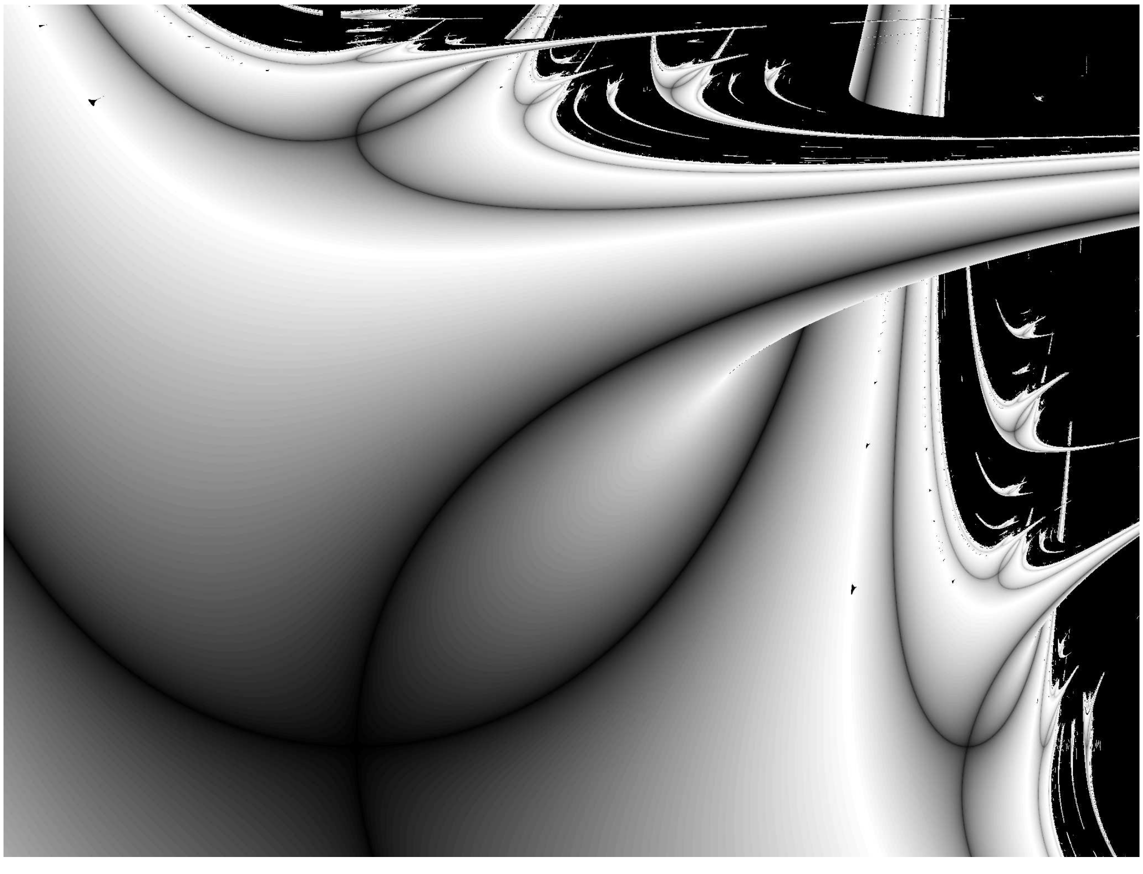
Unix

Comment débuter sur un vrai système



Laurent Mazet mazet@softndesign.org

Table des matières

Chapitre 1

Avant propos

Ce très court ouvrage n'est pas un manuel sur **Unix** ni même un livre de référence. Il ne s'agit que d'une énumération des quelques commandes qu'il est indispensable de connaître avant d'essayer de développer. C'est un peu le b-a-ba mais que cela ne vous empêche pas de feuilleter de véritables livres détaillant le fonctionnement d'**Unix**, vous les trouverez dans toutes les bonnes crèmeries voire même à la bibliothèque universitaire.

1.1 Une petite idée d'Unix

Unix est un système multi-utilisateurs et multi-processus. C'est-à-dire que le système accepte que plusieurs utilisateurs travaillent en même temps sur une station (via des terminaux textes ou graphiques). De plus chaque utilisateur peut effectuer plusieurs tâches en même temps. Il vous sera donc possible de compiler votre projet **Fortran** pendant que vous lirez votre courrier électronique sans pour autant être un génie de l'informatique.

