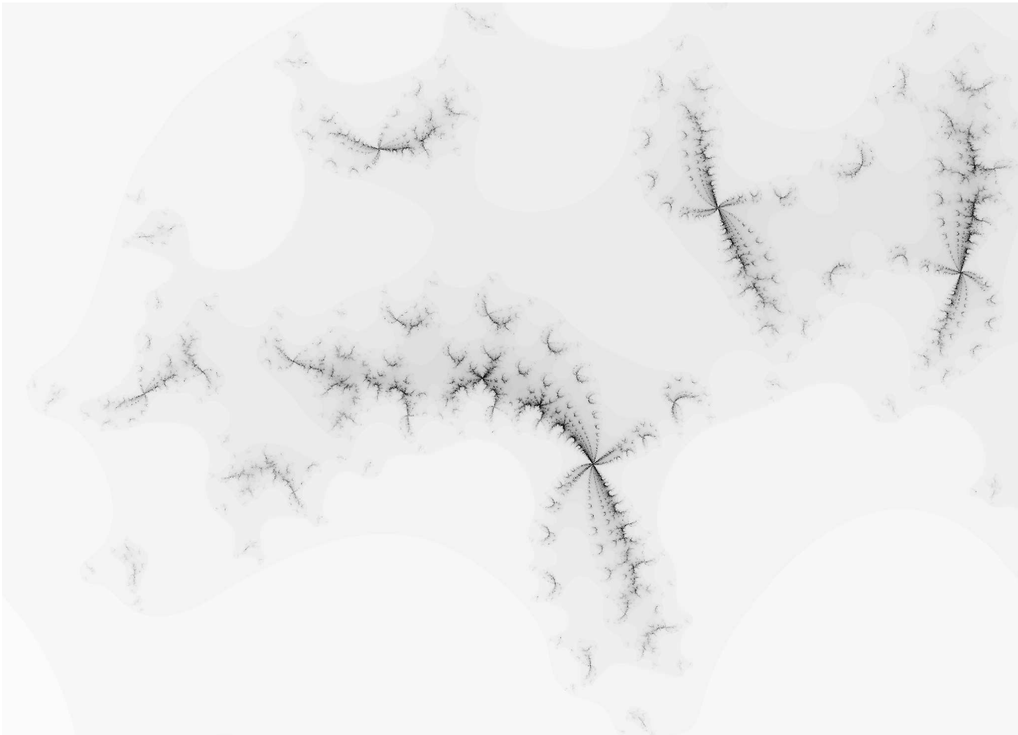
Unix

Petit manuel de référence



Laurent Mazet mazet@softndesign.org

Table des matières

Chapitre 1

Avant propos

1.1 Introduction

Le système UNIX fut conçu au début des années 70 par AT&T (actuellement USL). Du fait de la variété de machines sur lesquelles il tournait, il s'est enrichi de commandes crées par des générations successives de développeurs. Sous l'impulsion de différents organismes (comme USL ou l'université californienne Berkeley), plusieurs standards UNIX ont ainsi vu le jour et ont évolué en parallèle. Actuellement, la normalisation POSIX a apporté une homogénéisation des principaux UNIX (System V et BSD) ainsi, quelque soit leur marque, il existe un ensemble de commandes communes à tous les UNIX actuellement distribués.

1.2 Manuel de référence

Ce manuel présente les commandes qui ont le plus d'intérêt pour les utilisateurs de systèmes Unix sous forme de « manuel de références ».
Chaque entrée est indexée par le nom de la commande. La syntaxe est suivie d'une brève description, de la liste des principales options disponibles et, souvent, d'exemples d'utilisation.

1.3 Conventions

Les conventions typographiques sont les suivantes :

Gras	est utilisé pour les répertoires, les fichiers, les commandes ainsi que les options. Ce sont les mots clés qu'il faut utiliser tel quel (sans changer un seul caractère).
Italique	est utilisé pour les expressions génériques d'arguments et d'options.
Gras italique	est utilisé pour les titres.
`Courrier`	est utilisé pour les exemples.
`$`	présente le prompt du shell.
[.]	encadre les listes de paramètres optionnels.
EOF	indique le caractère « fin de fichier ». Classiquement CONTROL-D
|	sépare les paramètres dont un seul à la fois est nécessaire.

Chapitre 2

Commandes UNIX

ar	**ar** [**-V**] [-]*clé*[*options*] [*fichpos*] *archive* [*fichiers*] Gère un ensemble de *fichiers* associés dans un fichier *archive*. Ceci est le plus souvent utilisé pour créer et mettre à jour des bibliothèques utilisées par l'éditeur de liens (**ld**). Seul une lettre clé peut être utilisée mais elles peuvent être associées à un nombre quelconque d'*options*. *fichpos* est le nom d'un fichier inclus dans l'*archive*. Lors du déplacement ou du remplacement de *fichiers*, il est possible de préciser qu'ils soient placés avant ou après *fichpos*. **-V** affiche le numéro de version de **ar** sur le canal d'erreur. ***Clé*** **d** — Efface les *fichiers* de l'*archive*. **m** — Déplace les *fichiers* en fin d'*archive*. **p** — Affiche les *fichiers* de l'*archive*. **q** — Ajoute les *fichiers* dans l'*archive*. **r** — Remplace les *fichiers* dans l'*archive*. **t** — Liste le contenu de l'*archive* ou simplement les *fichiers* nommés. **q** — Extrait les fichiers contenu de l'*archive* ou simplement les *fichiers* nommés. ***Options*** **a** — S'utilise avec **r** et **m** pour placer les *fichiers* après *fichpos*. **b** — S'utilise avec **r** et **m** pour placer les *fichiers* avant *fichpos*. **c** — Crée une *archive* sans demander de confirmation. **i** — Identique à **b**. **s** — Impose la remise à jour de la table des symboles de l'*archive* (utile après avoir lancé **strip**). **u** — S'utilise avec **r** pour ne remplacer que les *fichiers* ayant changés après avoir été inclus dans l'*archive*. **v** — Affiche une description sommaire de la commande. ***Exemple*** Changement des objets du répertoire courant dans l'archive **mabiblio.a** : `ar r mabiblio.a 'ls *.o'`

at	**at** *options1 heure* [*date*] [+*incrément*] **at** *options1* [*tâches*] Exécute des commandes entrées sur le canal d'entrée standard à une *heure* précise et, éventuellement, à une certaine *date*. La fin de l'entrée est marquée par *EOF*. L'*heure* peut être écrite sous forme numérique (avec accessoirement les minutes et les codes horaires) ou à l'aide d'un mot clé. La *date* peut être de la forme mois/jour ou un jour de la semaine ou un mot clé spécial. L'*incrément* est un entier positif suivi d'un mot clé. ***Option1*** **-f** *fichier* — les commandes contenues dans le *fichier*. **-m** — Envoie un mél à l'utilisateur une fois la tâche effectuée. ***Option2*** **-l** — Affiche la liste de toutes les tâches (ou seulement les *tâches* précisées) par l'utilisateur. Voir aussi **atq**. **-r** — Retire les *tâches* précédemment programmées. Vous devez avoir des droits particuliers ou être propriétaire de la tâche. Utilisez **-l** d'abord pour connaître la liste des tâches programmées. Voir aussi **atrm**. ***Heure*** *hh* :*mm* [*codes*] L'heure peut être un nombre à un ou deux chiffres (par défaut, l'horloge est supposée de 24 heures) ; les minutes optionnelles peuvent être écrites sur un ou deux chiffres ; les deux points peuvent être omis : le format est *h*, *hh* ou *hhmm* c'est-à-dire que 5, 5 :30, 0530 et 1945 sont des heures valides. Si les *codes* horaires **am** ou **pm** sont ajoutés, l'horloge est alors de 12 heures. Le code **zulu** indique une référence au méridien de Greenwich. **midnight** | **noon** | **now** Ces mots clés peuvent être utilisés en remplacement d'une heure numérique. **now** doit être suivi d'*incrément*. ***Date*** *mois nombre*[, *année*] Le *mois* (en anglais) doit être écrit en toute lettre ou simplement à l'aide de son abréviation en trois lettres ; *nombre* représente un jour du mois en chiffre ; l'*année* s'écrit quand à elle à l'aide d'un nombre à quatre chiffres. Si le *mois* précisé dans la ligne de commande est antérieur au mois en cours, *at* comprend automatiquement que cela se réfère à l'année prochaine. *jour* Le *jour* (en anglais) doit être écrit en toute lettre ou simplement à l'aide de son abréviation en trois lettres. **now** | **tomorrow** Référence le jour en cours ou le suivant. Si la *date* n'est pas précisée, **at** programme la tâche au jour en cours si l'*heure* n'est pas déjà passée, sinon elle est programmée pour le lendemain.

	Incrément
	Ajoute un incrément numérique si vous voulez programmer le jour ou l'heure d'exécution d'une tâche *relativement* à la date actuelle. Le nombre doit être suivi d'un des mots clé suivant : **minute**, **hour** (heure), **day** (jour), **week** (semaine), **month** (mois) ou **year** (année) (ou leurs formes plurielles). Le mot clé qui peut être utilisé, est un équivalent de **+1**.
	
