Ircam
documentation

- Research reports
- Musical works
- Software

# PatchWork

## Utils

*première édition, mars 1997*

IRCAM Centre Georges Pompidou

Ce manuel a été réalisé par Laurent Pottier, et produit sous la responsabilité éditoriale de Marc Battier, département de la Valorisation, Ircam.

PatchWork a été conçu et programmé par Mikael Laurson, Camilo Rueda et Jacques Duthen.

La librairie Utils a été écrite par Laurent Pottier.

Cette documentation correspond à la version 1.0 de la librairie et à la version 2.0 ou ultérieure de PatchWork.

Première édition de la documentation, mars 1997.



**Ircam**
**1, place Igor-Stravinsky**
**F-75004 Paris**
**Tel. 01 44 78 49 62**
**Fax 01 44 78 15 40**
**E-mail ircam-doc@ircam.fr**

# Groupe d'utilisateurs IRCAM

L'utilisation de ce programme et de sa documentation est strictement réservée aux membres des groupes d'utilisateurs de logiciels Ircam. Pour tout renseignement supplémentaire, contactez :


Département de la Valorisation
Ircam
1, Place Stravinsky
F-75004 Paris
France

Tél. 01 44 78 49 62
Fax 01 44 78 15 40
Courrier électronique: bousac@ircam.fr


Veuillez faire parvenir tout commentaire ou suggestion à :


M. Battier
Département de la Valorisation
Ircam
1, Place Stravinsky
F-75004 Paris
France
Courrier électronique: bam@ircam.fr

http://www.ircam.fr/forumnet

# Contenu

To see the table of contents of this manual, click on the Bookmark Button located in the Viewing section of the Adobe Acrobat Reader toolbar.

# 1 Introduction

La librairie "Utils" est une librairie de PatchWork qui regroupe quelques fonctions utilitaires simples.

Elle comprend une liste de séries mathématiques usuelles, une fonction de réduction de BPF ainsi qu'une fonction pour modifier l'affichage des BPF. Quelques fonctions permettent l'écriture de données PatchWork sur disque ou l'accès au nom de documents stockés sur disque. Enfin, une fonction permet d'éviter des évaluations multiples au sein d'un fragment de patch.

Les fonctions sont organisées comme indiqué dans le menu suivant:

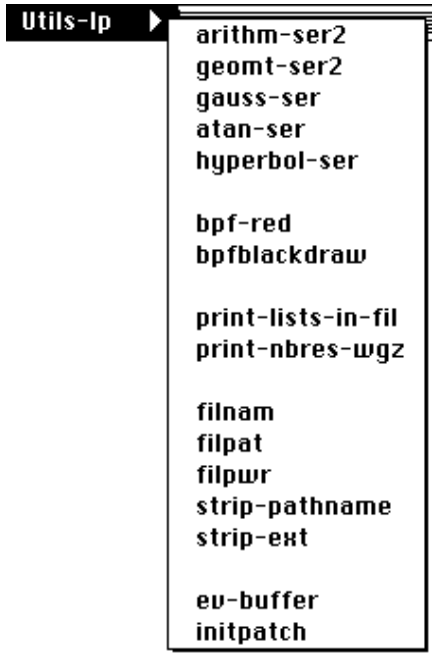



Figure 1 Liste des menus de la librairie "Utils"

## 1.1 Exemples

Un premier patch permet de visualiser le type de progressions engendrées par les différentes séries proposées. Les fonctions **gauss-ser** et **hyperbol-ser** produisant des valeurs comprises entre 0 et 1 et la fonction

