Criação de novos símbolos em abcm2ps

Hudson Lacerda (2004−2006)

Conteúdo

Criação de novos símbolos em abcm2ps

Hudson Lacerda (2004−2006)

(Re)definição de operadores PostScript

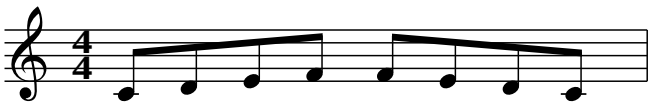
O gerador de partituras abcm2ps (**<http://moinejf.free.fr/>**) permite a criação e modificação de operadores *PostScript*. Esse recurso é muito útil na personalização das partituras, além de viabilizar a implementação de símbolos inexistentes no programa.

Para enviar uma linha de código *PostScript* para o arquivo de saída, basta utilizar o comando **%%postscript** no arquivo ABC.

O exemplo abaixo altera a espessura das linhas, redefinindo o operador **dlw** gerado por abcm2ps:

```
%%postscript /dlw {2.5 setlinewidth} def
```

dlw original (0.7pt)

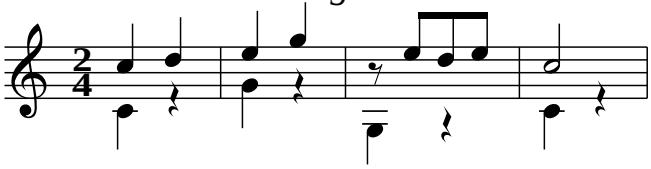


dlw redefinida (2.5pt)

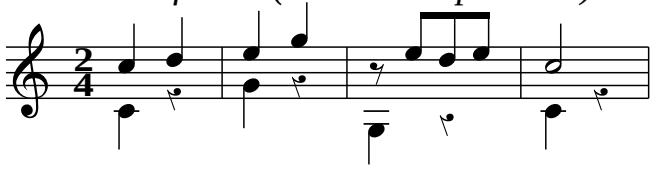
O exemplo seguinte redefine o operador **r4** − que desenha a pausa de semínima − utilizando a pausa de colcheia (**r8**) como base.

```
%%postscript /r4{gsave translate [−1 0 0 1 0 0] concat 0 0 r8 grestore}def
```

r4 original



r4 redefinida (colcheia espelhada)

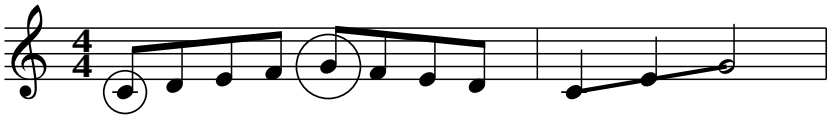


Pode ser bastante útil inserir um código *PostScript* para desenhar em uma posição precisa em relação a uma nota. O exemplo abaixo possibilita a escrita de comandos em uma anotação (*annotation*). O texto da anotação deve começar com o caractere **':'** para que seja interpretado como um comando.

```
%%postscript /cp {currentpoint}!
%%postscript /anshow { dup 0 get (:) 0 get ne
%%postscript     {/gchshow load cshow}
%%postscript     {dup length 1 sub 1 exch getinterval cvx exec}ifelse}!
```

Incluindo−se o código acima no início de um arquivo ABC, pode−se executar comandos como estes:

```
X:1
K:C
"@0,0:cp newpath 10 360 0 arcn stroke"CDEF \
"@0,0:cp newpath 15 360 0 arcn stroke"GFED |\
"@0,0:cp /yy exch def /xx exch def"C2 E2 \
"@0,0:gsave xx yy lineto 2 setlinewidth stroke grestore"G4|
```



Novos símbolos (%%deco)

Além das possibilidades já exemplificadas, abcm2ps permite o acréscimo de novas decorações (símbolos tais como acentos, dinâmicas, articulações etc.). Isso é feito através do comando **%%deco**. Sua sintaxe é a seguinte:

%%deco <nome> <tipo> <ps> <altura> <largE> <largD> [texto]

<nome> é o identificador do símbolo;
<tipo> é um número de 0 a 7 que define o tipo do símbolo;
<ps> é o nome do operador *PostScript* a ser executado;
<altura> é o espaço ocupado verticalmente pelo símbolo, em pontos;
<largE> e **<largD>** indicam o espaço à esquerda e à direita do símbolo
(funciona desde abcm2ps-4.8.2, somente para decorações do tipo 6);
[texto] é um texto (opcional) a ser utilizado pelo operador *PostScript*
(usado somente por decorações dos tipos 3, 4 e 6).

O **<tipo>** do símbolo define seu posicionamento:

0: junto à cabeça da nota, como **!tenuto!** ou **.** (*staccato*);
1: à esquerda da cabeça da nota, como **!slide!**;
2: à esquerda de um acorde, como **!arpeggio!**;
3, 4: expressões genéricas acima e abaixo da pauta respectivamente;
5: símbolo longo acima da pauta, como **!trill(!** ou **!trill)!**;
6: genérico (normalmente abaixo da pauta);
7: símbolo longo abaixo da pauta, como **!crescendo(!** ou **!crescendo)!**.

Símbolos dos tipos 0 a 2 são associados a notas, e ficam dentro da pauta. Tipos 3 a 5 situam-se fora da pauta, embora sejam associados a notas. Por sua vez, os tipos 6 e 7 são associados à pauta.

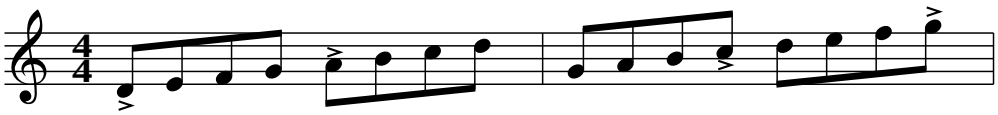
Há ainda uma categoria especial de decoração: quando o **<nome>** começa com **head-**, a decoração substitui a cabeça da nota em que for usada.

Como criar uma decoração do tipo 0

Decorações do tipo 0 situam-se junto à cabeça da nota. Dois exemplos são **!tenuto!** e **!dot!**. O operador *PostScript* deve utilizar dois argumentos: as coordenadas **<x>** e **<y>**, que determinam a posição do símbolo. O comando **%%deco** não deve fazer uso do argumento **[texto]**.