	Exemples
	Remarquez que les deux premiers exemples sont identiques.
	`at 1305 August 3`
	`at 1:05pm Aug 3`
	`at 11 am Friday`
	`at now + 7 hours`
	`at noon next month`

atq	**atq** [*options*] [*utilisateurs*] Affiche les tâches crées par la commande **at** qui sont encore en attente. Habituellement les tâches sont affichées dans l'ordre où elles vont être exécutées. Si aucun *utilisateur* n'est précisé, ou que vous n'avez pas de droit particulier, seules vos tâches sont listées. ***Options*** **-c** Classe les tâches par ordre de programmation via la commande **at**. **-o** Affiche uniquement le nombre de tâches en attente.
atrm	**atrm** [*options*] [*utilisateurs* | *idTâches*] Enlève les tâches programmées par **at** correspondantes aux identificateurs « *idTâches* ». Si vous avez des droits particuliers, vous pouvez aussi préciser le nom de l'*utilisateur* qui a programmé des tâches. ***Options*** **-a** Enlève toutes les tâches programmées appartenant à un *utilisateur* courant. **-f** Enlève les tâches sans envoyer d'information d'effacement. **-i** Demande une confirmation (**y** ou **n**).
banner	**banner** *phrase* Affiche sous forme de poster une *phrase* sur le canal de sortie standard. Tous les mots doivent contenir moins de 10 caractères.
c++	**c++** [*options*] *fichiers* Compilateur C++. Voir **g++**

cal

cal [[*mois*] *année*]

Sans argument, **cal** affiche le calendrier du mois en cours. Sinon elle affiche les douze mois de l'*année* précisée (par un nombre de 1 à 9999) ou le *mois* de l'*année* précisé (par un nombre de 1 à 12).

Exemples

```
cal 8 1973
cal 1998 > fichier_annee
```

---

cat

cat [*options*] [*fichiers*]

Lit un ou plusieurs *fichiers* et les affiche sur le canal de sortie standard. **cat** lit les données provenant du canal d'entrée standard si aucun fichier n'est précisé ou si « **-** » fait partie de la liste des *fichiers* ; la fin est marquée par *EOF*. Vous pouvez utiliser l'opérateur « $>$ » pour concaténer plusieurs *fichiers* en un nouveau fichier. De même, l'opérateur « $> \ >$ » vous permet d'ajouter des *fichiers* en fin d'un fichier existant.

Options

-A	Equivalent à **-vTE**.
-b	Numérote les lignes non vides.
-E	Affiche un $en fin de chaque ligne.
-e	Equivalent à **-vE**.
-n	Numérote toutes les lignes.
-T	Affiche un ˆI pour chaque tabulation.
-t	Equivalent à **-vT**.
-v	Affiche tous les caractères « invisibles » via les notations ˆ et M- sauf pour les retours à la ligne et les tabulation.

Exemples

`cat fich1`	Affiche le contenu du fichier fich1 sur la sortie standard.
`cat f1 f2 > f3`	Concatène les fichiers f1 et f2 en le fichier f3.
`cat f4 >> f5`	Ajoute le fichier f4 a la fin du fichier f5.
`cat > f6 << STOP`	Crée un fichier à partir du terminal jusqu'à `STOP`.

---

chmod

chmod [*options*] *mode fichiers*

Change les droits d'accès (suivant un certain *mode*) de un ou plusieurs *fichiers*. Seul le propriétaire des *fichiers* ou un utilisateur ayant des droits particuliers peut changer le *mode* d'accès. Le format du *mode* d'accès se décompose en trois parties : *Niveau(x)OpérationDroit(s)*. Si aucun *niveau* n'est précisé, le changement est effectué à tous les niveaux. Une seul *opération* n'est possible à la fois.

Options

-c	N'affiche que les opérations au cours desquelles les droits des fichiers ont changés.
-f	N'affiche aucun message d'erreur.
-R	Effectue le changement de droit de façon récursive. C'est-à-dire que si la liste des *fichiers* contient des répertoires, leurs contenus voient aussi leurs droits changer.
-v	Affiche les opérations effectuées fichier par fichier.

Niveaux

u	Propriétaire
g	Groupe
o	Autre
a	Tout niveau (c'est-à-dire **ugo**)

Opération

+	Ajoute des droits.
-	Supprime des droits.
=	Fixe des droits en enlevant les permission non spécifiées.

Droits

r	Droits de lecture
w	Droits d'écriture
x	Droits d'exécution
X	Droits d'exécution pour des fichiers exécutables ou des répertoires (propagation du **x** aux niveaux qui ne l'ont pas encore)
s	Récupération des droits du propriétaire (ou du groupe) lors de l'exécution.
t	« Sticky bit »
u	Droits actuels du propriétaire
g	Droits actuels du groupe
o	Droits actuels des autres

Il est aussi possible d'écrire le *mode* sous forme d'un nombre octal de 3 chiffres. Le premier désigne les droits du propriétaire, le deuxième ceux du groupe et le troisième ceux des autres. Ces chiffres sont calculés par addition de codes fonction des droits (4 $\leftrightarrow$ lecture, 2 $\leftrightarrow$ écriture et 1 $\leftrightarrow$ exécution). Un quatrième chiffre peut précéder cette séquence pour compléter des droits exceptionnels (4 $\leftrightarrow$ **u=s**, 2 $\leftrightarrow$ **g=s** et 1 $\leftrightarrow$ **=t**).

	Exemples
	Ajoute les droits d'exécution au propriétaire et au groupe à **prog** :
	`chmod ug+x prog`
	Les deux opérations suivantes sont équivalentes :
	`chmod 751 prog`
	`chmod u=rwx,g=rx,o=x prog`
	Elles fixent les droits de l'utilisateur en lecture, écriture et exécution, les droits du groupe en lecture et écriture et les droits des autres uniquement en exécution.

cc	**cc** [*options*] *fichiers* Compilateur C. Voir **gcc**
chfn	**chfn** [*options*] [*utilisateur*] Permet de changer les informations utilisées par **finger**. Seul l'administrateur peut changer les informations d'un autre utilisateur que lui même. Si aucune option n'est précisée, **chfn** demande champ par champ à entrer les modifications. ***Options*** **-f** *noms* Change le nom et le prénom. **-o** *bureau* Change le nom du bureau. **-p** *btel* Change le numéro de téléphone du bureau. **-h** *mtel* Change le numéro de téléphone personnel.
chsh	**chsh** [*options*] [*utilisateur*] Permet de changer de shell. Seul l'administrateur peut changer le shell d'un autre utilisateur que lui même. Si aucune option n'est précisée, **chsh** demande quel est le nouveau shell. ***Options*** **-s** *shell* Précise le nouveau *shell*. **-l** Affiche la liste des shells disponibles (fichier **/etc/shells**).
compress	**compress** [*options*] [*fichiers*] Réduit la taille d'un ou plusieurs *fichiers* en utilisant l'algorithme de compression Lempel-Ziv et les renomme en *fichier***.Z**. **uncompress** et **zcat** permettent de revenir aux anciens fichiers. Ce compresseur est moins efficace que **gzip**. ***Options*** **-c** Ecrit sur le canal standard de sortie (sans changer les *fichiers*). **-f** Force l'opération sans demander de confirmation. **-r** Décompresse le contenu de répertoires. **-v** Affiche le taux de compression des *fichiers*.

cp	**cp** [*options*] *fichier1 fichier2* **cp** [*options*] *fichiers répertoire* Copie *fichier1* en *fichier2* ou copie un ou plusieurs *fichiers* sous le même nom dans un autre *répertoire*. Si la destination est un fichier qui existe déjà, le fichier est écrasé. ***Options*** **-i** Demande une confirmation avant d'écraser un fichier existant. **-p** Garde la date de dernière modification et les droits lors de la copie. **-r** Effectue une copie récursive (par un parcours des sous-répertoires) dans un autre *répertoire*. ***Exemple*** Copie deux fichiers dans leur répertoire père. `cp unix.txt comp.c ..`
diff	**diff** *fichier1 fichier2* Compare deux fichiers et indique les différences. Les lignes provenant de *fichier1* sont préfixées par $>$ et celles provenant de *fichier2* par $<$. Des lignes de contexte indiquent les blocs de lignes qui ont été modifiés (**c**), ajoutés (**a**) ou détruits (**d**) ; le principe étant d'indiquer les modifications à effectuer pour convertir *fichier1* en *fichier2*.
f77	**f77** [*options*] *fichiers* Compilateur Fortran 77. Voir **g77**
find	**find** *répertoires conditions operations* Recherche (à partir d'un ou plusieurs *répertoires*) en fonction d'une ou plusieurs *conditions* des noms de fichiers et y effectue des opérations. Cette commande est extrêmement puissance. Il ne faut pas oublier de préciser au minimum une *condition* et un *répertoire*. Le point d'exclamation (« **!** »permet de nier une condition. ***Conditions*** **-group** *g* Recherche tout fichier appartenant au groupe *g*. **-name** *m* Recherche tout nom correspondant à *m*. **-perm** [-]*ooo* Recherche tout fichier dont les droits d'accès correspondent exactement à *ooo* (code octal). Avec Le « **-** », **find** recherche tout fichier dont les droits correspondent au moins à *ooo*. **-type** *t* Recherche tout fichier de type *t* (**b** $\leftrightarrow$ périf. bloc, **c** $\leftrightarrow$ périf. caractère, **d** $\leftrightarrow$ répertoire, **l** $\leftrightarrow$ lien symbolique ou **f** $\leftrightarrow$ fichier normal). **-size** *s* Recherche tout fichier dont la taille est de *s*. **-user** *u* Recherche tout fichier appartenant à l'utilisateur *u*.

Commandes

-exec *c* **{}\;**	Exécute une commande *c* sur chacun des noms trouvés. **{}** est remplacé par le nom trouvé.
-follow	Suit les liens symboliques.
-ok *c* **{}\;**	Demande une confirmation avant d'exécuter la commande *c*.
-print	Affiche chaque nom trouvé.

