Linux,

commandes, outils

et autres “sources” d’amusements

3 novembre 1999

Merci à zecastor de m'avoir supporté…

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Bases et commandes utiles

Attention ! Ne pas arrêter le système n'importe comment ! ! !

Voir section 9.13 page 240.

Unix (et Linux) fait la différence entre les majuscules et les minuscules, il faut donc faire attention lorsque on tape le nom d'un fichier ou l'option d'une commande. Pour chacune des commandes, on peut en connaître en général les options en tapant le nom de la commande suivit de *-h* ou *–help* suivant les cas.

1.1 Redirections

`> / >>` : redirection de la sortie standard (l'écran) vers un fichier / id sans écraser ;

`>& / &>` : redirige la sortie standard et la sortie d'erreurs ;

`2>` : redirige seulement les messages d'erreur ;

`2>&1` : permet de rassembler en un seul canal la sortie standard et la sortie d'erreurs standard.

`(l; ps; who) > liste` : permet de rediriger le résultat de la suite de commandes vers le fichier `liste`. En l'absence de parenthèses, seule le résultat de la dernière commande serait redirigée.

`<` : redirection en entrée. Certaines commandes, comme `tr`, ont besoin d'une redirection en entrée pour lire les données dans un fichier, sans quoi les données seront directement lues sur l'entrée standard (le clavier).

On peut ouvrir un nouveau canal :
`exec 5> Liste`
et rediriger des commandes vers ce nouveau canal :
`ls -l >&5; ps -ef >&5; who >&5`
ou
`(ls -l; ps -ef; who) >&5`
et on peut rediriger la sortie standard vers ce nouveau canal, et la sortie d'erreur standard vers ce même canal :
`(ls -l; ps -ef; who) 1>&5 2>&1`

1.2 Utilisation de tubes

`prog1|prog2` : permet d'envoyer en entré de `prog2` la sortie de `prog1` ; par exemple :

```
du|sort -rn|less
```

1.3 Filtre grep

`grep` est un filtre. Il peut trouver un mot dans un fichier, par exemple :

```
grep malloc *.c
```

cherche la chaine de caractères *malloc* dans tous les fichiers dont le nom se termine par *.c* (`*.c`).

On peut insérer une variable dans le critère de recherche (utile pour les scripts shells) :

```
grep "^[^:]*:[^:]*:$1:" /etc/passwd
grep "^$utilisateur
grep "^$utilisateur" > /dev/null 2>&1
```

ou utiliser un tube pour filtrer la sortie d'une commande :

```
locate Quick | grep dosemu
```

Pour élaborer un modèle de recherche, on dispose d'un certain nombre de caractères spéciaux (tableau 1.1).

Caractère	**Signification**
[...]	Plage de caractères permis.
`[^...]`	Plage de caractères interdits.
`^`	Début de ligne.
`.`	Un caractère quelconque, y compris un espace.
`*`	Caractère de répétition, agit sur le caractère placé avant l'étoile. Accepte également l'absence du caractére placé devant lui.
`$`	Fin de ligne.
`\{...\}`	Répétition.
`\{Nombre\}`	Répétition de *Nombre* exactement.
`\{Nombre,\}`	Répétition de *Nombre* au minimum.
`\{Nombre1 Nombre2\}`	Répétition de *Nombre1* à *Nombre2*.

TAB. 1.1 – Caractères spéciaux servant de modèle pour **grep**.

Le tableau 7.1 page 174 présente d'autres modèles pouvant être utilisés.

Par ailleurs, plusieurs options peuvent être utilisées. Voici quelque options utiles de la commande `grep` (tableau 1.2) :

Option	**Signification**
-c	Nombre de ligne trouvées (sans les afficher).
-i	Ne fait pas la différence entre majuscule et minuscule.
-n	Affiche le numero de la ligne.
-l	Affiche le nom du fichier contenant la ligne (et pas la ligne).
-v	Affiche toutes les lignes qui ne contiennent pas le mot en question.

TAB. 1.2 – Options courantes de la commande **grep**.

Il existent deux autres commandes qui élargissent les possibilitées de la commande `grep` : `egrep` et `fgrep`.