1.2 Le terminal et vous

Vu qu'un ordinateur manque cruellement de conversation, c'est à vous de commencer. Tout d'abord, il vous faut demander au terminal de se connecter sur la station sur laquelle vous voulez travailler puis répondre gentiment aux deux questions qu'elle va vous poser (votre nom d'accès (**login**) et votre code d'accès (**password**)).
Exemple :

```
Local> connect beyrouth
Connected to beyrouth.
Escape character is '^]'.

Red Hat Linux release 4.1 (Vanderbilt)
Kernel 2.0.27 on an i386
login: mazet
Password:
beyrouth ~ $
```

Le « **$** » vous indique que la station attend vos ordres.

1.3 La station et vous

Comme **Unix** est multi-utilisateurs, il est indispensable que chacun ait un répertoire qui lui soit propre pour stocker ses fichiers et ses sous-répertoires. Ainsi, quand vous vous loguez, vous êtes placé directement dans votre répertoire « racine » (c'est-à-dire votre répertoire principal). La figure 1.3 illustre une arborescence classique.

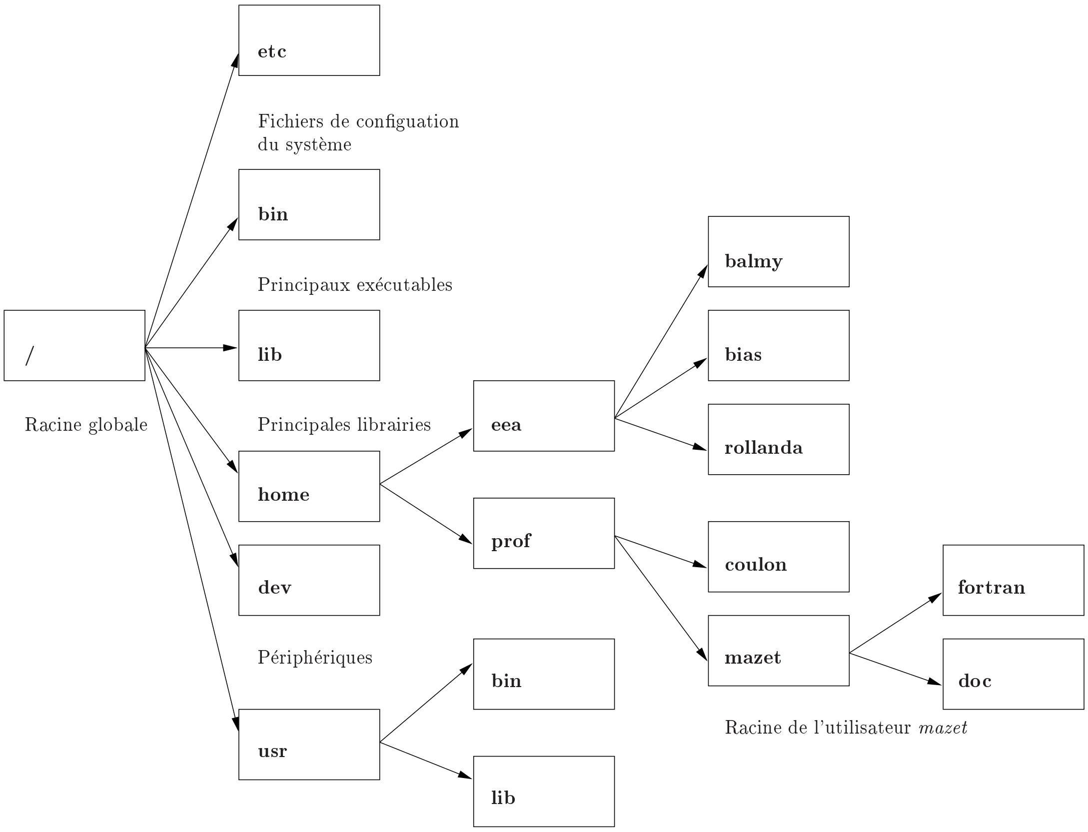


FIG. 1.1 – Un exemple d'arborescence

D'autre part, vous n'êtes propriétaire que de vos fichiers. Il vous est donc impossible de créer ou de modifier des fichiers qui ne vous appartiennent pas. Veuillez donc travailler uniquement dans votre répertoire.