Exemples

Recherche dans **/trash** de tous les fichiers appartenant à **bill**.

```
find /trash -user bill -print
```

Compression de tous les fichiers non déjà compressés (depuis le répertoire courant).

```
find ./ ! -name '*.gz' -exec gzip {} \;
```

Efface les fichiers vides appartenant à **porte** dans son répertoire principal.

```
find ~porte/ -size 0 -exe rm -f {} \;
```

finger

finger [*options*] *utilisateurs*

Affiche des informations concernant un ou plusieurs *utilisateurs* ainsi que les fichiers **.plan**, **.project** et **.forward** contenus dans les racines des *utilisateurs*. Vous pouvez préciser le nom de login (exact), le prénom ou le nom de famille (**finger** renverra tous les utilisateurs dont l'un des noms correspond). Il est possible d'interroger une machine distante avec la syntaxe *utilisateur@machine*.

Options

-l	Permet d'utiliser le format long (par défaut).
-m	Les nom donnés ne doivent correspondre qu'à des noms de login.
-p	N'affiche pas les fichiers **.plan** et **.project**.
-s	Permet d'utiliser le format court.

ftp

ftp *machine*

Permet de communiquer avec une *machine* distante en utilisant le protocole FTP (File Transfer Protocol) et ainsi de transférer des données entre deux machines via le réseau. Les principales commandes de **ftp** sont :

	Commandes **acsii** Indique que le transfert des fichiers s'effectue sur 7 bits (fichiers textes). **binary** Indique que le transfert des fichiers s'effectue sur 8 bits (tout autre que les fichiers textes). **cd** Change de répertoire sur la machine distante. **close** Ferme une session FTP. **delete** Efface un fichier sur la machine distante. **get** Copie un fichier de la machine distante vers la machine locale. **hash** Affiche des dièses « # » pour chaque kilo-octet transféré. **help** Affiche une aide sommaire pour une commande donnée. Si aucune commande n'est indiquée, **help** affiche la liste des commandes. **ls** Affiche la liste des fichiers sur la machine distante. **lcd** Change de répertoire sur la machine locale. **mdelete** Efface plusieurs fichiers sur la machine distante. **mget** Copie plusieurs fichiers de la machine distante vers la machine locale. **mkdir** Crée un répertoire sur la machine distante. **mput** Transfère plusieurs fichiers de la machine locale vers la machine distante. **open** Ouvre une session FTP. **prompt** Si **prompt** est **on**, une confirmation sera demandée lors de toute opération sur un ensemble de fichiers. **put** Transfère un fichier de la machine locale vers la machine distante. **quit** Quitte le logiciel FTP. **rmdir** Efface un répertoire sur la machine distante.
g++	**g++** [*options*] *fichiers* Compile un ou plusieurs *fichiers* sources C++. **g++** fonctionne exactement de la même façon que **gcc** ; le détail des options est donc le même (voir **gcc**).
g77	**g77** [*options*] *fichiers* Compile un ou plusieurs *fichiers* sources Fortan 77. **g77** fonctionne exactement de la même façon que **gcc** ; le détail des options est donc le même (voir **gcc**).
gcc	**gcc** [*options*] *fichiers* Compile un ou plusieurs *fichiers* sources C. **gcc** appelle automatiquement l'éditeur de liens **ld** (tant que l'option **-c** n'est pas utilisée). Lorsque **gcc** doit générer un fichier objet, si aucun nom n'est précisé, ce dernier prend le nom sans extension du fichier C avec comme suffixe **.o**. Par défaut, tout exécutable crée est nommé **a.out**.

Options

-c	Compile sans faire l'édition de liens (génération des fichier objet **.o**.
-D*ct*[**=***v*]	Définit une constante *ct* valant *v* pour le pré-processeur (par défaut *ct* prend la valeur 1).
-g	Effectue la compilation avec des symboles de débogage utilisés par **gdb**.
-I*rep*	Ajoute le répertoire *rep* dans la liste des répertoires explorés lors de la rechercher des fichiers à inclure (**#include** <*fichier***.h**>).
-L*rep*	Ajoute le répertoire *rep* dans la liste des répertoires explorés lors de la recherche des bibliothèques pour l'édition de liens.
-l*truc*	Ajoute la bibliothèque **lib***truc***.a** dans la liste des bibliothèques pour l'édition de liens.
-O*num*	Effectue des optimisations plus ou moins performantes, fonction de *num* (de 1 à 3).
-o *fichier*	Donne un *nom* au fichier généré (objet ou exécutable).
-W	Affiche les messages de mise en garde les plus flagrants.
-Wall	Affiche tous les messages de mise en garde.

Exemple

Compilation puis édition de liens de **prog.c**.

```
gcc -Wall -c prog.c
gcc -o prog prog.o
```

gdb

gdb [*programme*]

Débogueur C, C++ et Fortran 77. Permet d'exécuter pas à pas et de suivre l'évolution des variables d'un *programme*. Pour cela, il ne faut pas oublier de compiler le *programme* avec l'option **-g** du compilateur. Les commandes principales de ce débogueur sont les suivantes :

Commandes

break [*f*:]*fc*	Précise un point d'arrêt à la fonction *fc* (dans le fichier *f*).
run [*arg*]	Lance un programme (avec une liste d'arguments).
bt	Affiche la pile du programme.
print *exp*	Affiche la valeur de l'expression *exp*.
c	Continue l'exécution d'un programme (après un arrêt).
next	Exécute en bloc la ligne suivante du programme.
step	Exécute la ligne suivante du programme et entre dans une fonction si nécessaire.
help	Affiche une aide pour chaque commande.
quit	Quitte **gdb**.

gdb possède une sur-couche graphique plus facile d'utilisation **xxgdb**. Toutefois, cette dernière ne sera pas traitée dans ce manuel. Pour plus d'information, reportez-vous à la page de manuel de **xxgdb**.

grep

grep [*options*] *expression* [*fichiers*]

Recherche dans un ou plusieurs *fichiers* la présence d'une *expression* régulière.

	Options
	-b Précède chaque ligne de sa référence en octet depuis le début du fichier.
	-c Affiche uniquement le nombre de fois qu'apparaît l'*expression*.
	-h Affiche les lignes sans préciser le nom des *fichiers*.
	-i Ne fait pas de distinction entre majuscules et minuscules.
	-l Affiche le nom des *fichiers* dans les quels l'*expression* apparaît sans préciser les lignes (inverse de **-h**.
	-n Précède chaque ligne de son numéro.
	-s Supprime l'affichage des messages d'erreur.
	-v Affiche les lignes où l'*expression* n'apparaît pas.
	
	Exemples
	Affiche le nombre d'utilisateurs ayant pour shell Zsh
	`grep -c /bin/zsh /etc/passwd`
	Affiche la liste des fichiers inclus dans le programme c **prog.c**
	`grep '^#include' prog.c`

gzip	**gzip** [*options*] [*fichiers*] Réduit la taille d'un ou de plusieurs *fichiers* en utilisant l'algorithme de compression Lempel-Ziv et les renomme en *fichier***.gz**. **gunzip** et **zcat** permettent de revenir aux anciens fichiers. Ce compresseur est plus performant que **compress**. ***Options*** **-c** Ecrit sur le canal standard de sortie (sans changer les *fichiers*). **-d** Décompresse. **-f** Force l'opération sans demander de confirmation. **-l** Affiche des informations sur les *fichiers* compressés. **-r** Compresse le contenu de répertoires. **-t** Teste les *fichiers* compressés.
gunzip	**gunzip** [*options*] [*fichiers*] Décompresse un ou plusieurs *fichiers* traités par **gzip** (extension **.gz**) ***Options*** **-c** Ecrit sur le canal standard de sortie (sans changer les *fichiers*). **-f** Force l'opération sans demander de confirmation. **-l** Affiche des informations sur les *fichiers* compressés. **-r** Décompresse le contenu de répertoires. **-t** Teste les *fichiers* compressé.

head

head [*options*] *fichiers*

Affiche les premières lignes (par défaut les 10 premières) d'un ou plusieurs *fichiers* et/ou du canal d'entrée standard si « **-** » est précisé. Lorsque plusieurs fichiers sont traités, leur début est précédé d'une entête contenant leur nom entouré de ==> <==.

Options

-*nb*	Identique à **-n** *nb*.
-c *nb*[**bkm**]	Fixe la taille des débuts de fichiers à *nb* octet, *nb* blocs de 512 octets (**b**), *nb* kilo-octet (**k**) ou *nb* méga-octet (**m**).
-n *nb*	Fixe le nombre de lignes à *nb*.
-q	Interdit l'affichage d'entête.

kill

kill [*options*] *IDs*

Termine un ou plusieurs processus référencés par leurs *IDs*.

Options

-l	Affiche la listes des signaux.
-*signal*	Le *signal* (référencé par son numéro ou par son nom) est envoyé aux processus. Le signal 9 (KILL) tue systématiquement les processus.

Exemple

Terminaison du processus 5437.

```
$ ps
   PID TTY STAT TIME COMMAND
  5437  p3 S N  0:02 xdvi.bin -name xdvi unix-2.dvi
  5597  p1 R    0:00 ps
$ kill -9 5437
[1]  + 5597 killed   xdvi.bin
```

ld

ld [*options*] *fichiers*

Associe un ensemble de *fichiers* pour créer un exécutable. Par défaut, l'exécutable crée est nommé **a.out**. **ld** est l'éditeur de liens invoqué par **gcc**, **g++** et **g77**.