Eis um exemplo (acento junto à cabeça da nota):

```
%%postscript /exemplo0 {  moveto              % Uso: x y exemplo0 -
%%postscript              -3 -2 rmoveto
%%postscript               6  2 rlineto
%%postscript              -6  2 rlineto
%%postscript              1 setlinewidth stroke dlw
%%postscript           } def

%%deco ex0 0 exemplo0 6 0 0

X:1
K:C
!ex0!DEFG !ex0!ABcd | GAB!ex0!c def!ex0!g |
```

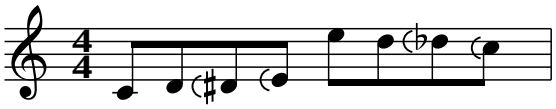


Como criar uma decoração do tipo 1

Decorações do tipo 1 situam−se à esquerda da cabeça da nota. Um exemplo é **`!slide!`**. Assim como para o tipo 0, o operador *PostScript* deve utilizar dois argumentos: as coordenadas **`<x>`** e **`<y>`**, que determinam a posição do símbolo. O comando **`%%deco`** não deve fazer uso do argumento **`[texto]`**. O argumento **`<altura>`** é irrelevante, podendo ser simplesmente zero. Note que a posição do símbolo é afetada pela existência de um acidente.

Eis um exemplo (meia−lua à esquerda da nota):

```
%%postscript /exemplo1 {  newpath             % Uso: x y exemplo1 −
%%postscript              exch 1 add exch
%%postscript              5 100 260 arc
%%postscript              .9 setlinewidth stroke dlw
%%postscript           } def

%%deco ex1 1 exemplo1 0 0 0

X:1
K:C
CD!ex1!^D!ex1!E ed!ex1!_d!ex1!c |
```



Como criar uma decoração do tipo 2

Decorações do tipo 2 situam−se à esquerda de um acorde, e possuem altura dependente do âmbito desse acorde. Um exemplo é **`!arpeggio!`**.

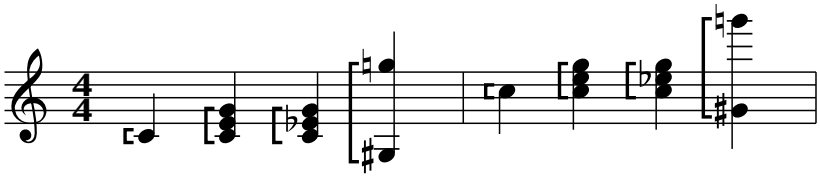
O operador *PostScript* deve utilizar três argumentos: a altura mínima do símbolo e as coordenadas horizontal e vertical do ponto mais baixo do símbolo.

O comando **`%%deco`** não deve fazer uso do argumento **`[texto]`**. O argumento **`<altura>`** é utilizado para definir a altura mínima do símbolo (desde `abcm2ps−4.8.8`).

A posição do símbolo é afetada pela existência de acidentes.

Eis um exemplo (colchete):

```
%%postscript /exemplo2 {   moveto             % Uso: alt x y exemplo2 −
%%postscript               4 0 rmoveto
%%postscript              −4 0 rlineto 0 exch rlineto
%%postscript               4 0 rlineto 1.6 setlinewidth stroke dlw
%%postscript           } def

%%deco ex2 2 exemplo2 6 0 0

X:1
L:1/4
K:C
!ex2!C !ex2![CEG] !ex2![C_EG] !ex2![^G,=g] |\
!ex2!c !ex2![ceg] !ex2![c_eg] !ex2![^G=g'] |
```

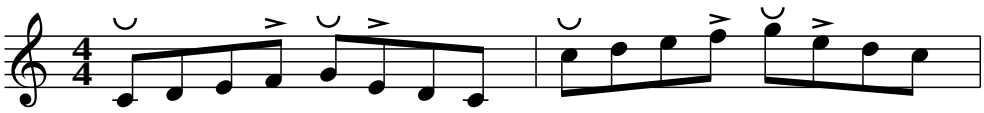


Como criar uma decoração do tipo 3

Decorações do tipo 3 situam−se geralmente acima da pauta. Alguns exemplos são: **!accent!**, **!segno!** e **!mordent!**. O operador *PostScript* deve utilizar dois ou três argumentos: as coordenadas horizontal e vertical e, opcionalmente, uma *string* de texto (se o comando **%%deco** empregar o argumento **[texto]**).

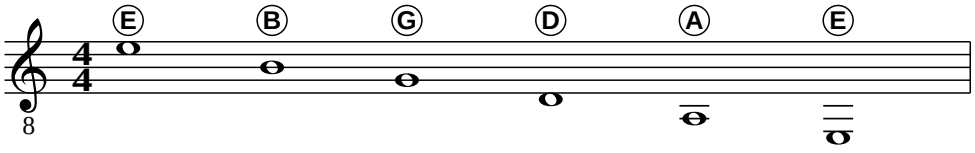
Eis um exemplo sem **[texto]** (não−acento):

```
%%postscript /exemplo3a {  newpath            % Uso: x y exemplo3a −
%%postscript               6.5 add
%%postscript               5 180 360 arc
%%postscript               1.2 setlinewidth stroke dlw
%%postscript            } def

%%deco ex3a 3 exemplo3a 8 0 0

X:1
K:C
!ex3a!CDELF !ex3a!GLEDC | !ex3a!cdeLf !ex3a!gLedc |
```



E a seguir um exemplo com **[texto]** (indicação de corda):

```
%%postscript /exemplo3b {  4 add 2 copy       % Uso: texto x y exemplo3b −
%%postscript               newpath 4 add 7 0 360 arc dlw stroke
%%postscript               moveto
%%postscript               /Helvetica−Bold 12 selectfont
%%postscript               showc
%%postscript            } def

%%deco ex3b_E 3 exemplo3b 16 0 0 E
%%deco ex3b_B 3 exemplo3b 16 0 0 B
%%deco ex3b_G 3 exemplo3b 16 0 0 G
%%deco ex3b_D 3 exemplo3b 16 0 0 D
%%deco ex3b_A 3 exemplo3b 16 0 0 A

X:1
L:1/1
K:C treble−8
!ex3b_E!e !ex3b_B!B !ex3b_G!G !ex3b_D!D !ex3b_A!A, !ex3b_E!E, |
```



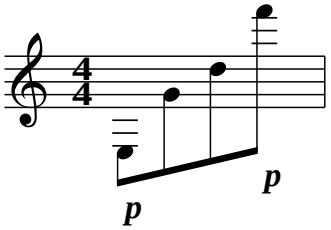
Como criar uma decoração do tipo 4

Decorações do tipo 4 situam−se sempre abaixo da pauta.