**atan-ser** des valeurs entre -1 et 1, ces valeurs doivent être multipliées par 100 pour être affichées dans une BPF.

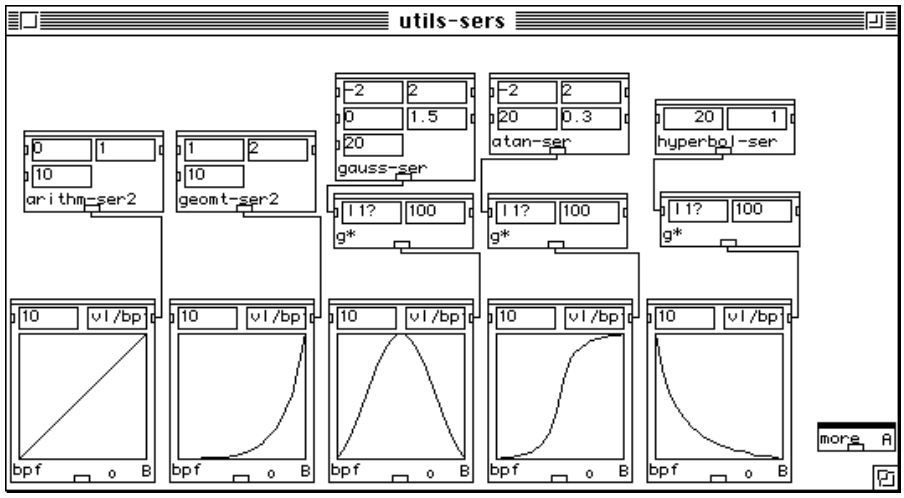



Figure 2 — Principales séries numériques de la librairie "Utils"

Le patch suivant donne des exemples de valeurs pour les coefficients des séries de gauss (écart-type ou dispersion), des séries tangentielles (écart ou facteur d'échelle) et des séries hyperboliques (exposant ou degré).

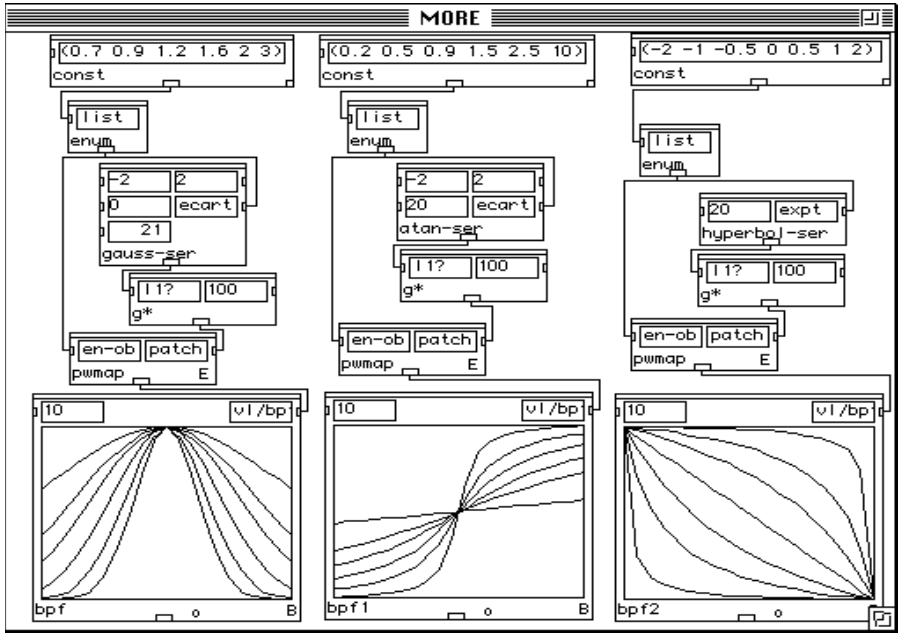



Figure 3 — Exemples de valeurs numériques pour la construction des séries de la librairie "Utils"

La fonction **bpf-red** permet de faire une réduction du nombre de points d'une BPF en supprimant les points qui créent les ruptures de pentes les plus faibles. Cette technique est d'autant plus efficace que les coordonnées horizontales (x-points) des points sont espacés régulièrement. Elle permet selon le cas de réduire en moyenne de 50% le nombre de points d'une BPF sans perte notable d'information.