Options

-g	Effectue la compilation avec des symboles de débogage utilisés par **gdb**.
-L*rep*	Ajoute le répertoire *rep* dans la liste des répertoires explorés lors de la recherche des bibliothèques pour l'édition de liens.
-l*truc*	Ajoute la bibliothèque **lib***truc***.a** dans la liste des bibliothèques pour l'édition de liens.
-o *fichier*	Donne un *nom* au fichier généré (objet ou exécutable).

Cette commande ne s'utilise quasiment pas directement.

ln	**ln** [*options*] *fichier1 fichier2* **ln** [*options*] *fichiers répertoire* Crée des pseudonymes (liens) pour des fichiers, permettant ainsi d'y accéder sous différents noms. La première syntaxe permet de créer un lien nommé *fichier2* pour *fichier1*. Si *fichier2* est un fichier qui existe déjà, il est écrasé. La seconde syntaxe permet de créer des liens portant le même nom dans un autre *répertoire*. ***Options*** **-f** Force l'opération sans demander de confirmation (en cas d'écrasement de fichiers). **-n** Interdit l'écrasement de fichiers. **-s** Crée un lien symbolique. Cela permet de garder une référence (visible à la commande **ls -l**) avec le fichier lié. La commande **ln** s'utilise en fait *systématiquement* avec l'option **-s**.
lpq	**lpq** [**-P***imprimante*] [*IDs*] [*utilisateur*] Affiche l'état des impressions en attente sur une *imprimante*. Il est restreint à une liste de demandes (précisées par leurs numéros d'identification (*IDs*) ou par un nom d'*utilisateur*.
lpr	**lpr** [*options*] [**-P***imprimante*] [*fichiers*|-] Envoie des *fichiers* á une *imprimante*. Pour imprimer le canal d'entrée standard, il faut pr'eciser « - » à la place des *fichiers*. ***Options*** **-h** Retirer la page de garde. **-m** Envoie un mèl en fin d'impression. **-r** Efface le ficher après la mise en attente ou l'impression (avec l'option **-s**). **-s** Crée un lien symbolique. Habituellement les *fichiers* sont copiés dans le répertoire d'attente (**/var/spool/lpd/...**). Cette option permet de ne créer que des liens symboliques plutôt que de copier de gros *fichiers*. Cela veut dire que les *fichiers* ne doivent pas être modifier ou effacer avant d'avoir été entièrement imprimés.
lprm	**lprm** [**-P***imprimante*] [*IDs*] [*utilisateur*] Annule les demandes d'impression faites par **lpr**. Il est possible de préciser les demandes par leur(s) *IDs*, par l'*imprimante* pour laquelle elles sont actuellement en attente ou par le nom de l'*utilisateur* qui a effectué ces demandes d'impression (il faut avoir des droits particuliers pour cela). Utilisez **lpq** pour connaître les numéros d'identifications (*ID*) des demandes d'impression.
ls	**ls** [*options*] [*noms*] Seule, **ls** liste les fichiers dans le répertoire courant. Si un ou plusieurs *noms* sont précisés, **ls** affiche les fichiers qui appartiennent aux répertoires nommés ou dont le nom correspond à l'un de ceux spécifiés (les *noms* peuvent contenir des métacaractères). Les *options* permettent de modifier la liste et de l'enrichir d'informations secondaires.

Options

-a	Affiche tous les fichiers même ceux cachés commençant par un « . ».
-b	Présente les caractères non affichables en octal.
-c	Affiche les fichiers par date de création/modification.
-C	Affiche les informations en colonnes classées verticalement (par défaut).
-d	Affiche uniquement les répertoires mais pas leur contenu.
-f	Interprète tous les *noms* comme des répertoires (les fichiers sont ignorés).
-F	Etiquette les répertoire par un « **/** », les exécutables par un « ***** » et les liens symboliques par un « **@** ».
-i	Liste les I-node de chaque fichier.
-l	Affichage long (droits, nom et groupe du propriétaire, taille, date de dernière modification).
-L	Affiche le contenu d'un répertoire pointé par un lien symbolique plutôt que le lien lui-même.
-m	Affiche les noms séparés par des virgules.
-n	En complément de **-l**, affiche le nom et le groupe des propriétaires en fonction de leur UID et de leur GID.
-o	Identique à **-l** sans afficher le groupe des propriétaires.
-q	Remplace les caractères non affichables par « **?** ».
-r	Classe les noms dans l'ordre inverse.
-R	Affiche le contenu des sous-répertoire de la même façon que le répertoire courant.
-s	Affiche la taille des fichiers en blocs (classiquement 512 octets ou 1 kilo-octet suivants les systèmes).
-t	Classe les fichiers par date de dernière modification.
-C	Affiche les informations en colonnes classées horizontalement (par défaut).
-1	Affiche en information une par une en ligne.

Exemples

Liste tous les fichiers du répertoire courant en colonnes avec des étiquettes pour les fichiers particuliers.

```
ls -aCF
```

Liste les fichiers C du répertoire **prog** par ordre de mise à jour.

```
ls -1tr prog/*.c
```

mail

mail [*options*] [*utilisateurs*]

Lit (si aucun *utilisateur* n'est spécifié) ou envoie à un ou plusieurs *utilisateurs* un courrier électronique. Il est possible d'obtenir une liste sommaire des commandes par « **?** » en mode lecture. Pour l'envoie de courrier, le corps du mèl est récupéré du canal d'entrée standard (fin marquée par *EOF*).

	Options d'envoie de mèl
	-b *u* — Envoie en secret une copie à *u*.
	-c *u* — Envoie une copie à *u*.
	-s *s* — Fixe le sujet du mèl à *s* (par défaut, **mail** demande un sujet pour la lettre).
	
	Option de lecture de mèl
	-f *b* — Indique que votre boite à mèl est *b* (par défaut celle-ci porte votre nom d'utilisateur et se trouve dans le répertoire **/var/spool/mail/**).
	
	Exemple
	Envoie de **prog.c** à **bill**.
	`cat prog.c │ mail -s "Programme C" bill`
	ou
	`mail -s "Programme C" bill < prog.c`

man	**man** [*options*] [*section*] *sujet*
	
	Recherche et affiche la page de manuel associée à un *sujet* (souvent une commande Unix, une fonction C ou Fortran ...). La recherche s'effectue de la section 1 à la section 10 et *man* s''arrête à la première page trouvée. Il est possible de préciser à quelle *section* se rapporte le *sujet* recherché. Si le *sujet* contient un **/** **man** essaie d'afficher le fichier correspondant au nom du *sujet*.
	
	Options
	-a — Affiche toutes les pages de manuel associé au *sujet*.
	-K — Recherche une chaîne de caractères parmi toutes les pages.
	-M *liste* — précise la *liste* des répertoires à explorer. Normalement, **man** récupère la liste contenu dans la variable **MANPATH**.
	-P *disp* — Utilise *disp* pour afficher les pages.
	-w — Affiche les fichiers contenants les pages de manuel. Sans *sujet*, **man** affiche la liste des répertoires explorés (variable **MANPAH**).
	
	Exemples
	Affiche la page **truc** contenue dans le répertoire courant.
	`man ./truc.4`
	Liste les pages concernant **printf**.
	`man -wa printf`

mesg	**mesg** [*options*] Autorise ou interdit l'affichage de messages dans votre terminal. Sans option, **mesg** affiche l'état de la permission. ***Options*** **n** — Interdit les messages de **write**. **y** — Autorise les messages de **write** (par défaut).
mkdir	**mkdir** [*options*] *répertoires* Crée un ou plusieurs *répertoires*. Vous devez avoir les droits d'écriture dans le répertoire père pour pouvoir créer un répertoire. Voir aussi **rmdir**. ***Options*** **-m** *mode* — Fixe les droits des nouveaux *répertoires*. **-p** — Crée les répertoires pères nécessaires s'ils n'existent pas. ***Exemple*** Création du répertoire **prog**. `mkdir prog`
more	**more** [*options*] [*fichiers*] Affiche les *fichiers* dans un terminal page à page. Certaines touches vont vous permettre de naviguer,et faire évoluer l'affichage. ***Touches*** RETURN — Passe à la ligne suivante. ESPACE — Passe à la page suivante. **h** — Affiche un écran d'aide. **q** — Quitte. **/** — recherche dans un fichier. **:n** — Passe au fichier suivant. ***Options*** **-c** — Affiche une page entière sans défilement. **-d** — **more** affiche en dernière ligne « [Press space to continue, 'q' to quit.] » et « [Press 'h' for instructions.] » au lieu de faire « sonner » le terminal lorsque l'on appuie sur une touche sans fonctionnalité. **-f** — Compte les lignes réelles plutôt que les lignes de l'écran. **-l** — Ignore le caractère ^L. **-s** — Ramène plusieurs lignes vides d'affilé en une seule. **-*nb*** — Utilise *nb* lignes par page (par défaut c'est le nombre total de lignes du terminal). **+*num*** — Commence l'affichage à la ligne *num*.

Exemples
more prog.c cat prog.c | more

mv

mv [*options*] *sources destination*

Permet de déplacer ou de renommer des fichiers et des répertoires. **mv** fonctionne de la façon suivante :

Sources	***Destination***	***Résultat***
Fichier	*nom*	Renomme le fichier en *nom*.
Fichier	Fichier existant	Ecrase le fichier destination par le fichier source.
Répertoire	*nom*	Renomme le Répertoire en *nom*.
Répertoire	Répertoire existant	Déplace le répertoire source dans le répertoire destination.
Un ou plusieurs fichiers	Répertoire existant	Déplace les sources dans le répertoire destination.