La commande `egrep` permet d'utiliser un fichier contenant un ou des critères de recherche avec l'option *-f*. Si plusieurs critères sont spécifiés, `egrep` cherchera pour chacun de ces critères (OU logique). Il existe par ailleurs plusieurs options qui peuvent simplifier les critères de recherche (tableau 1.3).

Caractère	**Signification**
+	Caractère de répétition. Le caractère placé devant ce signe doit exister au minimum une fois.
?	Caractère de répétition. Le caractère placé devant peut apparaître une fois ou pas du tout.
(a|b)	L'une ou l'autre des expressions sont autorisées.
(...)	Permettent de grouper des critères partiels.

TAB. 1.3 – Caractères spéciaux servant de modèle pour **fgrep**.

Voici quelque exemples d'utilisation :

```
[a-z]+
```

cherche toutes les lignes contenant au minimum une lettre en minuscule. Le critère avec `grep` aurait été `[a-z][a-z]*`.

```
^[0-9]\{3\}$
```

cherche toutes les lignes contenant uniquement un nombre à 3 chiffres.

```
:[0-9]\{2,\}:
```

cherche toutes les lignes contenant des nombres de minimum 2 chiffres avant les deux points (" **:**").

```
^[0-9]\{1,5\}:
```

cherche toutes les lignes commencant par des nombres de minimum 1 à 5 chiffres suivits par deux points (" **:**").

```
(une|deux) fois
```

cherche toutes les lignes contenant la chaîne "une fois" ou "deux fois".

La commande `fgrep` effectue une recherche plus rapide, mais ne reconnaît pas les caractères spéciaux. Comme pour `egrep`, on peut spécifier un fichier contenant un ou des critères de recherche avec l'option *-f*. Un tel fichier pourra par exemple contenir :

```
une fois
deux fois
```

et les lignes contenant l'un ou l'autre de ces textes seront recherchées.

1.4 La commande cut

La commande `cut` permet d'afficher des zones spécifiques d'un fichier. Par exemple :

```
cut -c1 /etc/passwd
```

affichera la première colonne du fichier `/etc/passwd`. Il existe d'autres spécifications (tableau 1.4) :

Option	**Signification**
-c1-5	Permet de sélectionner les colonnes 1 à 5.
-c14-	Permet de sélectionner de la colonne 14 à la dernière.
-c1-3,14-18	Permet de spécifier plusieurs plages de colonnes.

TAB. 1.4 – Spécifications de la commande **cut**.

On peut également spécifier un *séparateur de champs* avec l'option *-d*. Par exemple :

```
cut -d: -f6 /etc/passwd
```

affichera le 6^{eme} champ du fichier `/etc/passwd`, dont le séparateur de champs est le caractére double point (" **:**").

1.5 Trier avec sort

Le programme `sort` permet de trier les lignes d'un fichier. Les caractères "+" et "-" permettent de spécifier de quelle colonne à quelle colonne le tri doit s'effectuer (1^{ere} colonne pour 0, 2^{eme} colonne pour 1...) :

```
sort +1 -2 /etc/passwd
```

Si on spécifie plusieurs critères, le tri se fera d'abord sur le premier champ, puis sur le second si le tri sur le premier champ n'a pas suffit à départager certaines lignes, et ainsi de suite...Il existe diverses options (tableau 1.5) :

On peut spécifier la recherche sur un caractère situé à une position particulière, par exemple à la 2^{eme} position du 6^{eme} champ :

Option	Signification
-b	Saute les colonnes constituées de blancs.
-d	Trie de type dictionnaire.
-n	Trie par ordre numérique.
-f	Aucune différentiation n'est faite entre minuscules et majuscules.
-b	Ignore les espaces placés en début de champ.
-r	Trie inverse.
-M	Trie chronologiquement les mois.
-t :	Trie suivants les champs séparés par les caractères deux points (" **:**").

Tab. 1.5 – Options de la commande **sort**.

```
sort -t: +5.1 /etc/passwd
```

Pour plusieurs critères de recherche, il faut spécifier derrière *chaque* champ le type de tri à mettre en oeuvre pour ce critère. Par exemple :

```
sort -t: +0d -1 +2nr -3
```

triera le 1^{er} champ par ordre *dictionnaire*, et le 3^{eme} champ par ordre *numérique inverse*, et

```
sort -t: +4.3n -4.5 +4.0n -4.2
```

triera du 4^{eme} au 6^{eme} caractère du 5^{eme} champ par ordre *numérique*, et du 1^{er} au 3^{eme} caractère du 5^{eme} champ par ordre *numérique*, si le premier tri s'est avéré insuffisant.

