener en cuenta cuando se escriben archivos de LilyPond.

- ¿Qué ocurre si comete un fallo? La estructura de un archivo lilypond puede hacer que ciertos errores se hagan más fáciles (o más difíciles) de encontrar.
- ¿Qué ocurre si quiere compartir sus archivos con otras personas? De hecho, ¿y si quiere alterar sus propios archivos después de algunos años? Algunos archivos de lilypond se comprenden a primera vista; otros pueden tenerle rascándose la cabeza durante una hora.
- ¿Qué ocurre si quiere actualizar su archivo de lilypond para poderlo usar con una versión más reciente del programa? La sintaxis de la entrada se modifica de forma ocasional según lilypond se va perfeccionando. Casi todos los cambios se pueden hacer de forma automática con `convert-ly`, pero algunos podrían necesitar de una ayuda manual. Los archivos de LilyPond se pueden estructurar para que sean más fáciles (o más difíciles) de actualizar.

4.1.1 Sugerencias de tipo general

Presentamos algunas sugerencias que le pueden servir de ayuda para evitar o corregir problemas:

- **Incluya los números de `\version` en todos los archivos**. Dese cuenta de que todas las plantillas contienen una cadena como `\version "2.11.38"`. Le recomendamos mucho que siempre incluya la `\version`, sin importar cuán pequeño pueda ser su archivo. Desde la experiencia personal podemos decirle que es bastante frustrante intentar recordar el número de versión de LilyPond que estaba usando hace unos años. `convert-ly` requiere que declare qué versión de LilyPond utilizó.
- **Incluya comprobaciones**: el manual del usuario, `Comprobación del compás`, el manual del usuario, `Comprobación de la octava` y el manual del usuario, `Barnumber check`. Si incluye comprobaciones de vez en cuando, en caso de que cometa un error podrá localizarlo mucho más rápidamente. ¿Con qué frecuencia es 'de vez en cuando'? Depende de la complejidad de la música. Para una música muy sencilla, quizá tan sólo una o dos veces. Para una música muy compleja, quizá a cada compás.
- **Un compás por cada línea de texto**. Si hay algo muy complicado, ya sea en la propia música o en la salida que desea producir, a menudo conviene escribir un solo compás por cada línea. El ahorro en espacio de pantalla que se obtiene al amontonar ocho compases por línea no merece la pena si luego tiene que 'depurar' los archivos.
- **Comente los archivos**. Utilice o números de compás (de vez en cuando) o referencias a temas musicales ('segundo tema de los violines,' 'cuarta variación,' etc.). Puede que no necesite comentarios cuando introduce una pieza por vez primera, pero si quiere volver a ella o modificar algo al cabo de dos o tres años, y también si le pasa la fuente a un amigo, será todo un desafío determinar sus intenciones o de qué manera estaba estructurado el archivo si no le añadió los comentarios.
- **Aplique márgenes a las llaves**. Muchos problemas están causados por una falta de equilibrio en el número de `{` y `}`.

- **Escriba las duraciones explícitamente** al comienzo de las secciones e identificadores. Si especifica `c4 d e` al principio de una frase (en lugar de sólo `c d e`) se puede ahorrar problemas si reelabora la música más tarde.
- **Separe los trucos** de las definiciones musicales. Consulte el manual del usuario, `Saving typing with identifiers and functions` y el manual del usuario, `Hojas de estilo`.

4.1.2 Tipografiar música existente

Si está introduciendo música a partir de una partitura existente (es decir, tipografiando una hoja de música ya impresa),

- Introduzca un sistema del manuscrito (la copia física) cada vez (pero mantenga la práctica de escribir un compás por línea de texto), y compruebe cada sistema cuando lo haya terminado. Puede usar el comando `showLastLength` para acelerar el proceso – ver el manual del usuario, `Saltar la música corregida` – .
- Defina `mBreak = { \break }` e inserte `\mBreak` dentro del archivo de entrada donde el manuscrito tenga un saldo de línea. De esta forma le resultará mucho más fácil comparar la música de LilyPond con la original. Cuando haya terminado de revisar su partitura podrá definir `mBreak = { }` para quitar todos esos saltos de línea. Así permitirá a LilyPond colocar los saltos donde éste lo estime más oportuno.

4.1.3 Proyectos grandes

Al trabajar en proyecto grande se hace esencial tener una estructura clara en los archivos de lilypond.

- **Utilice un identificador para cada voz**, con un mínimo de estructura dentro de la definición. La estructura de la sección `\score` es la que cambiará con mayor probabilidad; por contra, es extremadamente improbable que cambie la definición de `violin` en versiones nuevas de LilyPond.

```
violin = \relative c'' {
g4 c'8. e16
}
...
\score {
  \new GrandStaff {
    \new Staff {
      \violin
    }
  }
}
```

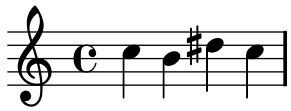
- **Separe los trucos de las definiciones musicales**. Ya se mencionó en el manual del usuario, `Sugerencias de tipo general`, pero para proyectos grandes es vital. Quizá tengamos que cambiar la definición de `fthenp`, pero en ese caso sólo lo tendremos que hacer una vez, y aún podremos evitar tocar nada dentro de `violin`.

```
fthenp = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
g4\fthenp c'8. e16
}
```