Chapitre 2

Arborescence

2.1 ls

La commande **ls** liste le contenu du répertoire courant.
Exemple :

```
beyrouth ~/doc/unix $ ls
#unix.tex#  unix.aux    unix.log    unix.tex    unix.toc
toto        unix.dvi    unix.ps     unix.tex~
beyrouth ~/doc/unix $
```

Il est possible d'obtenir des niveaux de détails plus importants grâce aux options suivantes :
– **ls -s** liste les fichiers ainsi que leurs tailles en kilo-octets.
– **ls -l** liste l'ensemble des informations inhérentes à chacun des fichiers.
– **ls -a** liste tous les fichiers (même ceux commençant par « . »).
Exemples :

```
beyrouth ~/doc/unix $ ls -s
total 100
   1 toto         28 unix.dvi     48 unix.ps       5 unix.tex~
   6 unix.aux      2 unix.log      7 unix.tex      3 unix.toc
beyrouth ~/doc/unix $ ls -l
total 98
-rw-r--r--   1 mazet    users           5 Sep 30 19:42 toto
-rw-r--r--   1 mazet    users        5297 Sep 30 19:33 unix.aux
-rw-r--r--   1 mazet    users       25960 Sep 30 19:33 unix.dvi
-rw-r--r--   1 mazet    users        1204 Sep 30 19:33 unix.log
-rw-r--r--   1 mazet    users       47775 Sep 30 01:46 unix.ps
-rw-r--r--   1 mazet    users        5559 Sep 30 19:44 unix.tex
-rw-r--r--   1 mazet    users        5093 Sep 30 19:30 unix.tex~
-rw-r--r--   1 mazet    users        2449 Sep 30 19:33 unix.toc
beyrouth ~/doc/unix $ ls -a
./         .titi      unix.aux   unix.log   unix.tex   unix.toc
../        toto       unix.dvi   unix.ps    unix.tex~
beyrouth ~/doc/unix $
```

2.2 Les répertoires . et ..

Tous les répertoires sous **Unix** contiennent ces deux répertoires. Le répertoire « **.** » est le répertoire dans lequel vous vous trouvez. Il est appelé répertoire courant. Le répertoire « **..** » est le répertoire qui contient le répertoire courant. Il est appelé répertoire supérieur.

2.3 cd

La commande **cd** permet la navigation dans l'arborescence des fichiers.
Exemple :

```
beyrouth ~/ $ cd /
beyrouth / $ cd bin
beyrouth /bin/ $ cd ..
beyrouth / $ cd /usr/local/
beyrouth /usr/local/ $ cd -
beyrouth / $ cd ~
beyrouth ~/ $ cd .
beyrouth ~/ $
```

- **cd /** va dans le répertoire **/** (c'est-à-dire la racine globale de l'arborescence).
- **cd bin** va dans le répertoire **bin**
- **cd ..** va dans le répertoire supérieur.
- **cd /usr/local/** va dans le répertoire **/usr/local/**
- **cd -** revient dans le répertoire d'où l'on vient.
- **cd ˜** se place sur le répertoire racine de l'utilisateur.
- **cd .** va dans le répertoire courant (utile pour la copie de fichiers vers le répertoire courant).

2.4 mkdir

La commande **mkdir** crée un répertoire dans le répertoire courant.
Exemple :

```
beyrouth ~/ $ ls
beyrouth ~/ $ mkdir truc
beyrouth ~/ $ ls
truc/
beyrouth ~/ $ cd truc
beyrouth ~/truc $ ls -a
./     ../
beyrouth ~/truc $
```

2.5 rmdir

La commande **rmdir** efface un répertoire vide (ne contenant pas de fichier).
Exemple :

```
beyrouth ~/ $ ls
truc/
beyrouth ~/ $ rmdir truc
beyrouth ~/ $ ls
beyrouth ~/ $
```