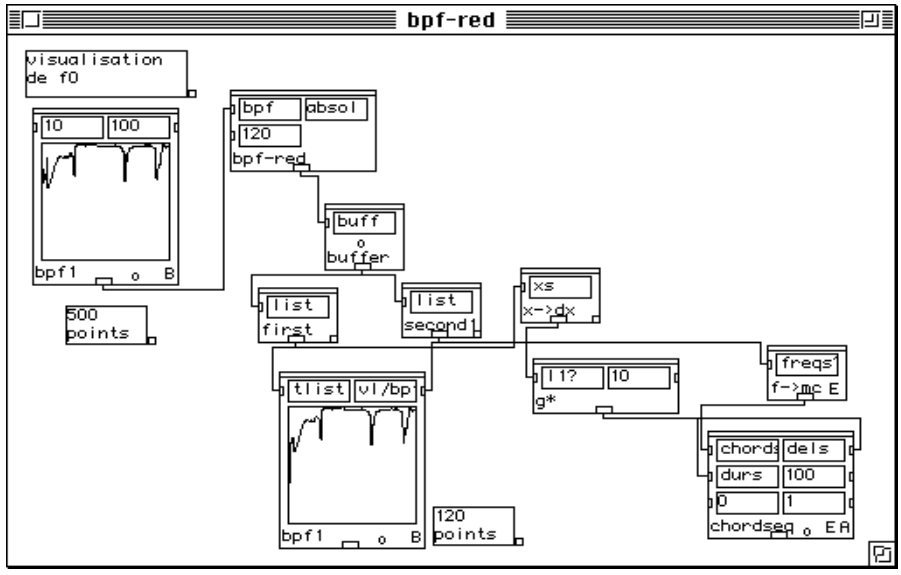



Figure 4 Réduction du nombre de points contenus dans une BPF (de 500 à 120 points)

Les fonctions **print-lists-in-fil** et **print-nbres-wgz** permettent d'écrire des données au format "texte" de façon à les rendre disponibles pour d'autres types de programmes (Audiosculpt, Csound, tableur, etc.).

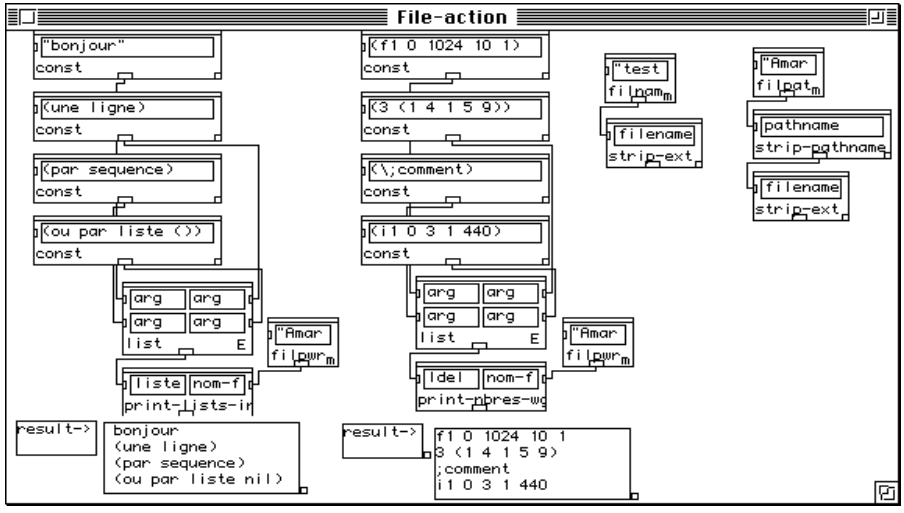



Figure 5 Ecriture de listes de données dans un fichier texte

La fonction **ev-buffer** associée à la fonction **initpatch** permet d'éviter des évaluations multiples d'un module dont la sortie est dirigée vers plusieurs modules. Lors de l'utilisation de fonctions aléatoires, il est souvent nécessaire de pouvoir récupérer une valeur aléatoire en plusieurs exemplaires. Il suffit alors de placer le module **ev-buffer** en sortie du module aléatoire, de connecter la sortie de **ev-buffer** vers les différentes destinations, puis en bas de patch de placer un module **initpatch** pour délimiter le fragment de patch concerné par l'évaluation unique.