Options

-f	Force l'opération sans demander de confirmation.
-i	Demande une confirmation avant d'écraser un fichier existant.
--	Indique la fin des options de **mv**. Ceci est utile pour pouvoir agir sur un fichier commençant par « - ».

passwd

passwd [*utilisateur*]

Permet de changer le mot de passe d'un *utilisateur*. Seul l'administrateur peut changer le mot de passe d'un autre utilisateur que lui même.

printenv

printenv [*variables*]

Affiche les *variables* d'environnement spécifiée. Si aucune variable n'est précisée, **printenv** affiche toutes les variables d'environnement.

ps

ps [**-**][*options*] [*PIDs*]

Affiche la liste des processus actif ou simplement ceux précisés par leur *PIDs* (numéro d'identification de processus). Sans option, **ps** affiche dans un format court la liste des processus lancés par l'utilisateur. Le « **-** » est un héritage de la normalisation BDS ; il ne devrait plus apparaître dans la normalisation Unix-98. Les informations sont affichées en colonnes. Suivant les options spécifiées un certain nombre de champs sont affichés dont les entêtes sont détaillées ci-dessous.

Options

l	Format long.
u	Format utilisateur.
j	Format tâche.
s	Format signal.
v	Format mémoire virtuelle.
m	Format mémoire.
f	Affichage en « arbre ».
a	Affichage des processus des autres utilisateurs.
x	Affichage de tous les processus (même ceux non liés à un terminal).
c	Raccourcit le nom de la commande associée au processus.
e	Affiche les variables d'environnement.
w	Allonge d'une ligne le format d'affichage. Vous pouvez mettre jusqu'à 100 **w**.
h	Retire l'entête.
r	Affiche uniquement les tâches en action.
n	Affiche les champs USER et WCHAN sous forme de nombre.

Champs

COMMAND	Commande associée.
CPU	Pourcentage de puissance de calcul utilisée.
MEM	Pourcentage de mémoire utilisée.
NI	Niveau de priorité. Plus il est élevé, moins le processus est prioritaire.
PAGEIN	Nombre de pages mémoire.
PID	Numéro d'identification du processus.
PPID	Numéro d'identification du père du processus.
PRI	Ordre de priorité.
RSS	Taille du programme en mémoire.
SHARE	Mémoire partagée.
SIZE	Taille de mémoire virtuelle (texte+données+pile).
START	Heure de lancement de la tâche.
STAT	Information d'état : R $\leftrightarrow$ actif, S $\leftrightarrow$ en sommeil, D $\leftrightarrow$ en sommeil non interrompable, T $\leftrightarrow$ arrêté, Z $\leftrightarrow$ zombie, W $\leftrightarrow$ sans page résidente et N $\leftrightarrow$ si le processus est moins prioritaire.
SWAP	Mémoire secondaire (en Ko).
TIME	Temps d'inactivité.
TRS	Taille résidente de texte.
TTY	Nom du terminal.
UID	Numéro d'identification du propriétaire.
USER	Nom du propriétaire.
WCHAN	Nom de la fonction du noyau dont le processus attend une réponse.

rm	**rm** [*options*] *fichiers* Efface un ou plusieurs *fichiers*. Vous devez avoir les droits d'écriture dans le répertoire sans pour autant avoir des lroits sur les *fichiers* eux-mêmes. Si vous n'avez pas de droits d'écriture sur les *fichiers*, un message vous demandera si vous confirmez l'opération ou non. ***Options*** **-f** Efface les fichiers protégés en écriture sans demander de confirmation. **-i** Demande une confirmation pour tout effacement. **-r** Efface des répertoires ainsi que leurs contenus. A utiliser avec précaution. **--** Indique la fin des options de **rm**. Ceci est utile pour pouvoir effacer un fichier commençant par « - ».
rmdir	**rmdir** [*options*] *répertoires* Efface les *répertoires* (les noms seulement pas leurs contenus). Pour cela les répertoires doivent être vide (sinon, **rm -r** peut être utilisé à la place). Voir aussi **mkdir**. ***Options*** **-p** Efface les répertoires pères précisés s'ils sont vides. Ceci est utile pour effacer des arbres de répertoires. **-s** Supprime les messages d'erreurs générés par **-p**.
tar	**tar** [-]*clé*[*options*] [*fichiers*] Gère un ensemble de *fichiers* associés dans un fichier de sauvegarde. Ceci est le plus souvent utilisé pour faciliter le transport de plusieurs *fichiers*. La liste des *fichiers* peut contenir des répertoires ; les fichiers et répertoires contenus dans ces derniers seront archivés en conservant l'arborescence initiale. Si le nom de la ***Clé*** **c** Crée une sauvegarde. **r** Ajoute des *fichiers* à une sauvegarde. **t** Liste le contenu d'une sauvegarde. **u** Ajoute des *fichiers* nouveaux ou plus récents à une sauvegarde. **x** Extrait des *fichiers* d'une sauvegarde. Par défaut, **tar** extrait tous les fichiers contenus dans la sauvegarde. ***Options*** **f** *sauv* Donne le nom *sauv* à la sauvegarde. Si *sauv* vaut « **-** », l'archive est envoyée sur le canal de sortie standard. Par défaut, la sauvegarde est **/dev/rmt0** (c'est-à-dire le lecteur de bande). Ce paramètre est donc essentiel. Par convention, l'extension d'une sauvegarde est **.tar**. **v** Affiche des informations concernant les *fichiers*.

	Exemple
	Création de la sauvegarde **prog.tar** contenant les fichiers du répertoire **c/**.
	`tar cfv prog.tar c/`
	Extraction du fichier **c/td1.c** de la sauvegarde.
	`tar xfv prog.tar c/td1.c`

sort	**sort** [*options*] [*fichiers*]
	Classe et affiche les lignes d'un ou plusieurs *fichiers* classiquement par ordre alphabétique.
	
	Options
	-b Supprime les espaces et les tabulations.
	-c Vérifie si les fichiers sont triés ou pas.
	-d Trie comme pour un répertoire téléphonique ne tenant compte que des lettres et des chiffres.
	-f Ignore la différence entre lettres minuscule et majuscule.
	-i Ignore les caractères non imprimables.
	-M Compare les 3 premiers caractères en tant que mois (JAN $<$ FEB $< \ldots <$ DEC).
	-m Lie les fichiers en entrée.
	-n Trie suivant l'ordre arithmétique.
	-o*fic* Redirige le canal de sortie standard vers le fichier *fic*.
	-r Trie dans l'ordre inverse.
	-t*c* Indique que le caractère de séparation entre les champs est le caractère *c*.
	-u Enlève les lignes identiques.
	+*n*[**-***m*] Les champs pris en compte pour le classement sont alors les champs *n* à *m*-1 (par défaut, la fin de la ligne).
	
	Exemples
	Classe la liste des fichiers par taille décroissante.
	`ls -1s | sort +1n -r`
	Classe la liste des comptes utilisateurs par rapport à leur UID (3^e champ).
	`sort +2n -t: /etc/passwd`

split	**split** [*options*] *fichier* [*debutN*] Découpe un *fichier* en plusieurs fichiers de taille identiques (soit en ligne soit en octet) dont les noms commencent par *debutN* (par défaut **x**). Par défaut **split** découpe les fichiers par 1000 lignes. Si *fichier* vaut « **-** », **split** récupère le canal d'entrée standard.

	Options **-***nb* — Identique à **-l** *nb*. **-b** *nb*[**bkm**] — Fixe la taille des fichiers de sortie à *nb* octet, *nb* blocs de 512 octets (**b**), *nb* kilo-octet (**k**) ou *nb* méga-octet (**m**). **-l** *nb* — Fixe le nombre de lignes à *nb*.
strings	**strings** *fichiers* Affiche les séquences de caractères imprimables trouvées dans un ou plusieurs *fichiers*. ***Options*** **-** — Identique à **-a**. **-a** — Recherche dans tout le fichier et non simplement dans la zone de données des fichiers objets. **-n** *n* — Fixe la taille minimale des chaînes à *n* caractères (par défaut 4). **-o** — Identique à **-t o**. **-t** *m* — Affixe la position de la chaîne dans le fichier dans la base fixée par *m* : octal ↔ **o**, hexadécimal ↔ **x** ou décimal ↔ **d**.
strip	**strip** *fichier* Supprime les symboles ajoutés par le compilateur. Cela permet de réduire la taille d'un exécutable, d'un fichier objet ou d'une archive (crée par **ar**).
tail	**tail** [*options*] *fichiers* Affiche les dernières lignes (par défaut les 10 dernières) d'un ou plusieurs *fichiers* et/ou du canal d'entrée standard si « **-** » est précisé. Lorsque plusieurs fichiers sont traités, leurs fins sont précédées d'une entête contenant leur nom entouré de `==> <==`. ***Options*** **-***nb* — Identique à **-n** *nb*. **-c** *nb*[**bkm**] — Fixe la taille des fins de fichiers à *nb* octet, *nb* blocs de 512 octets (**b**), *nb* kilo-octet (**k**) ou *nb* méga-octet (**m**). **-f** — Continue d'attendre de nouveaux caractères. Ceci est très utilise pour suivre l'évolution d'un fichier de log. **-n** *nb* — Fixe le nombre de lignes à *nb*. **-q** — Interdit l'affichage d'entête. ***Exemple*** Affiche des 10 dernières lignes du fichier **/var/log/message** avec poursuite. `tail -10 -f /var/log/message`

telnet	**telnet** *machine* [*port*]
	Permet de communiquer avec une *machine* distante en utilisant le protocole TELNET. Il est possible de préciser sur quel *port* doit s'établir la communication (par défaut 23). Vous pouvez obtenir le mode de commande de **telnet** en tapant CONTROL-].
	Exemple
	`$ telnet marval`
	`Trying 193.55.62.80...`
	`Connected to marval.univ-mlv.fr.`
	`Escape character is '^]'.`
	`...`

tee	**tee** [**-a**] *fichiers* Duplique le canal d'entrée standard vers un ou plusieurs *fichiers* et vers le canal de sortie standard. L'option **-a** permet d'ajouter le canal d'entrée standard à des *fichiers* sans les écraser.
uncompress	**uncompress** [*options*] [*fichiers*] Décompresse un ou plusieurs *fichiers* traités par **compress** (extension **.Z**) ***Options*** **-c** Ecrit sur le canal standard de sortie (sans changer les *fichiers*). **-f** Force l'opération sans demander de confirmation. **-l** Affiche des informations sur des *fichiers* compressés.