4.2 Saving typing with identifiers and functions

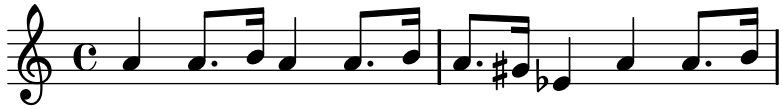
Llegado a este punto, usted ha visto cosas de este tipo:

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



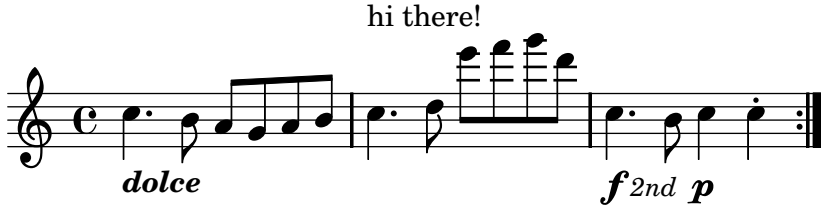
Incluso se dará cuenta de que esto puede ser útil en música minimalista:

```
fragA = \relative c'' { a4 a8. b16 }
fragB = \relative c'' { a8. gis16 ees4 }
violin = \new Staff { \fragA \fragA \fragB \fragA }
\score {
  {
    \violin
  }
}
```



Sin embargo también puede usar estos identificadores (que también se conocen como variables, macros o comandos definidos por el usuario) para hacer trucos:

```
dolce = \markup{ \italic \bold dolce }
padText = { \once \override TextScript #'padding = #5.0 }
fthenp=_\markup{ \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}
\score {
  {
    \violin
  }
\layout{ragged-right=##t}
}
```



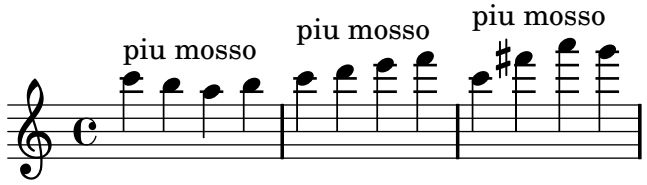
Obviamente estos identificadores son útiles para ahorrar tecleo. Pero son dignos de tener en cuenta incluso si se van a utilizar una sola vez: reducen la complejidad. Examinemos el ejemplo anterior reescrito sin ningún identificador. Encontrará que es mucho más difícil de leer, sobre todo la última línea.

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

Hasta ahora hemos contemplado la sustitución estática: cuando LilyPond se encuentra con `\padText`, lo sustituye con aquello que hemos definido que sea (es decir, todo lo que está a la derecha de `padtext=`).

LilyPond también puede manejar sustituciones no estáticas (piense en ellas como en funciones).

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

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



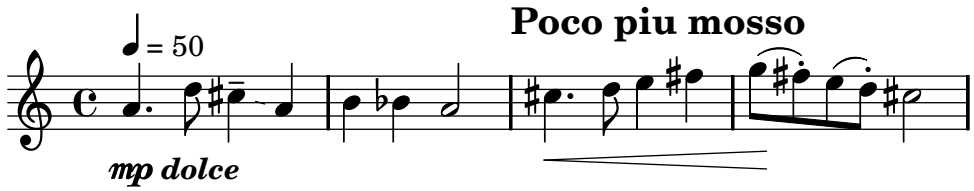
La utilización de identificadores también es una buena forma de reducir el trabajo si la sintaxis de entrada de LilyPond cambia (ver el manual del usuario, `Actualizar archivos antiguos`). Si tiene una sola definición (como p.ej. `\dolce`) para todos sus archivos (ver el manual del usuario, `Hojas de estilo`), y después la sintaxis se modifica, sólo tendrá que actualizar su definición `\dolce` única, en lugar de tener que hacer cambios en cada uno de los archivos `.ly`.

4.3 Hojas de estilo

La salida que produce LilyPond se puede modificar profundamente; consulte el manual del usuario, `Trucar la salida` para leer detalles sobre este asunto. Pero ¿qué ocurre si tiene muchos archivos a los que les quiere aplicar sus propios trucos? O ¿qué ocurre si, sencillamente, quiere separar los trucos de la propia música? Todo esto es bastante fácil de conseguir.

Veamos un ejemplo. No se preocupe si no entiende las partes que tienen todos los `#()`. Esto se explicará en el manual del usuario, `Trucos avanzados con Scheme`.

```
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```

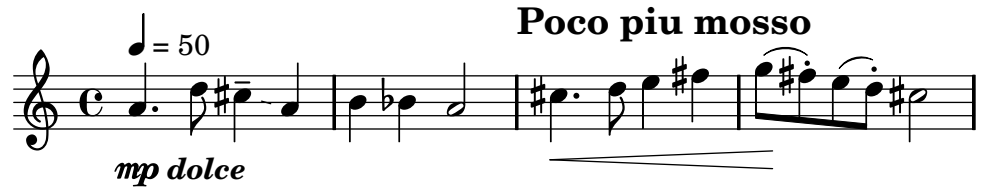


Existen varios problemas con la salida que se superpone; los arreglaremos utilizando las técnicas descritas en el manual del usuario, `Mover objetos`. Pero también haremos algo respecto a las definiciones `mpdolce` y `tempoMark`. Éstas producen la salida que deseamos, pero quizá las querríamos utilizar en otra pieza. Podríamos simplemente copiarlas y pegarlas al principio de cada archivo, pero sería bastante molesto. También hace que se queden las definiciones a la vista dentro de nuestros archivos de música, y yo personalmente encuentro todos los `#()` bastante poco estéticos. Los vamos a esconder dentro de otro archivo:

```
%%% guardar esto en un archivo de nombre "definiciones.ly"
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
```

Ahora modificaremos la música (guardemos este archivo como '`"musica.ly"`').

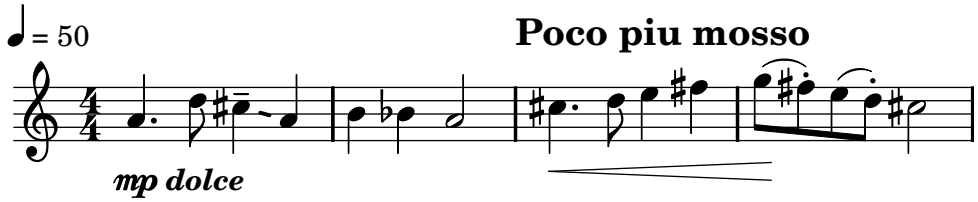
```
\include "definiciones.ly"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \once \override Score.RehearsalMark #'padding = #2.0
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```



Eso tiene mejor aspecto, pero haremos algunos cambios más. El glissando es difícil de ver, así que lo haremos más grueso y lo acercaremos a las cabezas de las notas. Pondremos la indicación metronómica encima de la clave, en lugar de ir encima de la primera nota. Y por último, mi profesor de composición odia las indicaciones de compás 'C', así que la convertiremos en '4/4'.

Sin embargo, no debe cambiar el archivo '`musica.ly`'. Sustituya nuestro archivo '`definiciones.ly`' con éste:

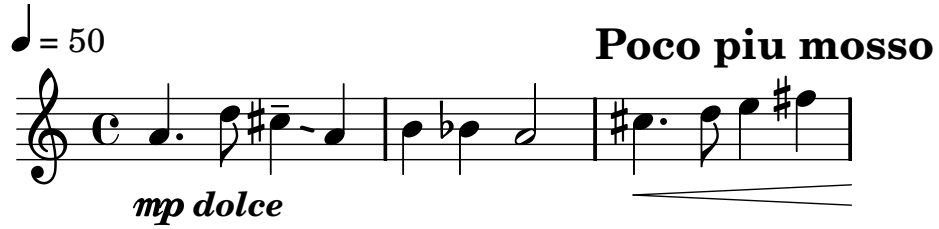
```
%%%  definiciones.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```

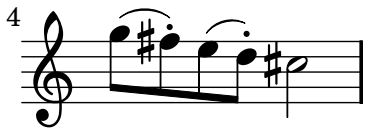


¡Eso tiene un aspecto mucho mejor! Ahora suponga que quiere publicar esta pieza. A mi profesor de composición no le gustan las indicaciones de compás 'C', pero yo les tengo cierto cariño. Copiaremos el archivo actual '`definiciones.ly`' a '`publicar-web.ly`' y modificaremos éste. Como el propósito de esta música es producir un PDF que va a mostrarse en la pantalla, también vamos a aumentar el tamaño general de la salida.

```
%%%  definiciones.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
```

```
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

#(set-global-staff-size 23)
\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```





Ahora, en la música, simplemente sustituyo `\include "definiciones.ly"` por `\include "publicar-web.ly"`. Por supuesto, podríamos hacer esto aún más práctico. Podríamos hacer un archivo '`definiciones.ly`' que contuviera solamente las definiciones de `mpdolce` y `tempoMark`, un archivo '`web-publish.ly`' que contuviera solamente la sección `\layout` que se mostró en el ejemplo, y un archivo '`universidad.ly`' que contendría solamente los trucos para producir la salida que le gusta a mi profesor. La parte más alta de '`musica.ly`' tendría entonces este aspecto:

```
\include "definiciones.ly"

%%%  ¡Quitar el comentario de una sola de estas líneas!
\include "publicar-web.ly"
%\include "universidad.ly"
```

Este enfoque puede ser útil incluso si va a producir sólo un conjunto de particellas. Yo utilizo media docena de archivos de 'hojas de estilo' para mis proyectos. Comienzo todos los archivos de música con `\include "../global.ly"`, que contiene

```
%%%   global.ly
\version "2.11.38"
#(ly:set-option 'point-and-click #f)
```

```
\include "../iniciar/iniciar-definiciones.ly"
\include "../iniciar/iniciar-disposicion.ly"
\include "../iniciar/iniciar-cabeceras.ly"
\include "../iniciar/iniciar-papel.ly"
```

4.4 Actualizar archivos antiguos

La sintaxis de la entrada de LilyPond cambia de manera ocasional. A medida que el propio LilyPond mejora, la sintaxis (el lenguaje de la entrada) se modifica en consonancia. A veces estos cambios se hacen para conseguir que la entrada sea más fácil de leer y escribir, y otras veces estos cambios son para dar cabida a nuevas funcionalidades de LilyPond.