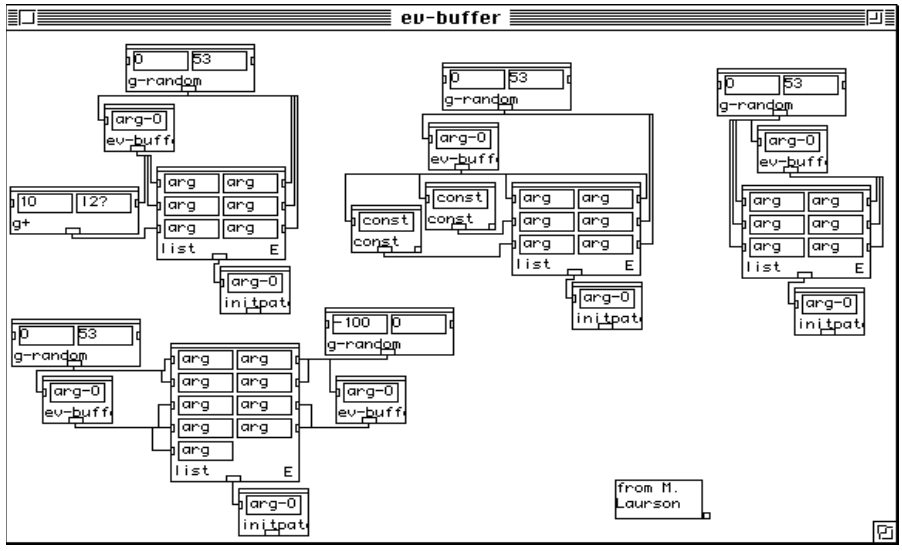
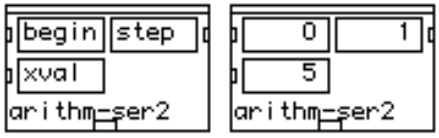



Figure 6 — Le module "ev-buff" permet d'évaluer une fois seulement une fonction partant dans plusieurs directions

# 2 Référence

## 2.1 arithm-ser2




**Syntaxe**

(arithm-ser2 *begin step xval*)

**Entrées**

| | |
|---|---|
| *begin* | nombre entier ou flottant |
| *step* | nombre entier ou flottant |
| *xval* | nombre entier ou flottant |

**Sortie**

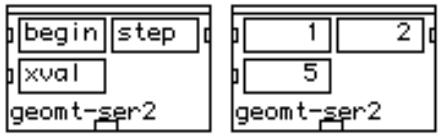
liste

Produit une série arithmétique partant de la valeur *begin*, de pas *step* et comportant *xval* valeurs.

**Exemple**

```
;(arithm-ser2 1 5 10)
-->(1 6 11 16 21 26 31 36 41 46)
```

## 2.2 geomt-ser2




**Syntaxe**

(geomt-ser2 *begin step xval*)

**Entrées**

| | |
|---|---|
| *begin* | nombre entier ou flottant |
| *step* | nombre entier ou flottant |
| *xval* | nombre entier ou flottant |

**Sortie**

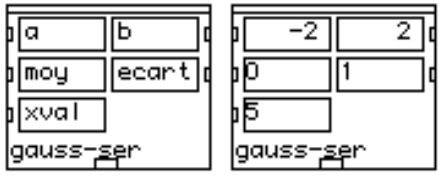
liste

Produit une série géométrique partant de la valeur *begin*, de pas *step* et comportant *xval* valeurs.