2.6 cp

La commande **cp** permet de copier un fichier *source* vers un fichier *destination*.
Exemple :

```
beyrouth ~/ $ ls
toto
beyrouth ~/ $ cp toto titi
beyrouth ~/ $ ls
```

```
toto   titi
beyrouth ~/ $ mkdir truc
beyrouth ~/ $ cd truc
beyrouth ~/truc/ $ cp ../toto .
beyrouth ~/truc/ $ ls
toto
beyrouth ~/truc/ $
```

2.7 rm

La commande **rm** efface un fichier.
Exemple :

```
beyrouth ~/ $ ls
toto titi
beyrouth ~/ $ rm toto
beyrouth ~/ $ ls
titi
beyrouth ~/ $
```

2.8 mv

La commande **mv** permet de déplacer un fichier ou un répertoire *source* vers un fichier ou un répertoire *destination*.
Exemple :

```
beyrouth ~/ $ ls
toto titi
beyrouth ~/ $ mv titi toto.bis
beyrouth ~/ $ ls
toto toto.bis
beyrouth ~/ $
```

2.9 pwd

La commande **pwd** affiche le chemin complet du répertoire courant.
Exemple :

```
beyrouth ~/truc/ $ pwd
/home/mazet/truc
beyrouth ~/truc/ $
```

2.10 *, ? et []

Ces caractères permettent de sélectionner un ensemble de fichiers suivant des critères spécifiques.
Le caractère « ***** » indique que toute chaîne de caractères peut se substituer à l'étoile.
Exemple :

```
beyrouth ~/c/convert $ ls con*
convert.c  convert.o
beyrouth ~/c/convert $
```

Le caractère « **?**» indique que tout caractère peut se substituer au point d'interrogation.
Exemple :

```
beyrouth ~/c/convert $ ls convert.?
convert.c  convert.o
beyrouth ~/c/convert $
```

Chacun des caractères entre « **[]** » peut se substituer à l'ensemble entre crochets.
Exemple :

```
beyrouth ~/c/convert $ ls
convert.a  convert.c  convert.ac  convert.z
beyrouth ~/c/convert $ ls convert.[acb]
convert.a  convert.c
beyrouth ~/c/convert $
```

Chapitre 3

Droits et propriétés

3.1 who

La commande **who** affiche la liste des utilisateurs logués sur la station.
Exemple :

```
beyrouth ~/ $  who
mazet    ttyp1    Sep 29 21:53 (:0.0)
beyrouth ~/ $
```

3.2 finger

La commande **finger** recherche des informations au sujet d'un utilisateur sur une station connectée au réseau.
Exemple :

```
beyrouth ~/ $ finger cazenave@balthazar.cybersoft.org
[balthazar.cybersoft.org]
Login: cazenave                        Name: Thierry Cazenave-Lavie
Directory: /home/cazenave              Shell: /bin/zsh
Never logged in.
No mail.
No Plan.
beyrouth ~/$
```

3.3 chmod

La commande **chmod** change les droits d'accès d'un fichier ou d'un répertoire. Ces droits sont en fait des droits d'écriture, de lecture ou d'exécution (ce dernier n'a de sens que pour les fichiers exécutables ainsi que les répertoires). Il y a trois catégories de droits :

– Ceux que le propriétaire du fichier s'accorde à lui-même.
– Ceux qu'il accorde aux personnes du même groupe que lui.
– Ceux qu'il accorde aux autres.

On peut rendre un fichier visible par tous, exécutable par toute personne du groupe et modifiable par soi uniquement.
Exemple :

```
beyrouth ~/doc/unix $ ls -l toto
----------   1 mazet    users           5 Sep 30 19:42 toto
beyrouth ~/doc/unix $ chmod u+rxw toto
beyrouth ~/doc/unix $ ls -l toto
```

```
-rwx------   1 mazet    users           5 Sep 30 19:42 toto*
beyrouth ~/doc/unix $ chmod g+rx toto
beyrouth ~/doc/unix $ ls -l toto
-rwxr-x---   1 mazet    users           5 Sep 30 19:42 toto*
beyrouth ~/doc/unix $ chmod o+r toto
beyrouth ~/doc/unix $ ls -l toto
-rwxr-xr--   1 mazet    users           5 Sep 30 19:42 toto*
beyrouth ~/doc/unix $ chmod a-x toto
beyrouth ~/doc/unix $ ls -l toto
-rw-r--r--   1 mazet    users           5 Sep 30 19:42 toto
beyrouth ~/doc/unix $
```

- **chmod u+rxw toto** ajoute les droits de lecture (**r**), d'exécution (**x**) et d'écriture (**w**) à son propriétaire (**u**).
- **chmod g+rx toto** ajoute les droits de lecture (**r**) et d'exécution (**x**) au groupe du propriétaire (**g**).
- **chmod o+r toto** ajoute le droit de lecture (**r**) aux autres (**o**).
- **chmod a-x toto** retire le droit d'exécution à tous (**a=ugo**).