Bien sur, on peut combiner les commandes `cut` et `sort`. Par exemple :

```
cut -d: -f3 /etc/passwd | sort -n > Nombres
```

1.6 Conversion de chaînes de caractères avec tr

La commande `tr` permet de convertir une chaîne de caractères. Par exemple :

```
tr "[A-Z]" "[a-z]" < /etc/passwd
```

ou encore

```
tr  "ai" "as" < /etc/passwd | less
```

Option	Signification
-c	Tous les caractères qui ne sont pas spécifiés dans la première chaîne sont convertis selon les caractères de la seconde.
-d	Efface le caractère spécifié.
-s	Si le caractère spécifié se répéte plusieurs fois de suite, il est réduit à une seule unité.

Tab. 1.6 – Options de la commande **tr**.

Le résultat s'affiche à l'écran. Le fichier ne sera pas modifié avec la commande précédente. Il existe plusieurs options, telles que (tableau 1.6) :

Voici plusieurs exemples illustrant l'utilisation de ces options :

- pour convertir les fichiers Dos au format UNIX (sans retour chariot) :

  ```
  tr -d '\015' < fichier.dos > fichier.linux
  ```

- pour effacer tous les "**0**" du fichier `/etc/passwd` :

  ```
  tr -d "0" < /etc/passwd
  ```

- pour réduire toute suite de "**0**" à un seul "**0**" dans le fichier `/etc/passwd` :

  ```
  tr -s "0" < /etc/passwd
  ```

- pour réduire toute suite de caractères non comprise (par ordre alphabétique) entre "**a**" et"**s**" à un seul dans le fichier `/etc/passwd` :

  ```
  tr -cs "[a-s]" < /etc/passwd | less
  ```

  Par exemple, "**root**" contient deux fois "**o**", mais "**o**" est compris entre "**a**" et "**s**", donc "**root**" ne sera pas changé. Par contre, "**uucp**" contient deux fois "**u**", et sera donc convertit en "**ucp**".

1.7 Duplication d'un flux de données avec tee

La commande `tee` permet de rediriger une commande tout en la transmettant à un tube. Par exemple :

```
cut -d: -f1,3 /etc/passwd | tee Liste.txt | sort -t: +0 -1
```

affichera à l'écran la liste des utilisateurs suivit de leurs numéros d'utilisateurs (commande *cut*), triée par ordre alphabétique (commande *sort*), tandis que la liste *non triée* sera redirigée vers le fichier `Liste.txt` (commande *tee*). Cette redirection écrasera le contenu du fichier. Avec l'option *-a* , la commande `tee` n' écrasera pas l'ancien fichier.

1.8 Comparaison du contenu de deux fichiers avec diff et cmp

La commande `diff` donne les modifications à apporter au premier fichier spécifié pour qu'il ait le même contenu que le second. Par exemple :

```
diff pass.tmp /etc/passwd
```

affichera les modifications à apporter au fichier `pass.tmp` pour qu'il ait le même contenu que le fichier `/etc/passwd`.

Les modifications à apporter sont données sous forme de message. Par exemple :

- **3a4,7** indique que après la troisième ligne du premier fichier doivent être incrustées les lignes 4 à 7 du second. Attention, les messages suivants indiqueront les numéros des lignes sans tenir compte des incrustations qui auront éventuellement été apportées.
- **5,8d4** indique que les lignes 5 à 8 du premier fichier doivent être supprimées, car elles n'existent pas derrière la ligne 4 du second.
- **10,15c12,17** indique que les lignes 10 à 15 du premier fichier doivent être intégralement changées contre les lignes 12 à 17 du second.

Dans les trois cas de figure, les lignes précédées du signe "<" se rapportent au premier fichier, et les lignes précédées du signe ">" se rapportent au second.

L'option *-b* permet de ne pas tenir compte des espaces lors de la comparaison des lignes.