LilyPond lleva incorporado un archivo que facilita esta actualización: `convert-ly`. Para ver detalles sobre cómo ejecutar este programa, consulte el manual de utilización del programa, `Actualizar ficheros con convert-ly`.

Desgraciadamente `convert-ly` no puede tratar todos los cambios en la entrada. Se ocupa de los cambios sencillos de búsqueda y sustitución (como `raggedright` que se convierte en `ragged-right`), pero algunos cambios son demasiado complicados. Los cambios de sintaxis que `convert-ly` es incapaz de manejar se relacionan en el manual de utilización del programa, `Actualizar ficheros con convert-ly`.

Por ejemplo, en la versión 2.4 y anteriores de LilyPond, los acentos y las letras no inglesas se introducían utilizando LaTeX: por ejemplo, '`No\"el`' (que significa 'Navidad' en francés). En LilyPond 2.6 y siguientes, el carácter especial '`ë`' debe introducirse directamente en el archivo de LilyPond como un carácter UTF-8. `convert-ly` no puede cambiar todos los caracteres especiales de LaTeX a caracteres de UTF-8; tendrá que actualizar manualmente sus archivos de LilyPond antiguos.

4.5 Resolución de problemas (tomar cada parte por separado)

Antes o después escribirá un archivo que LilyPond no podrá compilar. Los mensajes que LilyPond proporciona pueden ayudarle a encontrar el error, pero en muchos casos tendrá que llevar a cabo algún tipo de investigación para determinar el origen del problema.

Las herramientas más poderosas para este cometido son el comentario de una sola línea (indicado por `%`) y el comentario de bloque (indicado por `%{ ... %}`). Si no sabe dónde está el problema, comience convirtiendo grandes secciones del archivo de entrada en un comentario. Después de eliminar una sección convirtiéndola en un comentario, pruebe a compilar el archivo otra vez. Si funciona, entonces el problema debía estar en la porción que había eliminado. Si no funciona, continúe eliminando material (transformándolo en comentarios) hasta que tenga algo que funcione.

En un caso extremo podría terminar con sólo

```
\score {
  <<
    % \melodia
    % \armonia
    % \bajo
  >>
  \layout{}
}
```

(en otras palabras: un archivo sin música)

Si ocurre esto, no abandone. Descomente un trozo pequeño – digamos la parte del bajo – y observe si funciona. Si no es así, transforme en comentarios toda la música del bajo (pero deje el `\bajo` de la sección `\score` no comentado.

```
bajo = \relative c' {
%{
  c4 c c c
  d d d d
%}
}
```

Ahora empiece poco a poco descomentando cada vez más fracciones de la parte del `bajo` hasta que encuentre la línea del problema.

Otra técnica de depuración muy útil es la construcción de el manual del usuario, `Ejemplos mínimos`.

4.6 Ejemplos mínimos

Un ejemplo mínimo es un ejemplo tan pequeño como sea posible. Estos ejemplos son mucho más fáciles de comprender que los ejemplos largos. Los ejemplos mínimos se utilizan para

- Informes de fallo
- Solicitudes de ayuda a las listas de correo
- Añadir ejemplos al Repositorio de Fragmentos de Código de LilyPond

Para construir un ejemplo que sea lo más pequeño posible, la regla es bastante simple: quite todo lo que no sea necesario. Al tratar de quitar partes innecesarias de un archivo, es una buena idea convertir líneas en comentarios en vez de borrarlas. De esta forma, si descubre que en realidad sí *necesitaba* algunas de estas líneas, podrá descomentarlas y no tendrá que teclearlas de nuevo partiendo de cero.

Existen dos excepciones a la regla del "lo más pequeño posible":

- Incluya el número de `\version`.
- Si puede, ponga `\paper{ ragged-right=##t }` al principio del ejemplo.

En resumen, el objetivo de un ejemplo mínimo es que sea fácil de leer:

- Evite usar notas, tonalidades o compases demasiado complicados, a no ser que quiera demostrar algo sobre el comportamiento de estos elementos precisamente.
- No use comandos `\override` a no ser que ése sea el propósito del ejemplo.

5 Trucar la salida

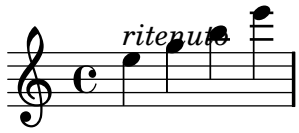
Este capítulo trata de cómo modificar la salida. LilyPond es extremadamente configurable; prácticamente todos los fragmentos de la salida se pueden cambiar.

5.1 Mover objetos

Aunque pueda sorprenderle, LilyPond no es perfecto. Ciertos elementos de notación se pueden superponer, lo que es una lástima, pero en casi todos los casos se resuelve fácilmente.