uptime	**uptime**
	Donne une ligne d'information sur la machine : l'heure, depuis combien de temps la machine fonctionne, combien d'utilisateurs sont connectés et la charge de calcul moyenné sur 1, 5 et 15 minutes.
	Exemple
	`$ uptime`
	`1:05pm up 4:35, 1 user, load average: 0.09, 0.20, 0.22`

which	**which** *programme* Indique le chemin d'accès exact d'un *programme* Unix.

	Exemple
	`$ which gcc`
	`/usr/bin/gcc`

who	**who** [*options*] [**am i**]
	Affiche des informations au sujet des utilisateurs actuellement logués. En l'absence de paramètre, **who** affiche les informations suivantes : nom de l'utilisateur, nom du terminal, date et heure de connection et nom de la machine distante ou du terminal X.
	Options
	-H Affiche une ligne d'entête.
	-i Ajoute le temps d'inactivité de chaque utilisateur.
	-m N'affiche que les informations concernant l'utilisateur associé au canal d'entrée standard.
	-q Affiche uniquement le nom des utilisateurs et leur nombre.
	-T Ajoute l'état de réception des messages : **+**, **-** ou **?**.
	-u Identique à **-i**.
	-w Identique à **-T**.
	La commande **who am i** (équivalente à **whoami**) indique uniquement le nom de l'utilisateur propriétaire du canal d'entrée standard.
	Exemple
	`$ who`
	`mazet    ttyp1    Oct 15 10:10 (:0.0)`

write	**write** *utilisateur* [*terminal*] *message* Ecrit un *message* dans le terminal d'un *utilisateur* logué. Un message se termine par *EOF*. Si un *utilisateur* est logué sur plusieurs terminaux, il faut préciser le numéro du *terminal*. Le message s'affichera si l'utilisateur l'a permis (grâce à la commande **mesg**).
yppasswd	**yppasswd** [*options*] [*utilisateur*] Permet de changer le mot de passe d'un *utilisateur* à travers le réseau (lorsque la table des mots de passe est distribuée via NIS). Seul l'administrateur peut changer le mot de passe d'un autre utilisateur que lui même. ***Options*** **-f** Equivalent à **ypchfn**. **-l** Equivalent à **ypchsh**. **-p** Equivalent à **yppasswd**.

ypchfn	**ypchfn** [*options*] [*utilisateur*] Permet de changer les informations utilisées par **finger**. Seul l'administrateur peut changer le informations d'un autre utilisateur que lui même. ***Options*** **-f** Equivalent à **ypchfn**. **-l** Equivalent à **ypchsh**. **-p** Equivalent à **yppasswd**.
ypchsh	**ypchsh** [*options*] [*utilisateur*] Permet de changer de shell. Seul l'administrateur peut changer le shell d'un autre utilisateur que lui même. ***Options*** **-f** Equivalent à **ypchfn**. **-l** Equivalent à **ypchsh**. **-p** Equivalent à **yppasswd**.
zcat	**zcat** [*options*] *fichiers* Décompresse et affiche le contenu un ou plusieurs *fichiers* compressés par **compress** ou **gzip**. Cette commande est équivalente à **uncompress -c** ou à **gunzip -c**.

Chapitre 3

Le Shell et ses commandes

3.1 Introduction au Shell

Commençons par une petite métaphore : supposons qu'un système Unix puisse s'assimiler à une voiture. Pour conduire une voiture, vous appuyez sur l'embrayage, vous actionnez le levier de vitesse, vous passez une vitesse et vous lâchez l'embrayage et appuyez sur l'accélérateur. Ce dont vous venez de vous servir ne sont en fait que des leviers de commandes, vous ne savez pas comment vos ordres ont été transmis au moteur de la voiture et, la plupart d'entre vous n'en ont rien à faire.
Un Shell a à peu près la même fonction vis-à-vis d'un système Unix. Il vous permet de naviguer dans le système, exécuter des commandes sans avoir à vous soucier de la façon dont réagit le système. C'est essentiellement un interpréteur de commande mais il permet aussi de gérer des processus, un ensemble de variables paramètrisant votre environnement et de programmer des tâches relativement complexes (fichier contenant un ensemble de commande Unix).
Suivant le système sur lequel vous travaillez, un ou plusieurs shells sont disponibles; ils n'ont pas exactement les mêmes fonctions ni la même syntaxe. La figure 3.1 vous permettra de mieux comprendre les différentes évolutions des shells Unix.

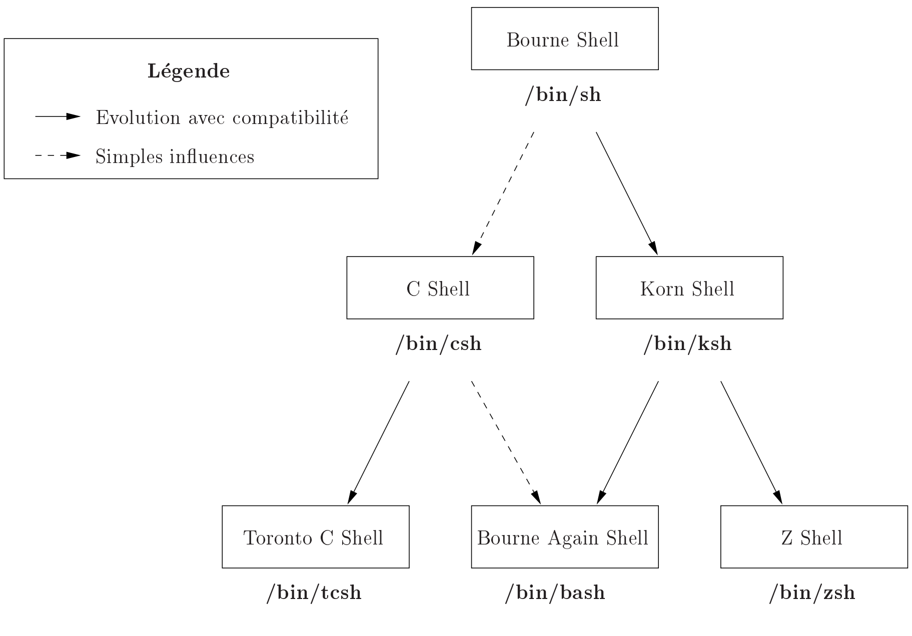


FIG. 3.1 – Evolution des Shells Unix.

Le Bourne Shell (**sh**) est le premier shell crée et il reste celui (ou un de ses clones) dont tout système Unix a besoin pour lancer les outils de configuration au démarrage de la machine. Les shell **ksh**, **bash** et **zsh** utilisent des syntaxes plus riches mais qui restent compatibles avec **sh**. Le **csh** et le **tcsh** ont par contre des

syntaxes beaucoup plus différentes (de **sh**) et très proches de celle du langage C.
La seule chose dont vous pouvez être sûr, c'est qu'au moins un shell compatible **sh** existe sur toute machine Unix. C'est pourquoi, ce manuel se limitera à décrire les fonctions essentielles du Bourne Shell bien que je vous recommande d'utiliser **zsh**.

3.2 Les méta-caractères

Ils permettent de créer des noms génériques.

*	Correspond à une chaîne de caractères quelconque (même vide).
?	Correspond à un caractère quelconque.
[*abc...*]	Correspond à un des caractères spécifiés dans la liste. Il est possible de décrire des listes de caractères en ne précisant que le premier caractère et le dernier caractère de la liste séparé par un moins (par exemple : a-z, I-M, 0-9).
[!*abc...*]	Correspond à tout caractère non spécifié dans la liste.

Exemples

`$ ls toto*`	Liste les fichiers **toto**, **toto.c** et **toto.txt**.
`$ cat prog?.c`	Affiche le contenu du fichier **prog2.c**.
`$ ls [A-D]*`	Liste les fichiers commençant par **A**, **B**, **C** ou **D**.

3.3 Les guillemets

Ils servent à délimiter des chaînes des caractères mais permettent aussi de créer de nouvelles chaînes par concaténation.

' '	Tous les caractères compris entre les apostrophes (*quote* en anglais) sont pris en compte tel quel.
‘ ‘	La chaîne comprise entre les anti-apostrophes (*back-quote* en anglais) est évaluée en tant que commande et son résultat vient se substituer à la chaîne initiale.
\	Tout caractère précédé d'un **** est pris en compte tel quel. Ceci est très utile pour affiche tous les caractères interprétés par le shell.
" "	Seul les caractères non interprétables (espace compris) sont pris en compte tel quel, les autres sont interprétés.