La commande `cmp` permet de comparer deux fichiers, et d'afficher le nombre de caractères et de lignes qui diffèrent. L'option *-l* affiche sur trois colonnes le numéro de ligne où il y a une différence et les valeurs ASCII en base huit des caractères différents.

1.9 Gestion des processus

1.9.1 Traitement en tâche de fond

On peut lancer une commande en tâche de fond (le nom de la commande est suivit par '&') pour pouvoir garder la main pour lancer d'autres commandes depuis le terminal (sans quoi on ne peut plus lancer aucune commande jusqu'à la fin de l'exécution de la commande). Attention cependant, cette commande ne doit doit pas avoir de sortie écran ou attendre d'entrée clavier. Par exemple, la commande suivante recherche tous les fichiers dont le nom commence par "*install*" :

```
find / -type f -name "install*" -print > liste 2> /dev/null &
```

Cette commande étant assez longue, on peut ainsi continuer à travailler sur la même console ou la même fenêtre **xterm**. Pour supprimer les sorties écran, on redirige la sortie standard vers un fichier, et la sortie d'erreur standard vers le "trou noir".

Lorsqu'on lance une commande en tâche de fond, deux numéros apparaissent :

```
[1] 439
```

qui indiquent les numéros de tâche de fond et de processus. Le numéro de tâche de fond est celui qui correspond au **numéro de processus en action**, qu'il tourne ou non. Si on lance une autre commande, elle aura un autre numéro de tâche de fond :

```
[2] 442
```

Le numéro de processus est le même que celui qui est affiché par la commande `ps`, et qui sert à forcer l'arrêt d'une commande grâce à la commande `kill` :

```
439  p2 D      0:02 find / -type f -name install* -print
```

La commande `jobs` permet d'afficher les commandes qui s'éxecutent en arrière plan :

```
[3]-  Running  find / -type f -name "install*"
-print >liste 2>err.moi &
```

L'option *-l* permet d'afficher en plus le numéro de processus :

```
[4]+ 455  Running  find / -type f -name "install*" -print
>liste 2> /dev/null &
```

En appuyant sur les touches `Ctrl+z` (ou celles définies par la ligne "*susp =* " de la commande `stty -a`), on peut stopper l'éxecution d'une commande. Elle se retrouve alors en arrière plan, comme l'indique le résultat de la commande `jobs` :

```
[4]+  Stopped  find / -type f -name "install*" -print
>liste 2> /dev/null
```

On peut relancer la commande en arrière plan avec la commande `bg %4`, ou en avant plan avec la commande `fg %4`. Au lieu du numéro de job, on peut spécifier le numéro de processus.

Si on veut voir exécuter la commande d'arrière plan en avant plan, il faut appeller la commande `kill` avec l'option *-STOP* pour suspendre le processus, et avec l'option *-CONT* pour le relancer (respectivement `kill -STOP %4` et `kill -CONT %4`).

Si il faut attendre la fin d'un processus particulier avant de reprendre la main, on peut lancer la commande en tâche de fond, puis invoquer la commande `wait %4`. On pourra continuer dès que le processus spécifié sera terminé.

1.9.2 Affichage des processus en cours avec ps

La commande `ps` permet d'afficher les programmes qui ont été lancés et qui ne n'ont pas été arrêtés. Le premier numéro correspond au numéro de processus. Plus de processus peuvent être listés avec les options *-a, -u, -x* ou les trois ensembles. L'option *-l* permet d'afficher le *numéro de priorité* (voir plus loin). L'option *-f* affiche les processus en cours de manière plus clair (les processus *pères* et *fils* sont mis en évidence).

1.9.3 Arrêt d'un processus avec kill

La commande `kill <numero de processus>` ou `kill %<numero de job>` arrête le programme correspondant au numéro de processus / numéro de job spécifié. Par défaut, c'est le signal **SIGTERM** qui est envoyé (comme *terminer*). Si le processus n'est par arrêté, employer l'option *-9* ou l'équivalent *-SIGKILL*, qui arrête à coup sûr un processus. L'option *-l* donne la liste de tous les signaux qu'on peut envoyer à la commande `kill`.