**Exemple**

```
;(geomt-ser2 1 2 10)
-->(1 2 4 8 16 32 64 128 256 512)
```

## 2.3 gauss-ser




**Syntaxe**

(gauss-ser *a b moy ecart xval*)

**Entrées**

| | |
|---|---|
| *a* | nombre entier ou flottant |
| *b* | nombre entier ou flottant |
| *moy* | nombre entier ou flottant |
| *ecart* | nombre entier ou flottant |
| *xval* | nombre entier ou flottant |

**Sortie**

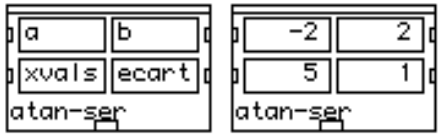
liste

Donne les images par une courbe de gauss (fonction normale de moyenne *moy* et d'écart-type *ecart*) de *xval* valeurs prises entre *a* et *b*

**Exemple**

```
;(gauss-ser 0 2 1 1 5 )
--> (0.37 0.78 1.0 0.78 0.37)
```

# 2.4 atan-ser




**Syntaxe**

(atan-ser *a b xvals ecart*)

**Entrées**

| | |
|---|---|
| *a* | nombre entier ou flottant |
| *b* | nombre entier ou flottant |
| *xvals* | nombre entier ou flottant |
| *ecart* | nombre entier ou flottant non nul |

**Sortie**

liste

Donne la transformée de la suite algébrique allant de *a* à *b* en *xvals* valeurs par la fonction **f** telle que :

f(x) = arctangente(x/ecart)

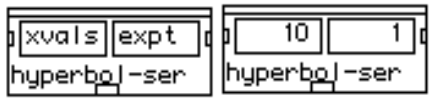
si ecart < 1 => forte concentration sur 0

si ecart > 1 => forte dispersion

**Exemple**

```
;(atan-ser -2 2 7 1)
-->(-0.7 -0.59 -0.37 0.0 0.37 0.59 0.7)
```

# 2.5 hyperbol-ser




**Syntaxe**

(hyperbol-ser *xvals expt*)

**Entrées**

| | |
|---|---|
| *xvals* | nombre entier ou flottant |
| *expt* | nombre entier ou flottant non nul |

**Sortie**

liste

Crée une série de *xvals* valeurs correspondant à une progression hyperbolique de degré *expt*, décroissante entre 0 et 1.

si n = 0 : fonction linéaire y = 1 - x

si n > 0 : rapide décroissance au début, lente à la fin

si n < 0 : décroissance lente au début, rapide à la fin

**Exemple**

```
;(hyperbol-ser 7 1)
-->(1.0 0.4 0.2 0.119 0.063 0.026 0.0)
```

## 2.6 bpf-red


**Syntaxe**

(bpf-red *bpf option red*)

**Entrées**

| | |
|---|---|
| *bpf* | objet BPF |
| *option* | menu (deux items : "relatif" et "absolu" |
| *red* | nombre entier ou flottant strictement positif |

**Sortie**

liste

Donne une liste pour x et une liste pour y correspondant à une version de *BPF* avec un nombre de points réduit.

Suivant la valeur de l'entrée *option* il s'agit d'une réduction relative (en % du nombre de points, *red* = 0.5 = 50 %) ou d'une réduction absolue (*red* donne le nombre de points de sortie).

## 2.7 bpfblackdraw


**Syntaxe**

(bpfblackdraw *color*)

**Entrées**

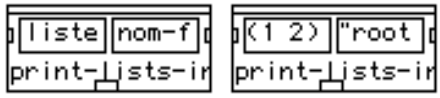
*color* menu (deux items : "black" et "gray")

**Sortie**

()

Permet de changer le trait des courbes secondaires pour une multi-BPF. Par défaut, ces traits sont gris. Lorsque la librairie "Utils" est chargée, ils passent en noir.