3.4 Les séparateurs

Ils jouent un rôle particulier en shell, puisque, la plus part du temps, ils ne sont pas utilisés.

cmd **&**	Lance la commande *cmd* en arrière plan.
cmd1 **;** *cmd2*	Séparateur de commandes. Permet d'écrire deux commandes sur la même ligne.
(*cmd1* **;** *cmd2*)	Ensemble de commandes groupées.
cmd1 | *cmd2*	*Pipe*. Permet de rediriger le canal de sortie de la commande *cmd1* vers le canal d'entrée de *cmd2*. Les informations vont être traitées d'abord par *cmd1* puis par *cmd2*.
cmd1 **&&** *cmd2*	Exécute la commande *cmd1* puis, si *cmd1* s'est bien déroulée, exécute *cmd2*.
cmd1 || *cmd2*	Exécute la commande *cmd1* puis, si *cmd1* s'est mal déroulée, exécute *cmd2*.
{ *cmd1* **;** *cmd2* **}**	Permet d'exécuter un ensemble de commandes dans un sous-shell.

Exemples

`$ gcc -c prog.c &`	Lance une compilation en tâche de fond.
`$ cd ; ls`	Revient à la racine perso. et liste les fichiers.
`$ (date; who) > log`	Les résultats sont envoyés vers un fichier.
`$ ls -1s | sort -n`	Liste les fichiers par taille croissante.

3.5 Les redirections

Il est souvent utile de pouvoir changer les canaux d'entrée, de sortie ou d'erreur de commande.

cmd [*n*]> *fich*	Envoie le canal *n* de la commande *cmd* vers le fichier *fich* (en écrasant *fich* s'il existe). Si *n* n'est spécifié ou s'il vaut 1, c'est le canal de sortie, s'il vaut 2 c'est le canal d'erreur.
cmd [*n*]>> *fich*	Envoie le canal *n* de la commande *cmd* vers fichier *fich* (en ajoutant à *fich* s'il existe). Si *m* n'est spécifié c'est le canal de sortie.
cmd < *fich*	Renvoie dans le canal d'entrée de la commande *cmd* le fichier *fich*.
cmd << [*texte*]	Renvoie dans le canal d'entrée de la commande *cmd* toutes les chaînes de caractères provenant du canal d'entrée standard jusqu'à qu'apparaisse la phrase *texte*. Si aucun *texte* n'est spécifié, la lecture s'arrête sur *EOF*.
cmd [*m*]>**&***n*	Envoie le canal *n* vers le canal *n*. Si *m* n'est spécifié c'est le canal de sortie.

Exemples

`$ cat prog.c	grep ^#`	Affiche les lignes de **prog.c** commençant par **#**.
`$ cat func.c >>  prog.c`	Ajout de **func.c** en fin de **prog.c**.
`$ gcc -c prog.c 2>g.log`	Récupère tous les messages d'erreurs de la compilation dans **g.log**

3.6 Les variables

Elles permettent de stocker des données qui sont utilisées soit par le shell lui-même, soit par les commandes et logiciels Unix (récupération de fichiers de ressources, liste d'options par défaut, ...).

var=*val*	Met *val* dans la variable *var*.
${*var***}**	Utilise le contenu de la variable *var*. Les accolades sont facultatives si le nom de la variable est séparable.
${*var* **:-***val***}**	Utilise le contenu de la variable *var* si elle existe sinon utilise la valeur *val*.
${*var* **:=***val***}**	Utilise le contenu de la variable *var* si elle existe sinon utilise la valeur *val* et assigne *val* à *var*.
${*var* **: ?***val***}**	Utilise le contenu de la variable *var* si elle existe sinon affiche « *var* **:** *val* » ou « *var* **: parameter not set** » si *val* est la chaîne nulle.
${*var* **:+***val***}**	Utilise la valeur *val* si la variable *var* existe sinon ne fait rien.

Exemples

`$ t=chose`	La variable **t** reçoit **chose**
`$ t2=$t`	La variable **t2** reçoit le contenu de **t**.
`$ t3="pas grand $t"`	La variable **t3** reçoit **pas grand chose**.
`$ t4='$t'`	La variable **t4** reçoit **$t**.

3.7 Les variables d'environnement du shell

Elles contiennent les informations nécessaires au bon fonctionnement du shell.

$#	Nombre d'arguments de la ligne de commande.
$-	Options actuellement utilisées.
$?	Retour de la dernière commande exécutée.
$$	Numéro du processus courant.
$!	Numéro du dernier processus lancé en arrière plan.
$0	Premier argument (c'est-à-dire, la commande elle-même).
$*n*	n^e argument (n est compris entre 1 et 9).
$*	Tous les arguments (**"$1 $2 ..."**).

CDPATH	Liste de répertoires utilisée par **cd** en tant que raccourcis. Les répertoires sont séparés par des deux points. Par défaut, cette variable est vide.
HOME	Contient le nom du répertoire principal de l'utilisateur du shell.
IFS	Liste de caractères marquant la séparation entre deux mots. Par défaut, cette variable contient les caractères « espace », « tabulation » et « fin de ligne ».
MAIL	Contient le nom du fichier boite aux lettres de l'utilisateur.
MAILCHECK	Temps (en seconde) entre deux vérification de l'état de la boite aux lettres.
MAILPATH	Liste des boites aux lettres séparées par des deux points. Par défaut le shell affiche, « **You have new mail** » lors de la réception de nouveaux messages.
PATH	Liste des chemins d'accès des commandes Unix (les répertoires sont séparés par des deux points). Par défaut, cette variable contient **/usr/bin**.
PS1	Prompt standard. Par défaut **$**.
PS2	Prompt de second niveau (utilisé pour des commandes tapées sur plusieurs lignes). Par défaut >.
SHELL	Nom du shell utilisé. Si vous utilisez un Bourne Shell, ce doit être **/bin/sh**.
TERM	Type du terminal. Si vous avez un problème, n'hésitez pas à revenir à un terminal basique tel que **vt100**.

3.8 Les fonctions shell

La définition d'une fonction s'écrit comme ceci :
nom **() {** *liste de commandes* **; }**
En Korn Shell, on peut ajouter le mot clé **function** mais cela n'ajoute qu'un peu de clarté à la programmation. Les fonctions s'utilisent alors comme de simples commandes.

Exemple

```
$ count () {
> ls $@ | wc -l
> }
```

La fonction **count** permet de compter le nombre de fichiers d'un répertoire ou plusieurs répertoires.

3.9 Le dièse et le nombre magique

Le dièse permet d'écrire des commentaires dans un fichier de commandes shell. Par contre, en début de tout fichier de commande shell, on trouve :
 # !*shell*
C'est le « nombre magique » qui indique que le fichier est un programme qu'il faut exécuter via le *shell* spécifié.

Exemple

Le programme **count** s'écrit donc :

```
#!/bin/sh

# Message
echo -n " Nombre de fichier dans le(s) repertoire $@ :"
# Commande comptant les fichiers
ls | wc -l
```

3.10 Les commandes shell

<table>
<tr><td>alias</td><td>alias [nom[='commande']]

C'est en fait une commande Korn Shell mais comme la plupart des shells l'inclue, vous pouvez l'utiliser pour vos fichiers de configurations. alias permet d'assigner une commande (avec ou sans paramètre) à un nom. Si la commande (et le signe égal) n'est pas précisé, alias affiche (si il existe) la définition de l'alias nom. Sans paramètre, alias affiche la liste de tous les alias précédemment définis. Voir unalias.

Exemples

<pre>allps='/bin/ps aux| grep -v "^root"'
clean='/bin/rm -rf *.o *~'
copy=cp
del='/bin/rm -i'
dir='ls -la'
la='/bin/ls -la'
ll='ls -lg'
ls='/bin/ls -CF'
m=less
nps='/bin/ps ux'
xtar='tar -xvf'</pre></td></tr>
<tr><td>bg</td><td>bg [tâche]

Passe en arrière plan une tâche référencée par son numéro (voir la commande jobs). Celle-ci a été le plus souvent interrompue par un CONTROL-Z. Sans paramètre, bg agit sur le dernier processus interrompu.
Pour un exemple, reportez-vous à la commande jobs.</td></tr>
<tr><td>break</td><td>break [n]
Permet de sortir de boucles for, while ou until. Le paramètre permet de sortir du n^e niveau de boucles.</td></tr>
<tr><td></td><td></td></tr>
</table>

<table>
<tr>
<td>case</td>
<td>
case valeur in
mot1) cmd1 ; ;
mot2) cmd2 ; ;
.
.
.
esac

Exécute la permière liste de commandes cmd1 si la valeur correspond au mot1 et sort (c'est-à-dire, l'exécusion reprend après le mot clé esac). Exécute la seconde liste de commandes cmd2 si la valeur correspond au mot2 et sort. Et ainsi de suite. Les mots peuvent comporter des méta-caractères et il est possible de faire des choix multiples (séparés par des |). Cette structure est très utile pour la gestion des paramètres d'un programme shell.