Si on lance une commande et qu'on veux se déconnecter, avec la commande `exit`, le signal **SIGHUP** est envoyé à tous les processus, si bien que la commande lancée sera terminée. Pour qu'elle continue à s'éxecuter malgrès tout, il faut lancer la commande `nohup` devant :

```
nohup find / -type f -name "install*" -print
> liste 2> /dev/null &
```

Avec l'emploi de la commande `nohup`, la redirection est faite d'office si rien n'est précisé par l'utilisateur vers le fichier `nohup.out` pour la sortie standard et `/dev/null` pour la sortie d'erreur standard.

1.9.4 Priorité d'un processus

Chaque processus se voit affecter une **priorité** qui correspond à un numéro. Lorsqu'une commande est lancée, le processus a une priorité maximale. Plus le processus occupe de temps d'éxecution pour le processeur, plus son numéro de priorité baisse, et moins le processus occupe de temps d'éxecution pour le processeur, plus son numéro de priorité augmente. Ainsi, plusieurs processus peuvent être éxecutés en même temps. La commande `nice` permet de diminuer la priorité du processus (pour les commandes longues et peu urgentes, par exemple). Le paramétre spécifié après l'option *-n* est un nombre compris entre 0 et 20 qui indique le facteur de diminution :

```
nice -n 20 find / -type f -name "install*" -print
> liste 2> /dev/null &
```

L'administarteur système (compte **root**) peut également augmenter la priorité avec un nombre négatif :

```
nice -n -20 find / -type f -name "install*" -print
> liste 2> /dev/null &
```

et la commande `renice` permet de changer le facteur de priorité en cours d'exécution de la commande, en spécifiant le nouveau facteur de priorité et le numéro de processus :

```
renice 10 733
```

Le résultat informe alors le super utilisateur du changement :

```
733: old priority 19, new priority 10
```

1.9.5 Mesure du temps d'exécution d'un processus

La commande `time` permet de mesurer le temps d'exécution d'une commande. Elle fournit les temps *réels* (temps total), *utilisateurs* (durée nécessaire au processeur pour exécuter les ordres du programme) et *systèmes* (durée nécessaire au processeur pour traiter les ordres du système d'exploitation). Voici un exemple d'utilisation :

```
time ls -lR / > liste.ls 2> /dev/null
```

Le résultat s'affiche alors :

```
real    2m39.458s
user    0m9.060s
sys     0m32.330s
```

L'option *-p* permet d'avoir les résultats en secondes :

```
real 144.55
user 8.75
sys 27.57
```

On peut ainsi calculer le **facteur d'évaluation** qui traduit la surcharge du système :

$$facteur = \frac{temps_{utilisateur} + temps_{systeme}}{temps_{reel}}$$

Ici, le rapport est de

$$\frac{8.75 + 27.57}{144.55} \sim \frac{1}{4}$$

Un facteur normal se situe entre $\frac{1}{5}$ et $\frac{1}{10}$. Un facteur supérieur à $\frac{1}{20}$ traduit une surcharge du système.

1.9.6 Droits d'accès

Les fichiers sont protégés en lecture, écriture ou exécution pour des raisons de sécurité. Pour avoir accès à un fichier, il faut en avoir les droits. La commande `chmod` permet de changer les droits d'accès à un fichier. Les droits peuvent être de lecture (r), d'écriture (w) ou d'exécution (x). On ajoute un droit à l'aide du signe "+" et on en retire à l'aide du signe "-", pour l'utilisateur (u), le groupe (g) ou les autres (o). Ainsi, la commande :

```
chmod +x fichier
```

rends exécutable un fichier (comme un script, par exemple), et

```
chmod ug+x fichier
```

rends exécutable un fichier pour l'utilisateur et tout le groupe. Les droits s'affichent avec la commande `ls -l` pour l'utilisateur, le groupe et les autres.

Une autre facon de faire est d'attribuer un chiffre correspondant aux droits recherchés comme suit :
- pour **l'utilisateur**, les droits d'accès en lecture sont de **400**, en écriture de **200** et en exécution de **100**.
- pour **le groupe**, les droits d'accès en lecture sont de **40**, en écriture de **20** et en exécution de **10**.
- pour **les autres**, les droits d'accès en lecture sont de **4**, en écriture de **2** et en exécution de **1**.
- on aditionne ensuite les droits pour chacun.