## 2.8 print-lists-in-fil




**Syntaxe**

(print-lists-in-fil *liste nom-fich*)

**Entrées**

| | |
|---|---|
| *liste* | liste |
| *nom-fich* | séquence de caractères |

**Sortie**

()

Ecrit le texte contenu dans *liste*, une ligne par liste ou par séquence, dans le fichier *nom-fich*.

**Exemple**

```
;(print-lists-in-fil
'("bonjour" (une ligne) (par sequence) (ou par liste ()))
-->
```

```
bonjour
(une ligne)
(par sequence)
(ou par liste nil)
```

# 2.9 print-nbres-wgz


**Syntaxe**

(print-nbres-wgz *ldel nom-fich*)

**Entrées**

| | |
|---|---|
| *ldel* | liste |
| *nom-fich* | chaîne de caractères |

**Sortie**

()

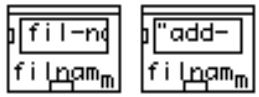
Ecrit le texte contenu dans *ldel*, une ligne par liste, chaque mot séparé par une tabulation, dans le fichier *nom-fich*. Cette fonction permet par exemple d'envoyer des données vers un tableur, vers CSound ou Audiosculpt.

**Exemple**

```
;(print-nbres-wgz
'((f1 0 1024 10 1) (3 (1 4)) (\;COMMENT) (i1 0 3 1 440))
-->
```

```
f1 0 1024 10 1
3 (1 4)
;comment
i1 0 3 1 440
```

# 2.10 filnam




**Syntaxe**

(**filnam** *fil-name*)

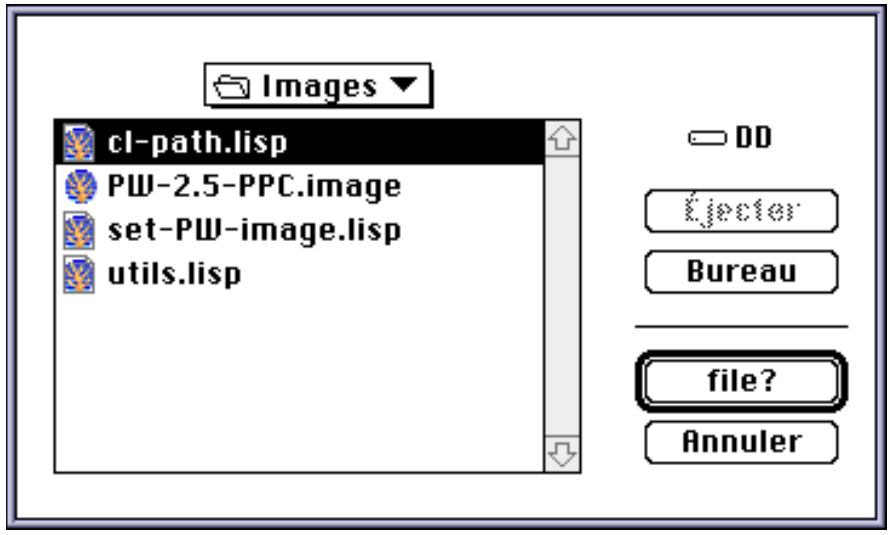
**Entrées**

*file-name* chaine de caractères

**Sortie**

chaine de caractères

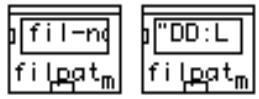
Donne le nom du fichier sélectionné en utilisant le mini-menu *m* placé en bas à droite du module. Si l'entrée est nil, le module ouvre automatiquement une fenêtre de dialogue.