De maneira similar aos tipos 3 e 6, o operador *PostScript* deve utilizar dois ou três argumentos: as coordenadas horizontal e vertical e, opcionalmente, uma *string* de texto (se o comando **%%deco** empregar o argumento **[texto]**).

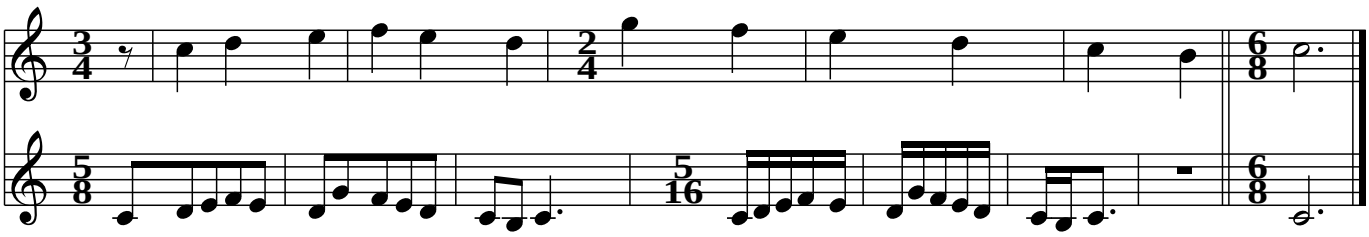
Um exemplo de aplicação é em símbolos de dinâmica próximos às notas (ao invés de serem alinhados horizontalmente). O operador **pf** de abcm2ps (próprio para imprimir símbolos de dinâmica) é aproveitado no exemplo. (Veja também ‘‘Como criar uma decoração do tipo 6’’.)

```
%%deco ex4a 4 pf 18 0 0 p

X:1
K:C treble
!ex4a!E,Gd!ex4a!f' |
```



Outra aplicação de decoração do tipo 4 é o desenho de barras de compasso, útil para escrita de música polimétrica.

```
%%postscript /exemplo4b{2 add M 0 24 RL dlw stroke}def
%%deco ex4b 4 exemplo4b 0 0 0

U: I = !ex4b!      % Usa 'I' como abreviação para '!ex4b!'

X:1
M:none
K:none
V:1
[M:3/4][L:1/4] z/  Iy  cde Iy  fed  Iy  \
[M:2/4]        gf  Iy  ed  Iy  cB || [M:6/8] c3 |]
V:2
[M:5/8] [L:1/8]  CDEFE  Iy  DGFED  Iy  CB,C3  Iy  \
[M:5/16][L:1/16] CDEFE  Iy  DGFED  Iy  CB,C3  Iy  z5 || [M:6/8] C12 |]
```



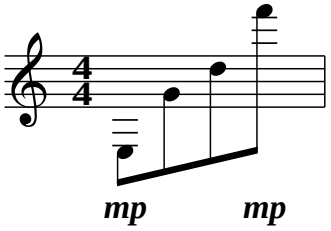
Como criar uma decoração do tipo 6

Decorações do tipo 6 situam−se geralmente abaixo da pauta. Exemplos típicos são os símbolos de dinâmica, tais como: **!p!**, **!mf!** e **!ff!**.

O operador *PostScript* deve utilizar necessariamente três argumentos: as coordenadas horizontal e vertical e uma *string* de texto. O argumento **[texto]** do comando **%%deco** não pode ser omitido.

Um exemplo simples e útil é a implementação da dinâmica *mp* (ausente antes de abcm2ps−4.11.5). O operador **pf** de abcm2ps é usado no exemplo. Note que as decorações do tipo 6 são alinhadas horizontalmente (diferentemente do tipo 4).

```
%%deco ex6 6 pf 18 10 8 mp

X:1
K:C treble
!ex6!E,Gd!ex6!f' |
```



Como criar uma decoração do tipo 5

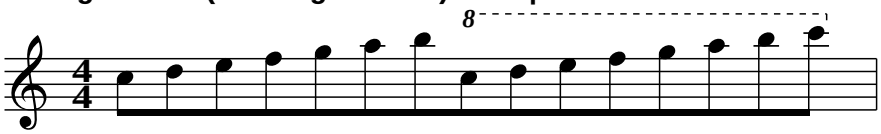
Decorações do tipo 5 são sinais longos que situam−se acima da pauta. Um exemplo é o trinado longo, iniciado com **!trill(!** e terminado com **!trill)!**.

Na decoração que inicia o símbolo, o campo **<ps>** do comando **%%deco** deve ser **'−'** (indicando que nenhum operador *PostScript* será utilizado). O **<nome>** da decoração inicial deve terminar com o caractere **'('**, e o da decoração final com **')'**.

O operador *PostScript* da decoração final é responsável pelo desenho, e recebe três argumentos: a largura do símbolo e as coordenadas horizontal e vertical do ínicio do símbolo.

Eis um exemplo (indicação de oitava acima):

```
%%postscript /exemplo5 {                    % Uso: larg x y exemplo5 −
%%postscript              moveto −3 2 rmoveto
%%postscript              /Times−BoldItalic 12 selectfont (8) show
%%postscript              gsave
%%postscript                 2 6 rmoveto
%%postscript                 [3] 0 setdash 12 add 0 rlineto
%%postscript                 currentpoint dlw stroke
%%postscript                 moveto [] 0 setdash 0 −3 rlineto stroke
%%postscript              grestore
%%postscript           } def

%%deco ex5( 5 − 10 0 0
%%deco ex5) 5 exemplo5 10 0 0

X:1
K:C
cdefgab!ex5(!cdefgab!ex5)!c' |
```



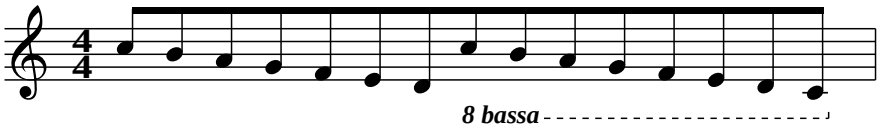
Como criar uma decoração do tipo 7

Decorações do tipo 7 são sinais longos que situam−se abaixo da pauta, como os sinais de dinâmica **!crescendo(!**, **!crescendo)!**, **!diminuendo(!** e **!diminuendo)!**.