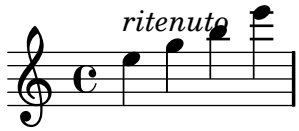
HACER: con las nuevas funcionalidades de espaciado en la versión 2.12, estos ejemplos específicos ya no son de relevancia. Sin embargo siguen demostrando las poderosas funcionalidades de lilypond, así que quedan aquí hasta que alguien elabore unos ejemplos mejores.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
e4^\markup{ \italic ritenuto } g b e
```



La solución más fácil es aumentar la distancia entre el objeto (texto en este caso, pero muy bien podrían ser digitaciones o dinámicas) y la nota. En LilyPond, esto se llama la propiedad `padding` (relleno); se mide en espacios de pentagrama. Para la mayor parte de los objetos, este valor ronda la cantidad de 1.0 o menos (varía dependiendo del objeto). Queremos aumentarlo, así que probaremos el valor 1.5

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'padding = #1.5
e4^\markup{ \italic ritenuto } g b e
```

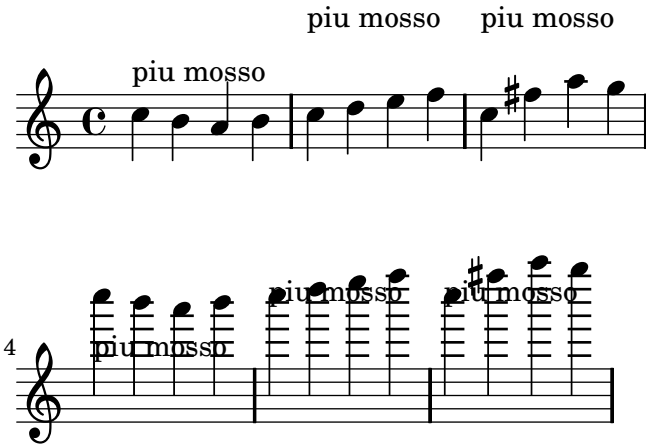


Esto tiene un mejor aspecto, pero no es suficiente. Después de probar con algunos valores, creemos que 2.3 es el mejor número en este caso. Sin embargo esta cantidad es el mero resultado del ensayo y error y de mi gusto personal acerca de la notación. Pruebe el ejemplo anterior con 2.3... pero también con otros valores mayores (y menores). ¿Cuál cree que queda mejor?

La propiedad `staff-padding` (relleno de pentagrama) está estrechamente relacionada. `padding` controla la cantidad de espacio mínima entre un objeto y el objeto más cercano (generalmente la nota o las líneas del pentagrama); `staff-padding` controla la cantidad mínima de espacio entre un objeto y el pentagrama. Ello supone una sutil diferencia, pero podrá observar el comportamiento a continuación.

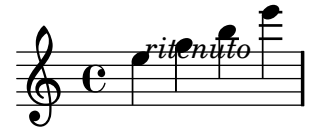
```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
```

```
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
\break
c'4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
```



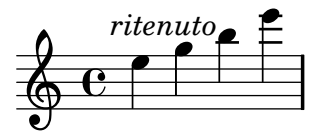
Otra solución nos proporciona un control absoluto sobre la situación del objeto: podemos moverlo horizontal o verticalmente. Se hace con la propiedad `extra-offset` (desplazamiento adicional). Es ligeramente más complicado y puede causar otros problemas. Cuando movemos objetos con `extra-offset`, el movimiento se hace después de que LilyPond haya colocado todos los demás objetos. Esto significa que el resultado podría entrar en conflicto con otros objetos.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( 1.0 . -1.0 )
e4^\markup{ \italic ritenuto } g b e
```



Con `extra-offset`, el primer número controla el movimiento horizontal (negativo hacia la izquierda); el segundo número controla el movimiento vertical (positivo hacia arriba). Después de algunos ensayos, hemos decidido que los siguientes valores son apropiados

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( -1.6 . 1.0 )
e4^\markup{ \italic ritenuto } g b e
```



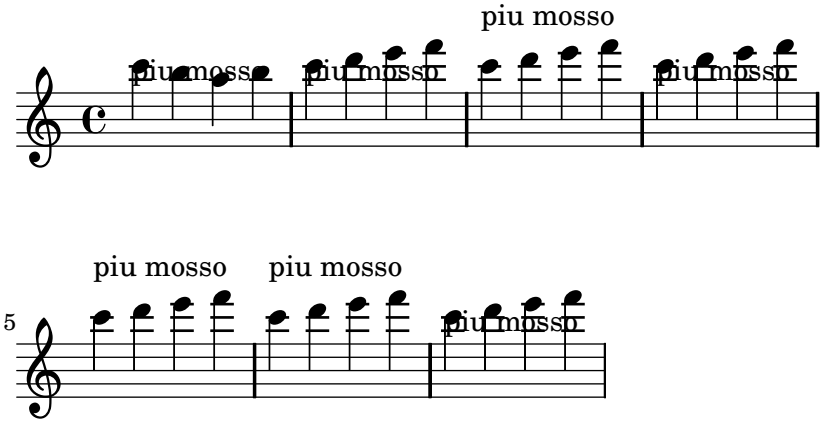
Una vez más, estos números son simplemente el resultado de algunos experimentos y de observar la salida. Quizá prefiera que el texto se encuentre algo más arriba, o a la izquierda, o en cualquier dirección. ¡Pruébelo y observe el resultado!