Chapitre 4

Gestion de processus

4.1 &, ; et |

Le caractère « **&** » permet de lancer un processus en tâche de fond et ainsi de récupérer le contrôle du shell pendant que la station travaille.
Exemple :

```
beyrouth ~/doc/unix $ gcc toto.c -c &
[2] 259
beyrouth ~/doc/unix $
```

Le caractère « **;** » permet de lancer plusieurs commandes les unes à la suite des autres.
Exemple :

```
beyrouth ~/c/convert $ ls ; gcc -Wall convert.c -c ; ls
convert.c
convert.c  convert.o
beyrouth ~/c/convert $
```

Le caractère « **|** » permet de relier deux commandes entre elles (la sortie de l'une devient l'entrée de l'autre).
Exemple :

```
beyrouth ~/c/convert $ cat convert.c | grep newC
  int nbCode, newCode = 0;
  while (!newCode) {
    newCode = 1;
      newCode = 0;
  char newC;
  char newCode[MAX_CODE];
    newC = (int) strtol (c, &c, 16);
      sprintf (newCode, "%c", (int) strtol (c+1, NULL, 16));
      (*l) = add(*l, newC, newCode);
      (*l) = add(*l, newC, c+1);
beyrouth ~/c/convert $
```

Ici la sortie de **cat** (affichage d'un fichier) est envoyée à **grep** (recherche d'une chaîne de caractères).

4.2 ps

La commande **ps** affiche la liste des processus en cours.
Exemple :

```
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:01 zsh
  163  p1 S N  0:02 xdvi unix.dvi
  166  p1 R    0:00 ps
beyrouth ~/doc/unix $
```

– `PID` : numéro d'identification du processus.
– `TIME` : temps de vie du processus.
– `COMMAND` : commande associée au processus.

4.3 kill

La commande **kill** transmet des signaux à des processus (via leur `PID`) travaillant en arrière-plan. Le plus connu (et sûrement le plus utilisé) de ces signaux est le signal **9** appelé *KILL*. Il arrête le processus.
Exemple :

```
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:01 zsh
  163  p1 S N  0:02 xdvi unix.dvi
  166  p1 R    0:00 ps
beyrouth ~/doc/unix $ kill -9 163
[1]  + 163 killed     xdvi unix.dvi
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:03 zsh
  272  p1 R    0:00 ps
beyrouth ~/doc/unix $
```

4.4 ˆC

La combinaison de touche **ˆC** (`Control C`) arrête toute commande en cours d'exécution.
Exemple :

```
beyrouth ~/doc/unix $ gcc toto.c -c
[2]    287 exit 1     gcc toto.c -c
beyrouth ~/doc/unix $
```

4.5 ˆZ

La combinaison de touche **ˆZ** (`Control F`) stoppe toute commande en cours d'exécution. Le processus n'est plus actif mais peut reprendre si on lui envoie le bon signal.
Exemple :

```
beyrouth ~/doc/unix $ gcc toto.c -c

zsh: 295 suspended  gcc toto.c -c
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:03 zsh
  276  p1 S N  0:09 xdvi unix.dvi
  295  p1 T    0:00 gcc toto.c -c
  296  p1 R    0:00 ps
beyrouth ~/doc/unix $ kill -9 295
```

```
[2]  + 295 killed     gcc toto.c -c
beyrouth ~/doc/unix $
```

4.6 fg

La commande **fg** bascule en premier plan (à la place du shell) le dernier processus stoppé ou lancé en arrière-plan.
Exemple :

```
beyrouth ~/doc/unix $ gcc toto.c -c

zsh: 297 suspended  gcc toto.c -c
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:03 zsh
  276  p1 S N  0:09 xdvi unix.dvi
  297  p1 T    0:00 gcc toto.c -c
  298  p1 R    0:00 ps
beyrouth ~/doc/unix $ fg
[2]  - continued  gcc toto.c -c
gcc: toto.c: No such file or directory
gcc: No input files
beyrouth ~/doc/unix $
```