São criadas da mesma maneira que as decorações do tipo 5, com a diferença que as do tipo 7 são impressas abaixo da pauta.

Eis um exemplo (indicação de oitava abaixo):

```
%%postscript /exemplo7 {                    % Uso: larg x y exemplo7 −
%%postscript              moveto −6 0 rmoveto
%%postscript              /Times−BoldItalic 12 selectfont (8 bassa) show
%%postscript              gsave
%%postscript                 2 2 rmoveto
%%postscript                 [3] 0 setdash 22 sub 0 rlineto
%%postscript                 currentpoint dlw stroke
%%postscript                 moveto [] 0 setdash 0 3 rlineto stroke
%%postscript              grestore
%%postscript           } def

%%deco ex7( 7 − 10 0 0
%%deco ex7) 7 exemplo7 10 0 0

X:1
K:C
cBAGFED!ex7(!cBAGFED!ex7)!C |
```



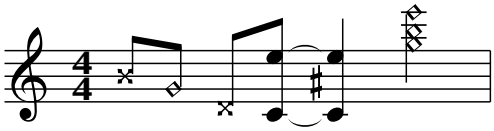
Como criar uma decoração head−xxx (cabeça de nota)

Uma decoração cujo **<nome>** começa com **head−** substitui a cabeça da nota em que é usada. Decorações **head−xxx** são declaradas como sendo de qualquer um dos tipos 0−7, usando os parâmetros apropriados ao tipo declarado.

Alguns exemplos úteis são as cabeças de nota em forma de losango (harmônico), em forma de **x** (percussão) e invisíveis (notas sem cabeça). Neste exemplo, o **x** é na verdade um *dobrado sustenido* (desenhado pelo operador *PostScript* **dsh0**, gerado por abcm2ps). Note ainda o efeito de uma decoração **head−xxx** em um acorde: ela se aplica a todas as suas notas.

Veja também a seção ‘‘Modificando as cabeças das notas globalmente’’.

```
%%postscript /harm{gsave T 40 rotate          % Uso: x y harm −
%%postscript             .25 dup setlinewidth
%%postscript             −3 2.3 3 −1 roll 2 div sub M 6 0 RL
%%postscript             0 −4.6 RM                   −6 0 RL stroke
%%postscript             1.4 dup setlinewidth
%%postscript             3 1 index 2 div sub 2.3 M 0 −4.6 RL
%%postscript             −6 add 0 RM               0  4.6 RL stroke
%%postscript       grestore}!
%%postscript /pop2{pop pop}!

%%deco head−h 0 harm 0 0 0
%%deco head−x 0 dsh0 0 0 0
%%deco head−i 0 pop2 0 0 0

X:1
K:C
!head−x!B!head−h! G!head−x!D[Ce] − [C!head−i!^Ae]2 !head−h![gbe’]2 |
```

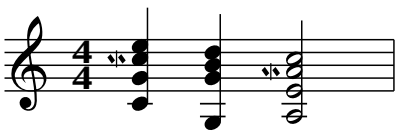


Decorações em notas individuais de acordes

Uma decoração pode ser usada em uma nota específica pertencente a um acorde (*i.e.* uma nota contida entre **‘[’** e **‘]’**). Nesse caso, uma decoração comum pode ficar em uma posição inadequada. Então deve−se utilizar um operador *PostScript* (e uma decoração) especial para uso em nota individual de acorde.

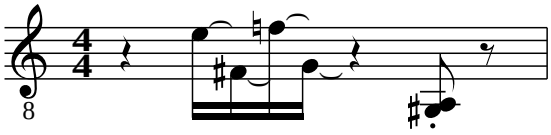
Como primeiro exemplo, um mordente (baseado no operador **lmrd** de abcm2ps) que é posicionado à esquerda de uma cabeça de nota. Ele foi declarado como uma decoração tipo 1 para reservar um pouco de espaço à esquerda.

```
%%postscript /morde{exch 11 sub exch 4 sub lmrd}!
%%deco morde 1 morde 0 0 0

X:1
K:C
[CG!morde!ce]2 [G,GBd]2 [A,E!morde!Ac]4 |
```



E a seguir, ligaduras abertas que indicam "*laissez vibrer*".

```
%%postscript /lv { gsave T (')eq{1 -1 scale}if -90 rotate 3 7 M
%%postscript       /Times-Roman 12 selectfont (\)) show grestore } !
%%deco lv' 3 lv 0 0 0 '
%%deco lv, 3 lv 0 0 0 ,

X:1
K:C treble-8
z2 [!lv'!e]/[!lv,!^F]/[!lv'!=f]/[!lv,!G]/ z2 .[^G,A,] z |
```



Acidentes microtonais

Desde a versão 4.3.2, abcm2ps permite o uso de acidentes especiais para microtonalismo. Alturas microtonais são indicadas por uma fração depois de um acidente, como **^3/4c**. Por *default*, o numerador é definido como 1 e o denominador como 2 (**^/c** é o mesmo que **^1/2c**). O numerador e o denominador devem ser números inteiros de 1 a 256.

abcm2ps inclui sustenidos e bemóis 1/2 e 3/2 (para quartos de tom). Para outros valores, novos operadores *PostScript* devem ser definidos (através de **%%postscript**). O nome de um operador *PostScript* para acidente deve ser:

<accidental_type><micro_value>

onde **<accidental_type>** deve ser: **sh** (sustenido), **ft** (bemol), **nt** (bequadro), **dsh** (dobrado sustenido) ou **dft** (dobrado bemol).