Una advertencia final: en esta sección hemos usado

```
\once \override TextScript ...
```

Esto altera la presentación del texto para la nota siguiente. Si la nota no tiene ningún texto, este truco no hace nada (y **no** se queda esperando al siguiente fragmento de texto). Para cambiar el comportamiento permanentemente a partir del comando, omita el `\once`. Para detener este truco, use `\revert` (revertir). Todo esto se explica en profundidad en el manual del usuario, `El comando \override`.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b
\once \override TextScript #'padding = #4.6
  a4 b
c4^"piu mosso" d e f
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\break
\override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\revert TextScript #'padding
c4^"piu mosso" d e f
```



Véase también

En el presente manual: el manual del usuario, `El comando \override`, el manual del usuario, `Trucos comunes`.

5.2 Arreglar notación con superposiciones

En el manual del usuario, `Mover objetos`, pudimos ver cómo mover un objeto `TextScript`. El mismo mecanismo se puede usar para mover otros tipos de objetos; simplemente sustituya `TextScript` con el nombre de otro objeto.

Para encontrar el nombre del objeto, consulte la sección '**véase también**' al final de la página relevante dentro de la documentación. Por ejemplo, al final de el manual del usuario, `Matices dinámicos`, vemos

Véase también

Referencia del programa: `DynamicText`, `Hairpin`. La posición vertical de estos símbolos se maneja por medio de `DynamicLineSpanner`.

Así que para mover expresiones dinámicas verticalmente, usamos

```
\override DynamicLineSpanner #'padding = #2.0
```

No podemos listar todos y cada uno de los objetos, pero presentamos a continuación una lista de los objetos más comunes.

Tipo de objeto	**Nombre del objeto**
Expresiones dinámicas (verticalmente)	`DynamicLineSpanner`
Expresiones dinámicas (horizontalmente)	`DynamicText`
Ligaduras de unión	`Tie`
Ligaduras de expresión	`Slur`
Articulaciones	`Script`
Digitaciones	`Fingering`
Texto, p.ej. `^"text"`	`TextScript`
Llamadas de ensayo o marcas de texto	`RehearsalMark`

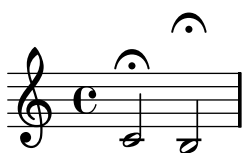
5.3 Trucos comunes

Algunas sustituciones son tan comunes que se proporcionan comandos preestablecidos como atajos, como `\slurUp` (ligadura hacia arriba) y `\stemDown` (plica hacia abajo). Estos comandos se describen dentro de la Referencia de Notación bajo las secciones correspondientes.

La lista completa de modificaciones disponibles para cada tipo de objeto (como ligaduras o barras de corchea) están documentadas en la Referencia del Programa. Sin embargo, muchos objetos de la presentación comparten propiedades que se pueden usar para aplicar trucos genéricos.

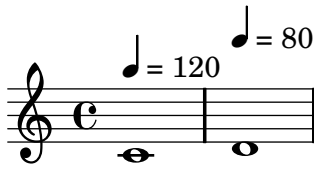
- La propiedad `padding` (relleno) se puede establecer de forma que incremente (o disminuya) la distancia entre símbolos que se imprimen encima o debajo de las notas. Se aplica a todos los objetos con `side-position-interface`.

```
c2\fermata
\override Script #'padding = #3
b2\fermata
```



```
% This will not work, see below:
\override MetronomeMark #'padding = #3
\tempo 4=120
```

```
c1
% This works:
\override Score.MetronomeMark #'padding = #3
\tempo 4=80
d1
```

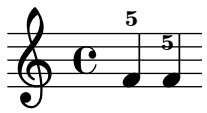


Observe en el segundo ejemplo cuán importante es determinar qué contexto maneja un objeto determinado. Debido a que el objeto `MetronomeMark` (indicación metronómica) se maneja en el contexto `Score`, los cambios de propiedades dentro del contexto `Voice` no se tendrán en cuenta. Para ver más detalles, consulte el manual del usuario, `Construir un truco`.

- La propiedad `extra-offset` mueve objetos en la salida; requiere una pareja de números. El primer número controla el movimiento horizontal, un número positivo moverá el objeto hacia la derecha. El segundo número controla el movimiento vertical; un número positivo lo desplazará hacia arriba. La propiedad `extra-offset` es una funcionalidad de bajo nivel: el motor de formateo es completamente olvidadizo respecto de estos desplazamientos.

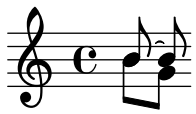
  En el ejemplo siguiente, la segunda digitación se desplaza un poco hacia la izquierda y 1.8 espacios de pentagrama hacia abajo:

```
\stemUp
f-5
\once \override Fingering
    #'extra-offset = #'(-0.3 . -1.8)
f-5
```



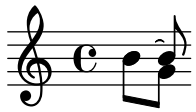
- El establecimiento de la propiedad `transparent` provocará que un objeto se imprima con 'tinta invisible': el objeto no se imprime, pero se conserva todo el resto de su comportamiento. El objeto aún ocupa un espacio, toma parte en las colisiones, y se le pueden adjuntar ligaduras de unión o de expresión y barras de corchea.

  El ejemplo siguientes demuestra cómo conectar distintas voces utilizando ligaduras. Normalmente las ligaduras sólo unen dos notas de la misma voz. Al introducir una ligadura en una voz distinta,



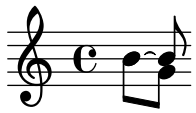
y suprimiendo la primera plica hacia arriba en dicha voz, la ligadura parece cruzarse de una voz a otra:

```
<< {
  \once \override Stem #'transparent = ##t
  b8~ b8\noBeam
} \\ {
  b[ g8]
} >>
```



Para asegurarse de que la plica que hemos suprimido no aprieta demasiado a la ligadura, también alargamos la plica, estableciendo su `length` (longitud) a `8`,

```
<< {
  \once \override Stem #'transparent = ##t
  \once \override Stem #'length = #8
  b8~ b8\noBeam
} \\ {
  b[ g8]
} >>
```



Las distancias en LilyPond se miden en espacios de pentagrama, mientras que las propiedades de grosor se miden en grosores de líneas de pentagrama. Algunas propiedades son diferentes; por ejemplo, el grosor de las barras de corchea se mide en espacios de pentagrama. Para más información, consulte la porción correspondiente de la referencia del programa.

5.4 Archivos por omisión

La documentación de la Referencia del Programa contiene una gran cantidad de información sobre LilyPond, pero más información aún se puede obtener a partir de la observación de los archivos internos de LilyPond.

Algunos ajustes por omisión como las definiciones de las `\header{}`s (encabezamientos) están almacenados en archivos `.ly`. Otros ajustes como las definiciones de los comandos de marcado se almacenan como archivos `.scm` (de Scheme). Cae fuera del ámbito de presente manual cualquier explicación más profunda; los usuarios están advertidos de que se necesita una considerable cantidad de conocimientos técnicos para comprender estos archivos.

- Linux: '*directorio_de_instalación*`/lilypond/usr/share/lilypond/current/`'
- OSX: '*carpeta_de_instalación*`/LilyPond.app/Contents/Resources/share/lilypond/current/`'. Para llegar aquí, o bien entre con `cd` en este directorio desde el Terminal, o haga control-clic sobre la aplicación LilyPond y elija 'Mostrar el Contenido del Paquete'.
- Windows: '*carpeta_de_instalación*`/LilyPond/usr/share/lilypond/current/`'

Los directorios '`ly/`' y '`scm/`' son de especial interés. Archivos como '`ly/property-init.ly`' y '`ly/declarations-init.ly`' definen todos los trucos comunes.

5.5 Encajar la música en menos páginas

A veces puede acabar con uno o dos pentagramas en una segunda página (o tercera, o cuarta...). Es fastidioso sobre todo si observa las páginas anteriores y parece haber espacio suficiente en ellas.

Al investigar asuntos relacionados con la presentación, la herramienta `annotate-spacing` (anotar el espaciado) no tiene precio. Este comando imprime los valores de algunos comandos de espaciado; consulte el manual del usuario, `Mostrar el espaciado` para ver más detalles. A partir de la salida de de `annotate-spacing`, podemos ver qué márgenes podríamos desear alterar.

Aparte de los márgenes, existen otras opciones para ahorrar espacio:

- Puede indicarle a LilyPond que coloque los sistemas tan juntos como sea posible (para que quepan tantos sistemas como sea posible sobre una página), pero luego separar estos sistemas para que no haya ningún espacio vacío al final de la página.

```
\paper {
  between-system-padding = #0.1
  between-system-space = #0.1
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

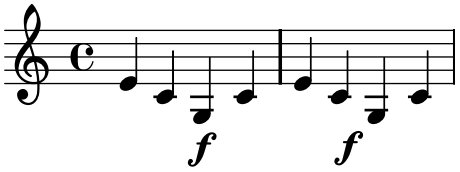
- Puede forzar el número de sistemas (es decir, si LilyPond quiere tipografiar la música con 11 sistemas, puede forzarlo para que use 10).

```
\paper {
  system-count = #10
}
```

- Evite (o reduzca) el uso de objetos que aumenten el tamaño vertical de un sistema. Por ejemplo, las repeticiones con primera y segunda vez necesitan espacio adicional. Si estas repeticiones abarcan dos sistemas, ocuparán más espacio que un solo sistema con las repeticiones y otro sistema sin ellas.

  Otro ejemplo es desplazar las expresiones dinámicas que se 'asoman por encima' de un sistema.

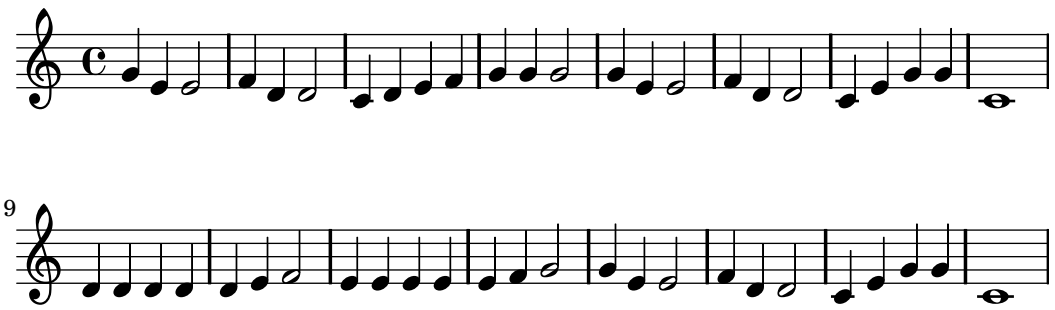
```
\relative c' {
  e4 c g\f c
  \override DynamicLineSpanner #'padding = #-1.8
  \override DynamicText #'extra-offset = #'( -2.1 . 0)
  e4 c g\f c
}
```