4.7 bg

La commande **bg** bascule en arrière-plan le dernier processus stoppé.
Exemple :

```
beyrouth ~/doc/unix $ xdvi unix.dvi

zsh: 300 suspended  xdvi unix.dvi
beyrouth ~/doc/unix $ ps
  PID TTY STAT  TIME COMMAND
  150  p1 S    0:03 zsh
  300  p1 T    0:01 xdvi unix.dvi
  301  p1 R    0:00 ps
beyrouth ~/doc/unix $ bg
[1]  - continued  xdvi unix.dvi
beyrouth ~/doc/unix $
```

Chapitre 5

Edition

5.1 cat

La commande **cat** affiche un fichier.
Exemple :

```
beyrouth ~/doc/unix $ cat toto
fe
d
beyrouth ~/doc/unix $
```

5.2 more

La commande **more** formate l'affichage d'un fichier en page-écran. Il est ensuite possible de se déplacer à l'intérieur.

- `Return` avance d'une ligne vers la fin du fichier.
- `Space` avance d'une page vers la fin du fichier.
- `b` avance d'une page vers le début du fichier.
- `q` quitte **more**.

5.3 grep

La commande **grep** recherche un chaîne de caractères dans un fichier et affiche les lignes où cette chaîne apparaît.
Exemple :

```
beyrouth ~/c/xtetris $ grep game tetris.c
#include "game.h"
    fprintf (stderr, INC " freeware game\n");
    fprintf (stderr, INC " freeware game\n");
  createFrameOut (contexte, misc->game.begin_x-misc->thickness,
                  misc->game.begin_y-misc->thickness,
                  misc->game.end_x+misc->thickness,
                  misc->game.end_y+misc->thickness, misc->thickness);
beyrouth ~/c/xtetris $
```

5.4 vi

La commande **vi** est un éditeur de texte. Elle permet de créer ou de modifier des fichiers. Attention le mode d'emploi de **vi** est loin d'être intuitif, mémorisez en les concepts de base avant d'être perdu...
Exemple :

```
beyrouth ~/doc/unix $ vi toto

~
~
~
"toto" [New File]
```

Si vous voulez insérer une phrase ou plusieurs phrases, appuyez sur la touche **i**. Lorsque vous avez fini, appuyez sur la touche **ESC** (*Escape*). Vous pouvez vous déplacer de droite à gauche avec la touche **h**, de gauche à droite avec **l**, de haut en bas avec **j** et de bas en haut avec **k**. Pour effacer un caractère, appuyez sur la touche **x** et pour effacer une ligne entière, appuyez deux fois sur la touche **d**. Pour sauvegarder, tapez **:w** puis **Return**. Enfin pour quitter ce superbe éditeur, tapez **:q** puis **Return**.
Le tableau 5.4 récapitule l'ensemble des commandes que vous avez à connaître.

Commande	Touche(s)	Fin
Insertion sous le curseur	**i**	**ESC**
Insertion en début de ligne	**I**	**ESC**
Ajout devant le curseur	**a**	**ESC**
Ajout en fin de ligne	**A**	**ESC**
Rajout d'une ligne sous le curseur	**o**	**ESC**
Rajout d'une ligne au-dessus du curseur	**O**	**ESC**
Déplacement vers la droite	**l** ou $\rightarrow$	
Déplacement vers la gauche	**h** ou $\leftarrow$	
Déplacement vers le haut	**j** ou $\uparrow$	
Déplacement vers le bas	**k** ou $\downarrow$	
Déplacement de *4* lignes vers le bas	**4k** ou **4**$\downarrow$	
Placement à la ligne *23*	**:23** puis **Return**	
Effacement d'un caractère	**x**	
Effacement de *5* caractères	**5x**	
Effacement d'une ligne	**dd**	
Effacement de *3* lignes	**3dd**	
Sauvegarde d'un fichier	**:w** puis **Return**	
Arrêt de l'édition	**:q** puis **Return**	
Sauvegarde et arrêt	**:wq** puis **Return**	
Ajout du fichier *truc*	**:r truc** puis **Return**	
Forçage d'une commande	**!**	
Copie d'un bloc de *6* lignes	**6yy**	
Collage d'un bloc au-dessous du curseur	**p**	
Collage d'un bloc au-dessus du curseur	**P**	
Recherche de la chaîne *toto*	**/toto** puis **Return**	
Suite de la recherche vers la fin	**n**	
Suite de la recherche vers le début	**N**	