<micro_value> deve ser um número computado da fração como:

(numerador−1)*256 + (denominador−1).

No exemplo a seguir, a primeira linha de música utiliza os acidentes disponíveis em abcm2ps. A segunda linha utiliza acidentes alternativos, constituídos de um sustenido ou bemol com uma seta. No código ABC desse exemplo, um acidente com **9/** (ou **9/2**) usa seta para cima, e um acidente com **6/** (ou **6/2**) usa seta para baixo.

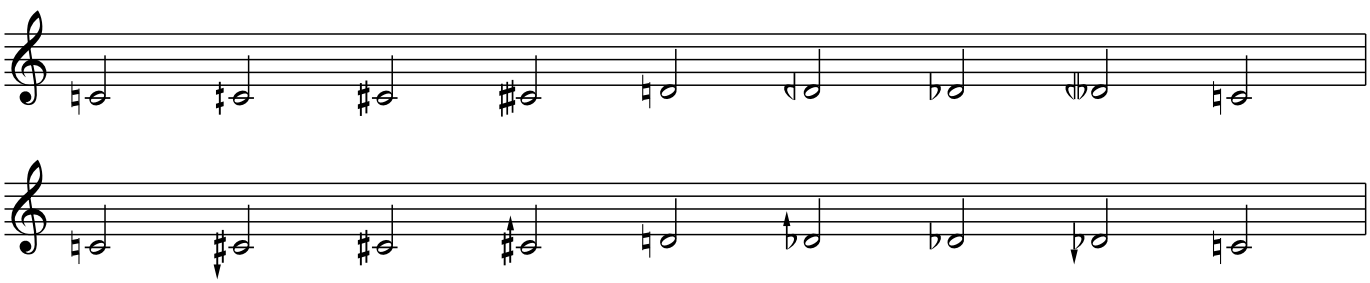
Note os valores de **<micro_value>** nos nomes dos operadores *PostScript*, calculados para as frações **9/2** e **6/2**:

(**9**−1)*256 + (**2**−1) = **2049**

(**6**−1)*256 + (**2**−1) = **1281**

Eis o código:

```
%%postscript /seta {M 1.7 −7 RL −3.4 0 RL fill}!
%%postscript /sh2049 {2 copy sh0 exch 1.5 add exch 15 add seta}! % ^9/2
%%postscript /sh1281 {2 copy sh0 exch 1.2 sub exch 15 sub       % ^6/2
%%postscript          gsave T 180 rotate 0 0 seta grestore}!
%%postscript /ft2049 {2 copy ft0 exch 1.8 sub exch 14 add seta}! % _9/2
%%postscript /ft1281 {2 copy ft0 exch 1.8 sub exch 11 sub       % _6/2
%%postscript          gsave T 180 rotate 0 0 seta grestore}!

X:1
M:none
L:1/2
K:C
=C  ^/C ^C ^3/C =D  _/D _D _3/D =C |
=C ^6/C ^C ^9/C =D _9/D _D _6/D =C |
```



Alguns operadores especiais

O código *PostScript* gerado por abcm2ps implementa diversos operadores úteis na escrita de extensões (**%%deco / %%postscript**). Merecem destaque: abreviações para comandos *PostScript*, o sinalizador **defl** e os auxiliares de posicionamento **y<N>**.

Abreviações

abcm2ps gera as seguintes abreviações para comandos da linguagem *PostScript*:

```
!       bind def
T       translate
M       moveto
RM      rmoveto
RL      rlineto
RC      rcurveto
SLW     setlinewidth (abcm2ps-4.8.3)
```

defl

O operador **defl** (introduzido com abcm2ps-4.8.0) sinaliza:

- o estado de uma decoração de tipo 5 ou 7 (início ou final);
- a direção da haste da nota (para cima ou para baixo).

Ele é gerado/atualizado para cada nota que contenha uma decoração.

Para utilizá-lo, a partir de abcm2ps-4.9.4, é necessário acionar o comando **%%setdefl true**.

O primeiro bit à direita em **defl** indica se uma decoração longa está começando (**0**) ou se é uma continuação (**1**) (refere-se à extremidade esquerda da decoração);

O segundo bit a partir da direita indica se uma decoração longa está terminando (**0**) ou se ficará por continuar na próxima linha (**1**) (refere-se à extremidade direita da decoração);

O terceiro bit a partir da direita indica se a haste da nota está para baixo (**0**) ou para cima (**1**). É útil para criar um decoração dependente da direção da haste (por exemplo, a ser desenhada na própria haste).

Assim, para verificar o estado do primeiro bit (extremidade esquerda de decoração longa) pode-se usar uma seqüência *PostScript* com a seguinte estrutura:

```
defl 1 and 0 eq {
    % procedimento caso ponta esquerda seja início
}{
    % procedimento caso ponta esquerda seja continuação
} ifelse
```

Para verificar o segundo bit (extremidade direita de decoração longa):

```
defl 2 and 0 eq {
    % procedimento caso ponta direita seja final
}{
    % procedimento caso ponta direita seja a continuar
} ifelse
```

E para verificar o terceiro bit (direção da haste):

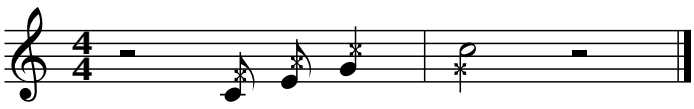
```
defl 4 and 0 eq {
    % procedimento caso haste seja para baixo
}{
    % procedimento caso haste seja para cima
} ifelse
```

Exemplos:

O exemplo abaixo utiliza **defl** para determinar onde está a haste de modo a nela desenhar um **x** (*sprechgesang*).