**Exemple**

```
--> "cl-path.lisp"
```

# 2.11 filpat




**Syntaxe**

(filpat *file-name*)

**Entrées**

*file-name* chaine de caractères

**Sortie**

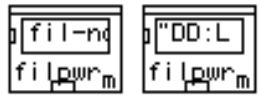
chaine de caractères

Donne en résultat le nom et le chemin d'accès du fichier sélectionné en utilisant le mini-menu *m* placé en bas à droite du module. Si l'entrée est nil, le module ouvre automatiquement une fenêtre de dialogue.

**Exemple**

```
--> "DD:Languages:PW 2.5.2 PPC:Images:cl-path.lisp"
```

## 2.12 filpwr




**Syntaxe**

(**filpwr** *file-name)*

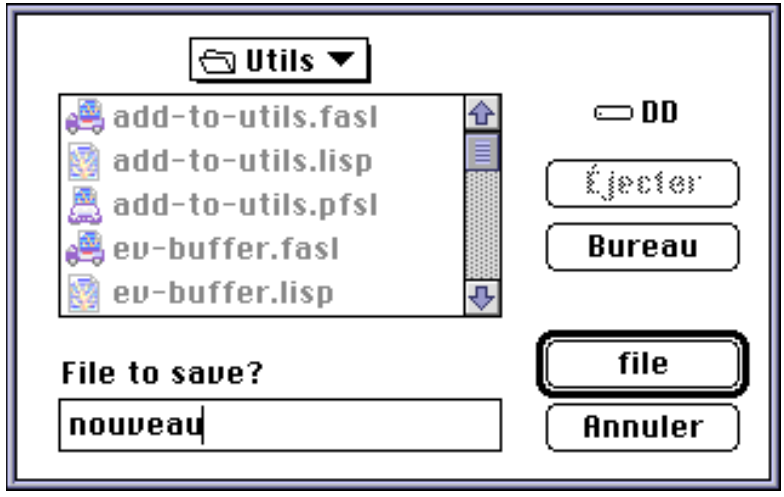
**Entrées**

*file-name* chaine de caractères

**Sortie**

chaîne de caractères

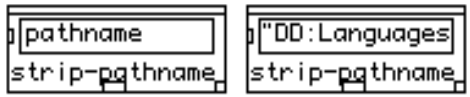
Donne en résultat le nom et le chemin d'accès du fichier indiqué en utilisant le mini-menu *m* placé en bas à droite du module et en tapant le nom de ce fichier. Si l'entrée est nil, le module ouvre automatiquement une fenêtre de dialogue.




**Exemple**

```
--> "DD:Languages:PW 2.5.2 PPC:User-library:Utils:nouveau"
```

## 2.13 strip-pathname




**Syntaxe**

(strip-pathname *pathname*)

**Entrées**

*pathname* chaîne de caractères

**Sortie**

chaîne de caractères

Donne le nom d'un fichier à partir d'une chaîne de caractères indiquant sa localisation.

**Exemple**

```
;(strip-pathname "DD:Languages:PW 2.5.2:Images:cl-path.lisp")
-->"cl-path.lisp"
```

## 2.14 strip-ext


**Syntaxe**

(**strip-ext** *filename*)

**Entrées**

*filename* chaîne de caractères

**Sortie**

chaîne de caractères

Supprime l'extension du nom du fichier fourni en entrée

**Exemple**

```
;(strip-ext "cl-path.lisp")
--> "cl-path"
```

## 2.15 ev-buffer


**Syntaxe**

(ev-buffer *val*)

**Entrées**

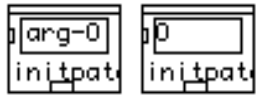
*val* toute entrée

**Sortie**

()

Ce module envoie la même valeur à tous les modules auquels est connectée sa sortie.

## 2.16 initpatch




**Syntaxe**

(**initpatch** *val*)

**Entrées**

*val* toute entrée

**Sortie**

()

Envoie vers le patch situé en amont une méthode d'initialisation destinée aux modules **ev-buffer**.

# Index