Ainsi :

```
rwxr-xr-x
```

équivaut à $400 + 200 + 100 = 700$ pour l'utilisateur, $40 + 10 = 50$ pour le groupe et $4 + 1 = 5$ pour les autres, soit au total $700 + 50 + 5 = 755$:

```
chmod 755 fichier
```

Pour attribuer les droits d'accès par défaut, il faut exécuter la commande `umask` et attribuer un numéro particulier. Ce numéro est obtenu en soustreyant le chiffre **7** à chaque chiffre du numéro de droit d'accès. Ainsi, pour attribuer par défaut la valeur **750** (tous les droits pour l'utilisateur, droits de lecture et d'exécution pour le groupe et aucun pour les autres), il faut fournir à la commande `umask` la valeur **027** ($7 - 7 = 0$, $7 - 5 = 2$ et $7 - 0 = 7$) :

```
umask 027
```

Cette ligne pourra figurer dans le fichier d'initialisation `.bashrc` pour que ces droits soient pris en compte à chaque session.

Pour définir des droits d'accès étendus (qui permettent à un utilisateur d'effectuer une opération sur des fichiers lui appartenant, comme modifier son mot de passe par exemple), utiliser l'option *+s* ou rajouter les valeurs **4000** pour l'utilisateur et **2000** pour le groupe. Pour le groupe :

```
chmod 2755
```

et pour l'utilisateur :

```
chmod 4755
```

Tout utilisateur peut copier des fichiers dans le répertoire `/tmp/`. Pour que ces fichiers ne puissent pas être effacés par un autre utilisateur que celui qui les a copiés et **root**, il faut positionner le *Sticky Bit* (en tant que root, bien sûr) :

```
chmod u+t /tmp
```

ou

```
chmod 1777 /tmp
```

1.10 Autres commandes utiles

less : permet de visualiser le contenu d'un fichier sans le modifier. L'option *+214* permet d'afficher à partir de la ligne **214**, et *+/ER/* permet d'afficher à partir de la ligne commencant par l'expression régulière **ER**. Par exemple :

```
less +/^toto
```

commencera l'affichage à partir de la ligne commencant par "toto".

tail : permet de n'afficher que les dernières lignes d'un fichier. L'option *-20* affichera les 20 **dernières** lignes, et l'option *+20* affichera les 20 **premières** lignes.

`stty -a` : affiche les parametres du terminal ; pour changer, faire par exemple : `stty erase '<Ctrl>p'`

`pwd` : affiche le répertoire courant.

`uname` : affiche les paramétres du système.

`whoami / who am i / who` : quel est mon login / id. avec plus d'informations / qui d'autre est logué.

`date` : affiche la date courante. Le format `date '+%Hh %Mmn'` affiche en format 11h 35mn et `date '+%d/%m/%y'` en format 31/05/98. On peut aussi insérer un commentaire :
`date "+Il est %H:%M, le %d.%m.19%y"`

`cal` : affiche le calendrier du mois courant. `cal 05 2002` affiche le calendrier du mois de mai 2002.

`passwd` : permet de changer de mot de passe.

`man` : accès au manuel. Il faut préciser de quelle commande on souhaite avoir des informations, par exemple : `man passwd`.

`apropos <mot>` : affiche toutes les pages de manuel contenant le *mot*.

`echo` : affiche le texte qui suit la commande, $V la valeur de la variable *V*.

`ls` : affiche la liste des fichiers. Les options sont nombreuses et des raccourcis existent pour les plus utiles (pour les connaître, taper `alias`).

Les plus utiles sont :

- `ls -l` : affiche les caractéristiques du fichier .
- `ls -a` : affiche en plus les fichiers cachés.

`ls -F` : permet de reconnaitre le type de fichier ("*" pour un exécutable, "/" pour un répertoire).

`ls -R` : affiche les fichier du répertoire et des sous répertoires.

`ls -t` : affiche les fichiers du plus récent au plus vieux.

`ls -i` : affiche le numéro d'inode du fichier ; ce numéro est unique et ne correspond donc qu'a un fichier.

Remarque : les options peuvent se combiner, ainsi `ls -alF` équivaut à `ls -a -l -F`.