A decoração é declarada como de tipo 1 porque sua posição é sempre relativa à cabeça da nota. Como efeito colateral, abcm2ps deixa um espaço extra à esquerda da nota (onde seria normalmente desenhado um tal tipo de decoração). Caso esse espaço seja indesejável, pode−se declarar um tipo 0 (ou 3) e sempre usar a decoração (juntamente com a nota) entre colchetes, como num acorde.

```
%%setdefl true
%%postscript /sprgsg{gsave T
%%postscript         defl 4 and 0 eq{1.5 −9}{8.5 9}ifelse T
%%postscript         .8 dup scale 0 0 dsh0 grestore}!

%%deco sprgsg 1 sprgsg 0 0

X:1
K:C
z4 !sprgsg!C !sprgsg!E !sprgsg!G2 | !sprgsg!c4 z4 |]
```

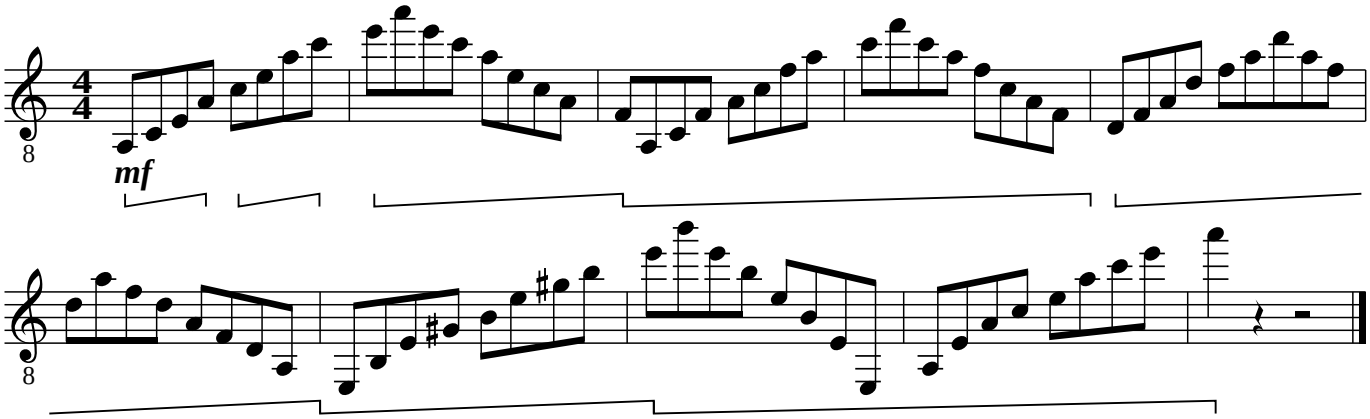


No seguinte exemplo é implementado um colchete abaixo da pauta (que pode ser usado para indicações de pedal de piano). O início e o final dessa decoração longa são reconhecidos graças ao operador **defl**. Observe (na partitura) que o colchete que começa no último compasso da primeira linha fica ‘aberto’, assim como o lado esquerdo de sua continuação na linha seguinte.

A primeira linha do código *PostScript* define o operador **defl** com zeros. Isso é totalmente desnecessário (e sem efeito) com abcm2ps−4.8.0 ou mais recente, mas torna o código compatível com versões anteriores. Ao usar uma versão anterior, a decoração sempre será desenhada com início e fim.

Outra observação é que as decorações de tipos 6 e 7 são normalmente usadas para sinais de dinâmica, portanto alinhadas na horizontal. Para evitar conflito com esses sinais, a indicação de pedal do exemplo seguinte é desenhada abaixo do espaço vertical a ela reservado. Em conseqüência, podem ocorrer colisões com elementos da pauta seguinte. A solução é aumentar a distância entre as pautas, com **%%staffsep** e/ou **%%sysstaffsep**.

```
%%setdefl 1
%%postscript /defl 0 def
%%postscript /colchete { −6 add exch 3 sub exch .9 setlinewidth
%%postscript             defl 1 and 0 eq{6 add M 0 −6 RL}{M}ifelse
%%postscript             7 add 6 RL
%%postscript             defl 2 and 0 eq{0 −6 RL}if
%%postscript             stroke dlw}!

%%deco colchete( 7 − 2 0 0
%%deco colchete) 7 colchete 2 0 0

X:1
%%staffsep 80pt          % bastante espaço entre as pautas
K:Am clef=treble−8
%
!colchete(!!mf!A,CE!colchete)!A !colchete(!cea!colchete)!c’ |\
!colchete(!e’a’e’c’ aecA |\
!colchete)! !colchete(! FA,CF Acfa  |  c’f’c’a fcAF  !colchete)! |\
!colchete(! DFAd fad’af  |
%
dafd AFDA,  !colchete)! !colchete(! |\
E,B,E^G Be^gb  |  !colchete)! !colchete(! e’b’e’b eBEE,  |\
A,EAc eac’e’  |  !colchete)! a’2 z2 z4  |]
```



Os mesmos princípios do exemplo acima poderiam ser aplicados para aprimorar os exemplos de decorações tipos 5 e 7 dados anteriormente (linhas de oitava acima e abaixo).

y0, y1, y2... y<n−1>

abcm2ps−4.8.6(?) introduz os operadores **y0**, **y1**, **y2**... **y<n−1>**, que auxiliam no posicionamento de símbolos em relação às pautas. Para todo sistema de pautas da música, abcm2ps define um operador **y<N>** associado a cada pauta (**y0** para a superior, **y1** para a segunda e assim por diante até **y<n−1>** para a **n**−ésima pauta). As definições são como esta:

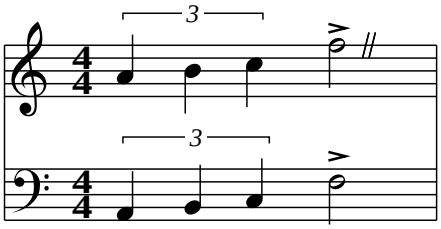
```
/y0{−47.0 add}def
```

Quando executados, esses operadores *PostScript* adicionam ao valor do topo da pilha a distância vertical entre o eixo *x* e a primeira linha da respectiva pauta.

A implementação de certas decorações pode tirar proveito dos operadores **y<N>**. Por exemplo, a indicação de *cesura* deve ter seu posicionamento vertical fixo em relação à pauta, sem sofrer influência da posição das notas como ocorre com outras decorações.