- Altere el espaciado horizontal por medio de `SpacingSpanner`. Consulte el manual del usuario, `Cambiar el espaciado horizontal` para ver más detalles.

```
\score {
  \relative c'' {
    g4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
```

```
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner
                #'base-shortest-duration = #(ly:make-moment 1 4)
    }
  }
}
```

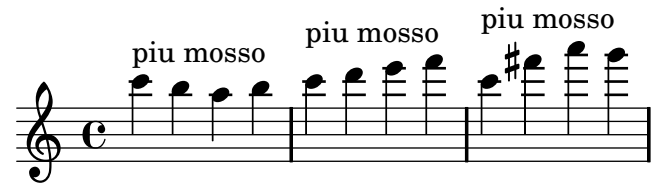


5.6 Trucos avanzados con Scheme

Hemos visto cómo la salida de LilyPond se puede modificar de manera muy profunda utilizando comandos como `\override TextScript #'extra-offset = ( 1 . -1)`. Pero tenemos un potencial incluso mayor si utilizamos Scheme. Para ver una explicación completa de esto, consulte el el manual del usuario, `Tutorial de Scheme` e el manual del usuario, `Interfaces para programadores`.

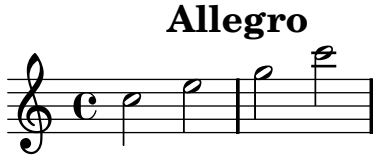
Podemos usar Scheme simplemente para sobreponer (`\override`) comandos,

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

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



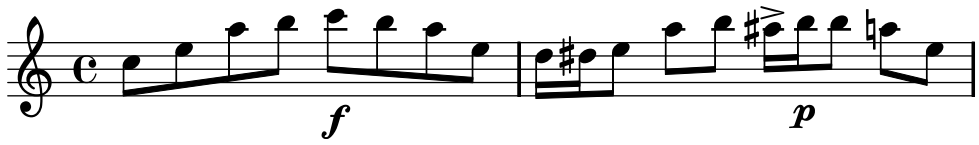
Podemos usarlo para crear comandos nuevos,

```
tempoMark = #(define-music-function (parser location padding marktext)
                                    (number? string?)
#{
  \once \override Score . RehearsalMark #'padding = $padding
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $marktext }
#})

\relative c'' {
  c2 e
  \tempoMark #3.0 #"Allegro"
  g c
}
```



E incluso se le pueden pasar expresiones musicales.

```
pattern = #(define-music-function (parser location x y) (ly:music? ly:music?)
#{
  $x e8 a b $y b a e
#})

\relative c''{
  \pattern c8 c8\f
  \pattern {d16 dis} { ais16-> b\p }
}
```



5.7 Evitar los trucos con un proceso ralentizado

LilyPond puede llevar a cabo comprobaciones adicionales al tiempo que procesa los archivos. Estos comandos consumen tiempo, pero el resultado puede necesitar menos trucos manuales.

```
%%  asegura que las marcas de texto y letras de las canciones se encuentran dentro de l
\override Score.PaperColumn #'keep-inside-line = ##t
```

Apéndice A Licencia de documentación libre de GNU

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of 'copyleft', which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The 'Document', below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as 'you'.

A 'Modified Version' of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A 'Secondary Section' is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The 'Invariant Sections' are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The 'Cover Texts' are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A 'Transparent' copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file

format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not 'Transparent' is called 'Opaque'.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The 'Title Page' means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, 'Title Page' means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled 'History', and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled 'History' in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the 'History' section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled 'Acknowledgments' or 'Dedications', preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled 'Endorsements'. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as 'Endorsements' or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to

the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled 'Endorsements', provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections entitled 'History' in the various original documents, forming one section entitled 'History'; likewise combine any sections entitled 'Acknowledgments', and any sections entitled 'Dedications'. You must delete all sections entitled 'Endorsements.'

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an 'aggregate', and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License 'or any later version' applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.0.1 ADDENDUM: cómo utilizar esta licencia para sus documentos

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
  Copyright (C)  year  your name.
  Permission is granted to copy, distribute and/or modify this document
  under the terms of the GNU Free Documentation License, Version 1.1
  or any later version published by the Free Software Foundation;
  with the Invariant Sections being list their titles, with the
  Front-Cover Texts being list, and with the Back-Cover Texts being list.
  A copy of the license is included in the section entitled `GNU
Free Documentation License'
.
```

If you have no Invariant Sections, write 'with no Invariant Sections' instead of saying which ones are invariant. If you have no Front-Cover Texts, write 'no Front-Cover Texts' instead of 'Front-Cover Texts being *list*'; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Apéndice B Índice de LilyPond

J

L

M

N

O

P

Q

R

S

T

V