`cat` : affiche le contenu du fichier qui suit cette commande. Si le fichier tient sur plus d'un écran, on peut alors utiliser la commande **more** (on peut même rechercher une chaîne de charactères en appuyant sur "/"), ou mieux encore la commande **less**. On peut aussi utiliser la commande `cat` pour éditer un fichier :

```
cat > toto
echo Bonjour !
<Ctrl>d
```

On sort alors du fichier.

`cp` : copie un fichier vers un autre. Si le nouveau fichier doit avoir le même nom que l'ancien, taper `cp /etc/hosts /home/mathieu/` par exemple.

`rm` : efface un fichier. L'option *-i* demande confirmation, *-r* détruit le répertoire et tous les fichiers qu'il contient, * tous les fichiers, *?as ? ?* tous les fichiers de 5 lettres ayant a en position 2 et s en position 3 dans leur nom.

`wc` : compte les lignes (*-l*), mots (*-w*) et caractères(*-c)* d'un fichier.

`chsh` : change de shell.

`Ctrl-t` : inverse 2 lettres.

`~` : représente le répertoire personnel de l'utilisateur. Ainsi, la commande :

```
ls ~
```

permet d'afficher la liste de tous les fichiers du répertoire personnel, où que l'on se trouve, et la commande :

```
ls ~piou
```

permet d'afficher la liste de tous les fichiers du répertoire personnel de l'utilisateur **piou**. Noter qu'il faut parfois appyuer 2 fois sur la touche pour que ca marche.

`du` : affiche le nombre de blocs qu'occupe chaque fichier dans le répertoire courant.

`export` : rajoute à une variable un argument. Par exemple :

```
export PATH=$PATH:/usr/sbin
```

La variable PATH contient les répertoires dans lesquels une commande va être recherchée. Ces répertoires sont classés et dès que le shell a trouvé la commande il s'arrête de chercher et l'exécute. Noter qu'une variable d'environnement ne peut commencer par un chiffre.

Pour avoir une liste des variables d'environnement définies, taper `set` pour toutes les variables, ou `env` pour avoir seulement la liste des variables d'environnement.

`alias` : permet de définir un alias. Par exemple :

```
alias ll='ls -l'
```

aura pour effet que si on tape `ll` dans le shell, c'est la commande `ls -l` qui s'exécutera. Pour supprimer l'alias, utiliser `unalias`.

`history` : permet d'afficher l'historique des commandes. `history 10` permet de n'afficher que les 10 dernières commandes passées. La commande `!500` permet d'exécuter la commande 500 dans l'historique, `!less` permet d'exécuter la dernière commande commencant par *less*, `!!` permet d'exécuter la dernière commande (la flêche du haut permet également d'y accéder), et on peut même modifier la commande : `!! | less`. On peut aussi modifier la dernière commande sans avoir à la resaisir. Par exemple si on tape :

```
less /etc/XF86config
```

ca ne marche pas car il faut taper */etc/XF86Config* et non */etc/XF86config*. Pour corriger, taper :

```
^con^Con
```

ce qui signifie : remplacer *con* par *Con*. Bon, assez d'insultes...

`ln` : permet de créer un lien d'un fichier vers un autre. Par exemple, le lien :

```
ln -s /etc/XF86Config XF86config
```

a pour conséquences que si on lance une commande (`less, emacs`...) sur le fichier *XF86config*, c'est sur le fichier `/etc/XF86Config` qu'elle s'effectuera. Les liens sont le plus souvent *symboliques* (avec l'option *-s*).

`which` : affiche le chemin complet d'un exécutable inclut dans la variable d'environnement **$PATH**. Par exemple, `which xv` affichera :

```
/usr/X11R6/bin/xv
```

Note : en général, les fichiers se terminant par *rc* ou commencant par un point sont des fichiers ASCII de configuration.

Il existe d'autre commandes utiles pour l'administration système comme la sauvegardes des données. Elles seront évoquées à la section 9.3 page 216.