No exemplo abaixo, a posição vertical original da decoração é descartada (**pop**), sendo substituída pela posição da quarta linha da pauta, situada 18 pontos acima da primeira linha (**18 y0**).

```
%%postscript /caesura0 { pop 18 y0 M
%%postscript             12 0 RM 3 12 RL 3 0 RM −3 −12 RL
%%postscript             dlw stroke }!

%%deco caesura0 3 caesura0 0 0 0

X:1
K:Am
[V:1 treble] (3A2B2c2 !caesura0!Lf4 |
[V:2 bass]   (3A2B2c2 !caesura0!Lf4 |
```



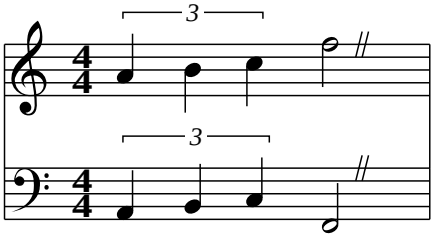
Útil para músicas a uma voz, a *cesura* definida acima não aparece na pauta inferior (voz 2), já que utiliza apenas o operador **y0**. (Substituir **y0** por **y1** colocará as decorações na outra pauta.)

Para criar decorações mais genéricas usando **y<N>** será necessário identificar a qual pauta a decoração está associada. O código a seguir permite identificar a pauta mais próxima à posição vertical *y* associada à decoração.

```
% ! RECURSOS PARA ADIVINHAR PENTAGRAMA (STAFF GUESSING RESOURCES) !

% Suporte para até seis pautas (Up to 6 staves support)

%%postscript /y0{10000 sub}!
%%postscript /y1{10000 sub}!
%%postscript /y2{10000 sub}!
%%postscript /y3{10000 sub}!
%%postscript /y4{10000 sub}!
%%postscript /y5{10000 sub}!

% Retorna altura da primeira linha da pauta mais próxima
% Return vertical offset of the nearest staff

%%postscript /staff_y_offset { % stack:  y  ->  y'
%%postscript     dm_guess_staff dm_y_offset} !

% Encontra pauta mais próxima e retorna o índice da pauta
% Find nearest staff

%%postscript /dm_guess_staff{ % stack:  y  ->  index
%%postscript     dup 12 sub neg y0 abs exch % y y0-midstaff y
%%postscript     dup 12 sub neg y1 abs exch % y y0-midstaff y1-midstaff y
%%postscript     dup 12 sub neg y2 abs exch
%%postscript     dup 12 sub neg y3 abs exch
%%postscript     dup 12 sub neg y4 abs exch
%%postscript         12 sub neg y5 abs      % y0..y5-midstaff
%%postscript     4 dict begin
%%postscript         /dm_i 6 def % max number of staves
%%postscript         /dm_min 10000 def
%%postscript         /min_ {2 copy gt {exch} if pop}!
%%postscript         dm_i {
%%postscript            /dm_temp exch def dm_min dm_temp min_ dm_temp eq{
%%postscript               /dm_i dm_i 1 sub def/dm_min dm_temp def}if
%%postscript         } repeat dm_i
%%postscript     end
%%postscript }bind def

% Obtém offset de y<N>
% Get offset of y<N>

%%postscript /dm_y_offset { % stack:  staff-index  ->  y-offset
%%postscript     dup 0 eq {pop 0 y0} {
%%postscript        dup 1 eq {pop 0 y1} {
%%postscript           dup 2 eq {pop 0 y2} {
%%postscript              dup 3 eq {pop 0 y3} {
%%postscript                  dup 4 eq {pop 0 y4} {
%%postscript                      5 eq {0 y5} {0} ifelse
%%postscript                  } ifelse
%%postscript              } ifelse
%%postscript           } ifelse
%%postscript        } ifelse
%%postscript     } ifelse
%%postscript }bind def
```

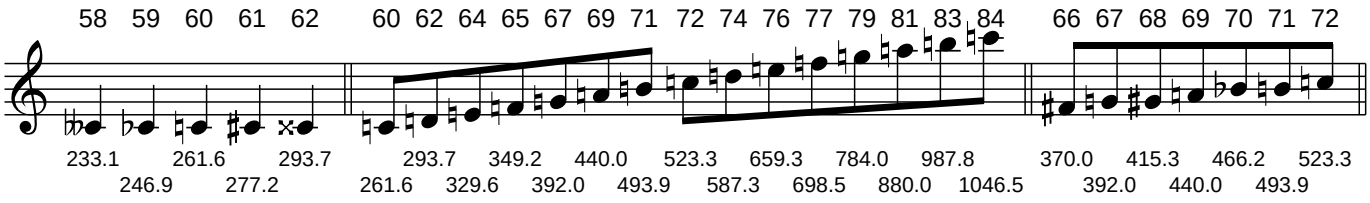
Usando as definições acima, segue-se uma outra implementação de *cesura*, que coloca a decoração na pauta mais próxima. (Em alguns casos, notas em linhas suplementares e redução do número de pautas usadas podem causar posicionamento incorreto das decorações.)

```
%%postscript /caesura { staff_y_offset 18 add M
%%postscript            12 0 RM 3 12 RL 3 0 RM -3 -12 RL dlw stroke }!
%%deco caesura 3 caesura 0 0 0

X:1
K:Am
[V:1 treble] (3A2B2c2 !caesura!f4 |
[V:2 bass]   (3A2B2c2 !caesura!F4 |
```