Exemple
Ce petit programme analyse le premier argument entré et initialise les variables adéquates.
<pre>
#!/bin/sh

case $1 in
  oui|non)
    reponse=presente
    ;;
  -[lL])
    liste=1
    ;;
  *)
    echo Usage: $0 [oui|non] [-lL]
    exit 1
    ;;
esac
</pre>
</td>
</tr>
<tr>
<td>cd</td>
<td>
cd [rép]
cd [-]
cd ancien noouveau

Sans argument, cd change le répertoire de travail en la racine de l'utilisateur. Sinon, elle change le répertoire de travail en rép. Si rép est un chemin relatif (ne commençant pas par /) ne se trouvant pas dans le répertoire courant, le shell recherche parmi les chemins indiqués dans la variable CDPATH. cd - renvoie dans le précédent répertoire de travail. La derniere syntaxe permet de remplacer dans le nom du répertoire de travail courant ancien par nouveau.
</td>
</tr>
</table>

	Exemple
	
	`$ pwd`
	`/var/spool/mail`
	`$ cd mail lpd`
	`/var/spool/lpd`

do	**do** Mot clé intiquant le début d'une liste de commandes se terminant par **done** pour les boucles **for**, **while** et **until**.
done	**done** Mot clé indiquant la fin d'une liste de commandes se commançant par **do** pour les boucles **for**, **while** et **until**.
echo	**echo** [**-n**] [*chaîne*] Affiche la *chaîne* de caractères sur le canal de sortie standard. Si l'option **-n** est précisée, la ligne ne se termine pas par un retour à la ligne. Certains caractères non affichables peuvent être utilisés via des codes précédés d'un ****. ***Codes*** **\b** — Effacement. **\c** — Omision de la fin de ligne (identique à l'option **-n**). **\f** — *Formfeed*. **\n** — Nouvelle ligne. **\r** — Retour chariot. **\t** — Tabulation. **** — *Backslash*. **\0***nnn* — Code ASCII *nnn* en octal.
esac	**esac** Voir **case**.
exec	**exec** *commande* Permet d'exécuter une commande à la place du shell courant. Ceci est utile pour changer de shell sans conserver le précédent.
exit	**exit** [*n*] Permet de sortir d'un script shell avec le status *n*. *n* peut valoir 0 (succés) ou tout autre valeur non nulle (échec). Si *n* n'est pas précisé, **exit** renvoie le status de la dernière commande exécutée.
export	**export** [*variables*] Permet de rendre global (visible pour des shells secondaires) des *variables* qui, par défaut, sont locales au shell courant. Sans paramètre, **export** affiche la liste des variables actuellement exportées.

	Exemple
	`TERM=vt100`
	`export TERM`

fg	**fg** [*tâche*] Relance en premier plan une *tâche* référencée par son numéro (voir la commande **jobs**). Celle-ci a été le plus souvent interrompue par un CONTROL-Z. Sans paramètre, **fg** agit sur le dernier processus interrompu.
fi	**fi** Mot clé indiquant la fin d'une structure **if**

for	**for** *x* **in** *liste*
	do
	    commandes
	done
	Pour la variable *x* appartenant à *liste* effectue l'ensemble de *commandes*.
	Exemple
	Ce petit script permet de compresser tous les fichiers qui ne le sont pas encore.
	`for i in 'ls *.[^Z]'`
	`do`
	`  compress $i`
	`done`

if	**if** *cond1* **then** *cmd1* [**elif** *cond2* **then** *cmd2*] . . . [**else** *cmd3*] **fi** Si la condition *cond1* est atteinte, effectue la commande *cmd1* ; sinon, si la condition *cond2* est atteinte, effectue la commande *cmd2* ; sinon effectue la commande *cmd3*. Les conditions s'écrivent grâce a la commande shell **test**.

	Exemple
	Ce petit morceau de programme ajoute un 0 à gauche au nombre inférieur à 10.
	`if [ $n -lt 10 ]`
	`then`
	`  n="0"$n`
	`fi`

jobs	**jobs** [*option*] [*tâches*]
	Affiche la liste des tâches (ou simplement celles précisées par *tâches*) lancées dans le shell ainsi que leur état (« **running** » lancée en arrière plan ou « **suspended** » interrompue). Vous pouvez intervenir sur ces tâches via les commandes **bg**, **fg** et **kill** en spécifiant leur numéro précédé d'un **%**.
	Option
	-l Affiche en plus le numéro du processus.
	Exemples
	`$ jobs`
	`[1]    running    xload`
	`[2]  + suspended  xroger`
	`[3]  - running    xhextris`
	`$ kill %1`
	`[1]    671 terminated  xload`
	`$ jobs`
	`[2]  + suspended  xroger`
	`[3]  - running    xhextris`
	`$ fg %2`
	`[2]  + continued xroger`

nohup	**nohup** *commande* **&** Permet de lancer une commande en arrière plan qui ne s'achèvera pas après que l'utilisateur soit délogué. ***Exemple*** `nohup octave < calcul.m > resultat.txt &`
pwd	**pwd** Affiche le chemin complet du répertoire de travail courant.

read	**read** *var1* [*var2* . . .] Lit une ligne du canal d'entrée standard et la stocke dans les variables *var1*, *var2* . . .. La dernière variable contient la fin de la ligne.
return	**return** [*n*] Permet de sortir d'une fonction shell avec le statut *n*. *n* peut valoir 0 (succès) ou tout autre valeur non nulle (échec). Si *n* n'est pas précisé, **exit** renvoie le statut de la dernière commande exécutée.
shift	**shift** [*n*] Passe en première position l'argument *n* (par défaut 2). Ceci est utile pour les boucles **while**.
stop	**stop** [*tâche*] Stoppe une *tâche* référencée par son numéro (voir la commande **jobs**) exécutée en arrière plan. C'est le complément de CONTROL-Z et de **suspend**. Sans paramètre, **stop** agit sur le dernier processus lancé en arrière plan.
suspend	**suspend** Stoppe la tâche en cour. Identique au CONTROL-Z.
test	**test** *condition* **[** *condition* **]** Evalue une *condition* et renvoie un statut 0 si le résultat est vrai sinon renvoie une valeur non nulle. Ces conditions permettent de tester des fichiers, des chaînes de caractères ou des nombres entiers. ***Condition sur les fichiers*** **-b** *f* — *f* existe et est un périphérique bloc. **-c** *f* — *f* existe et est un périphérique caractère. **-d** *f* — *f* existe et est un répertoire. **-f** *f* — *f* existe et est un fichier normal. **-g** *f* — *f* existe et possède les droits **g=s**. **-k** *f* — *f* existe et possède le « sticky bit ». **-p** *f* — *f* existe et est un « pipe ». **-r** *f* — *f* existe et est lisible. **-s** *f* — *f* existe et n'est pas vide. **-u** *f* — *f* existe et possède les droits **u=s**. **-w** *f* — *f* existe et est écrasable. **-x** *f* — *f* existe et est exécutable. ***Condition sur les chaînes de caractères*** **-n** *s1* — La chaîne *s1* est non vide. **-z** *s1* — La chaîne *s1* est vide. *s1* **=** *s2* — *s1* et *s2* sont identiques. *s1* **!=** *s2* — *s1* et *s2* sont différentes.

	Condition sur les nombres entiers
	n1 **-eq** *n2* — *n1* est égal à *n2*.
	n1 **-ge** *n2* — *n1* est supérieur ou égal à *n2*.
	n1 **-gt** *n2* — *n1* est strictement supérieur à *n2*.
	n1 **-le** *n2* — *n1* est inférieur ou égal à *n2*.
	n1 **-lt** *n2* — *n1* est strictement inférieur à *n2*.
	n1 **-ne** *n2* — *n1* n'est pas égal à *n2*.
	Couplage de conditions
	(*cnd1*) — Vraie si *cnd1* est vraie. Ceci est utile pour grouper des conditions.
	! *cnd1* — Vraie si *cnd1* est fausse.
	cnd1 **-a** *cnd2* — Vraie si *cnd1* et *cnd2* sont vraies.
	cnd1 **-o** *cnd2* — Vraie si *cnd1* ou *cnd2* sont vraies.
	Exemples
	`while test $# -gt 0` — Tant qu'il y a des arguments...
	`while [ -n "$1" ]` — Tant qu'il n'y a pas d'argument vide...
	`if [ -d Td2 ]` — Si **Td2** est un répertoire...
	`if [ "$t" != "n" ]` — si **t** n'est pas `n`...

type	**type** *commande*
	Indique si une *commande* est une commande shell, une commande Unix, un alias ou une fonction définie par l'utilisateur.
	Exemples
	`$ type mv echo`
	`mv is /bin/mv`
	`echo is a shell builtin`

unalias	**unalias** *nom* Efface l'alias *nom* (fonction Korn Shell). Voir **alias**.
unset	**unset** *var* Efface la définition de la variable *var*.
until	**until** *condition* **do** *commandes* **done** Juqu'à ce que la *condition* soit vraie, exécute l'ensemble des *commandes*. Les conditions s'écrivent grâce à la commande shell **test**.

while	**while** *condition* **do** *commandes* **done** Tant que la *condition* est vraie, exécute l'ensemble des *commandes*. Les conditions s'écrivent grâce à la commande shell **test**.

Annexe A

Références et licence

La dernière version est disposible à l'adresse http ://www.softndesign.org/ ?page=manuels/unix-2.php. Des versions Postscript et PDF y sont aussi disponibles.
Ce document est distribué sous licence GPL. Vous êtes autorisé à le copier et/ou le redistribuer intégralement ou en partie, à la seule condition que cette mention y reste présente et conformément aux dispositions de la Licence de Documentation Libre GNU telle que publiée par la Free Software Foundation ; version 1.2 de la licence, ou encore (à votre choix) toute version ultérieure.

Index