TAB. 5.1 – Commandes **vi**

5.5 joe

La commande **joe** est un autre éditeur de textes. Celui-ci est beaucoup plus simple d'utilisation que **vi**. Tapez **ˆK-H** (`Control K` puis `H`) puis lisez l'aide.

Chapitre 6

Gestion du compte

6.1 passwd

La commande **passwd** permet de changer son password.
Exemple :

```
beyrouth ~/doc/unix $ passwd
Password:
New password:
New password (again):
Password changed.
beyrouth ~/doc/unix $
```

6.2 chfn

La commande **chfn** permet de changer ses informations.
Exemple :

```
beyrouth ~/doc/unix $ chfn
Changing finger information for mazet.
Password:
Name [Laurent Mazet]:
Office []: J24
Office Phone []: 72.90
Home Phone []:

Finger information changed.
beyrouth ~/doc/unix $
```

6.3 chsh

La commande **chsh** permet de changer le shell (interpréteur de commande) de login d'un utilisateur.
Exemple :

```
beyrouth ~/doc/unix $ chsh
Changing shell for mazet.
Password:
New shell [/bin/zsh]: /bin/csh
Shell changed.
beyrouth ~/doc/unix $
```

Il est possible de connaître la liste des shells disponibles par la commande **chsh -l**.
Exemple :

```
beyrouth ~/doc/unix $ chsh -l
/bin/bsh
/bin/bash
/bin/sh
/bin/ash
/bin/zsh
beyrouth ~/doc/unix $
```

Chapitre 7

Compléments

7.1 man

La commande **man** affiche une aide relative à une commande.
Exemple :

```
beyrouth ~/doc/unix $ man man

man(1)                                                        man(1)

NAME
       man - format and display the on-line manual pages
       manpath - determine user's search path for man pages

SYNOPSIS
       man  [-adfhkKtwW] [-m system] [-p string] [-C config_file]
       [-M path] [-P pager] [-S section_list] [section] name  ...

DESCRIPTION
       man  formats  and displays the on-line manual pages.  This
       version knows about the MANPATH and (MAN)PAGER environment
       variables, so you can have your own set(s) of personal man
       pages and choose whatever program you like to display  the
       formatted  pages.  If section is specified, man only looks
       in that section of the manual.  You may also  specify  the
       order to search the sections for entries and which prepro-
       cessors to run  on  the  source  files  via  command  line
       options  or  environment  variables.  If name contains a /
       then it is first tried as a filename, so that you  can  do
       man ./foo.5 or even man /cd/foo/bar.1.gz.
:
beyrouth ~/doc/unix $
```

C'est la commande la plus importante que vous ayez à connaître.

7.2 mail

La commande **mail** poste des courriers électroniques.
Exemple :

```
beyrouth ~/doc/unix $ mail
To: cazenave@cybersoft.org
```

```
Subjet: About Emacs...
[. to end]
Salut,

Je t'envoie mon .emacs des que possible.
A+

       ;-)
.
[mail posted]
beyrouth ~/doc/unix $
```

7.3 alias

La commande **alias** crée des alias.
Exemple :

```
beyrouth ~/doc/unix $ lss
zsh: command not found: lss
beyrouth ~/doc/unix $ alias lss='ls -s'
beyrouth ~/doc/unix $ lss
total 130
  17 #unix.tex#    33 unix.dvi      15 unix.tex
   1 toto           2 unix.log       7 unix.tex~
   5 unix.aux      48 unix.ps        2 unix.toc
beyrouth ~/doc/unix $
```

Annexe A

Références et licence

La dernière version est disponible à l'adresse http ://www.softndesign.org/ ?page=manuels/unix-1.php. Des versions Postscript et PDF y sont aussi disponibles.