Outras aplicações para **y<N>** incluem barras de compasso especiais (pode−se aprimorar o exemplo4b) e seleção automática de cabeças de notas para "dó móvel" e "*shaped notes*". Ou até mesmo a identificação numérica das alturas e suas correspondentes freqüências através dos operadores de acidentes (**sh0**, **nt0**, **ft0**, etc.), como no exemplo seguinte:

```
%%postscript /sh0−orig/sh0 load bind def
%%postscript /ft0−orig/ft0 load bind def
%%postscript /nt0−orig/nt0 load bind def
%%postscript /dsh0−orig/dsh0 load bind def
%%postscript /dft0−orig/dft0 load bind def

%%postscript /deslocamento true def
%%postscript /calc−altura{gsave                % Uso: incr calc−altura −
%%postscript     y neg y0 neg 3 div 3 add round cvi 1 sub 7 5 mul add
%%postscript     /altura exch def
%%postscript     altura 7 mod 0 eq {0} if
%%postscript     altura 7 mod 1 eq {2} if
%%postscript     altura 7 mod 2 eq {4} if
%%postscript     altura 7 mod 3 eq {5} if
%%postscript     altura 7 mod 4 eq {7} if
%%postscript     altura 7 mod 5 eq {9} if
%%postscript     altura 7 mod 6 eq {11} if
%%postscript     add altura 7 idiv 12 mul add dup /altura exch def
%%postscript     10 string cvs
%%postscript     x 41 y0 M /Helvetica 12 selectfont showc
%%postscript     /deslocamento deslocamento not def
%%postscript     deslocamento {x −36} {x −24} ifelse
%%postscript     y0 M altura mid2cps 10 mul round 10 div
%%postscript     10 string cvs /Helvetica 10 selectfont showc
%%postscript     grestore}!
%     Observação: Os operadores ‘x’ e ‘y’ armazenam a posição da nota.

%%postscript /mid2cps{69 sub 440.0 2.0 3 −1 roll 12.0 div exp mul}!

%%postscript /sh0{1 calc−altura sh0−orig}!    % sustenido
%%postscript /ft0{−1 calc−altura ft0−orig}!   % bemol
%%postscript /nt0{0 calc−altura nt0−orig}!    % bequadro
%%postscript /dsh0{2 calc−altura dsh0−orig}!  % dobrado sustenido
%%postscript /dft0{−2 calc−altura dft0−orig}! % dobrado bemol

X:1
K:C
__C2 _C2 =C2 ^C2 ^^C2 || \
=C=D=E=F=G=A=B =c=d=e=f=g=a=b=c’ || ^F=G^G=A_B=B=c ||
```



Modificando as cabeças das notas globalmente

Em alguns casos é desejável modificar globalmente as cabeças das notas, para 'estilizar' a partitura ou criar recursos especiais. Para isso basta redefinir os operadores *PostScript* responsáveis pelo desenho das cabeças de notas.

Os operadores de abcm2ps que desenham cabeças de notas são:

hd Cabeça preta, para semínimas e notas mais curtas
Hd Cabeça de mínima
HD Semibreve
HDD Breve (formato arredondado)
breve Breve (formato quadrado)
longa Longa
pshhd Usada em nota com sustenido, quando a clave é **perc** (abcm2ps−4.4.5)
pfthd Usada em nota com bemol, quando a clave é **perc** (abcm2ps−4.12.1 − era **pflhd**)
ghd Cabeça preta para notinhas (ornamentos)

Podem−se ainda definir estes (que não têm código default):

pnthd Usada em nota com bequadro, quando a clave é **perc** (abcm2ps−4.12.1)
pdshhd Usada em nota com dobrado sustenido, quando a clave é **perc** (abcm2ps−4.12.1)
pdfthd Usada em nota com dobrado bemol, quando a clave é **perc** (abcm2ps−4.12.1)

Pausas são realizadas por:

r00 Pausa de longa
r0 Pausa de breve
r1 Pausa de semibreve
r2 Pausa de mínima
r4 Pausa de semínima
r8 Pausa de colcheia
r16 Pausa de semicolcheia
r32 Pausa de fusa
r64 Pausa de semifusa
r128 Pausa de quartifusa (também chamada trifusa)

É importante observar que todos esses operadores armazenam a posição da nota ou pausa como **x** e **y**. Eles são usados para posicionar outros símbolos, como hastes e pontos de aumento. Isso implica que, para escrever outras versões de cabeças de notas ou pausas, *é imperativo definir* **x** *e* **y**. Há dois modos de fazer isso:

(1) Utilizar o operador **xymove** de abcm2ps, que move para o ponto especificado e salva suas coordenadas como **x** e **y**.

(2) Definir **x** e **y** 'manualmente', caso não se deseje realizar nenhum movimento. Isso pode ser realizado com:

(2.a) **/y exch def /x exch def**

(que pega os dois valores do topo da pilha e os salva como **x** e **y**);

ou com:

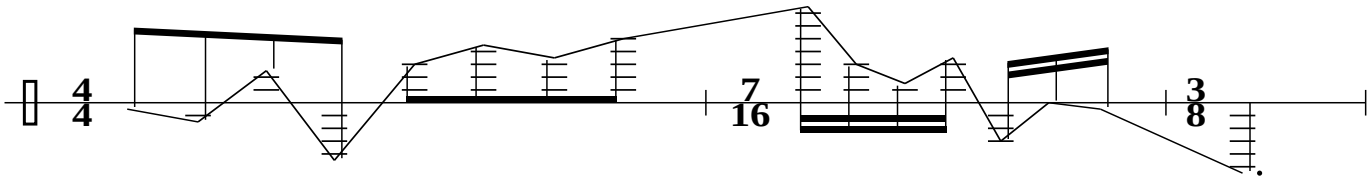
(2.b) **/x 2 index def /y 1 index def**

(que salva os dois valores do topo da pilha *sem deletá−los*).

Eis um exemplo de redefinição de cabeça de nota para desenhar um contorno melódico:

```
%%postscript /hdo/hd load bind def     % guarda cópia de 'hd' original
%%postscript /xx 10000 def /yy 0 def   % inicializa 'nota anterior'
%%postscript /hd { xymove              % liga à nota anterior (esquerda)
%%postscript       x xx gt {xx yy M x y lineto stroke} if
%%postscript       /xx x def /yy y def}!
```

```
X:1
L:1/8
M:none
K:C clef=perc stafflines=1  % (stafflines foi introduzido com abcm2ps-4.8.6)
[M:4/4] AFgG, ad'be' | [M:7/16] c''/a/e/b/ C/B/A/ | [M:3/8] E,3 |
%%postscript /hd/hdo load bind def     % restaura 'hd' original
```



− − A FAZER: − −

− ORDEM DE EXECUÇÃO DAS DECORAÇÕES:
(!head−x! e [!deco!c] por voz, !deco! por tempo)
− CABEÇAS DE NOTAS PARA PERCUSSÃO
− DECORAÇÕES !xxx(! E !xxx)! GENÉRICAS
− DECORAÇÕES EM HASTES
− %%postscript /Helvetica 20 selectfont count 5 string cvs 0 −40 M show
− %%beginps , %%endps E [I:postscript]
− SOBRE OS ARQUIVOS syms.c (operadores PS) E deco.c (parâmetros)
− SOBRE ARQUIVOS DE FORMATO