Chapitre 2

Utilisation de XFree86

2.1 Installation et configuration

2.1.1 Installation

Si les fichiers ont été récupérés sur un site, par exemple, commencer par décompacter l'archive dans le répertoire `/usr/X11R6` :

```
gzip -dc X31bin.tgz | tar xfB -
```

Il faut ensuite faire pointer (par un lien symbolique) les fichier suivants :

```
/usr/bin/X11/X /usr/X11R6/bin/X
```

vers le serveur souhaité (`/usr/X11R6/bin/XF86_SVGA` peut être une bonne idée) :

```
ln -sf /usr/X11R6/bin/XF86_SVGA /usr/bin/X11/X
ln -sf /usr/X11R6/bin/XF86_SVGA /usr/X11R6/bin/X
```

On peut vérifier que le lien est bien établit par la commande suivante :

```
ls -l /usr/bin/X11/X ls -l /usr/X11R6/bin/X
```

Le résultat pour le premier doit ressembler à ca :

```
lrwxrwxrwx 1 root root 16 Nov 3 18:08 /usr/bin/X11/X ->
/usr/X11R6/bin/XF86_SVGA
```

ou

```
lrwxrwxrwx 1 root root 16 Nov 3 18:08 /usr/X11R6/bin/X ->
/usr/X11R6/bin/XF86_SVGA
```

Le programme `/usr/X11R6/bin/SuperProbe` peut renseigner sur la carte vidéo et le serveur à utiliser. Avant de passer à la configuration, il faut s'assurer que :

1. le répertoire `/usr/X11R6/bin` est bien contenu dans la variable d'environnement **PATH** (voir la commande `export`).
2. le répertoire `/usr/X11R6/lib/` est localisable par l'éditeur de liens dynamiques `ld.so`. Pour cela, ajouter la ligne `/usr/X11R6/lib/` dans le fichier `/etc/ld.so.conf` puis exécuter `/sbin/ldconfig` pour que cette modification soit prise en compte.

2.1.2 Configuration

Pour être sûr ne ne pas avoir de problème lors du premier lancement du serveur X, il vaut mieux commencer par une configuration peu sophistiquée. Une fois qu'on est sur que tout le matériel est reconnu, on peut affiner la configuration (c'est en général là qu'apparaissent les problèmes...). Les fichiers suivants contiennent des informations indispensables à la configuration d'un serveur X :

- le fichier `/usr/X11R6/lib/X11/doc/README.Config`.
- la page de manuel de `XFree86`.
- la page de manuel de *XF86Config*.
- la page de manuel du serveur employé (**XF86_SVGA, XF86_S3**...).

La configuration d'un serveur X consiste à écrire un fichier, le fichier `/usr/X11R6/lib/X11/XF86Config`, contenant tous les paramètres nécessaires au bon fonctionnement de XFree86. On peut se servir du fichier de configuration générale `/usr/X11R6/lib/X11/XF86Config.eg` comme point de départ de fichier de configuration.

La section qui suit présente un exemple de fichier *XF86Config* et en explique les principes. Les données contenues dans ce fichier doivent impérativement être adaptées au matériel. De mauvais paramètres peuvent endommager la carte vidéo !

2.1.3 Le fichier XF86Config

Attention ! Les données contenues dans le fichier qui suit doivent impérativement être adaptées au

matériel possédé !

Sinon, risques de bobos pour la carte vidéo (Yo !).

Chaque section du fichier de configuration *XF86Config* commence par `Section` et se termine par `EndSection`. La première section s'appelle `Files` :

```
Section "Files"


#--------------------------------------------------------------------#
# Chemins d'acces des fichiers de couleurs et de polices d'ecriture. #
#--------------------------------------------------------------------#


# Le fichier /usr/X11R6/lib/X11/rgb.txt doit etre lisible par tous
# (mode 444).

    RgbPath     "/usr/X11R6/lib/X11/rgb"
    FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
EndSection
```

Il suffit juste de vérifier les chemins d'accès, et que chaque police installée (dans le répertoire `/usr/X11R6/lib/X11/fonts/`) est bien référencée par une ligne *FontPath* (en cas de problème, invoquer `mkfontdir` dans chaque répertoire de polices).

La section suivante, `ServerFlags`, en général vide, spécifie certaines options du serveur :

```
#-------------------------------------------------#
# Drapeaux pour valider ou non certaines options. #
#-------------------------------------------------#


Section "ServerFlags"
```

```
# Valider cette ligne pour provoquer un core-dump des la reception
# d'un signal. La console sera alors peut etre inutilisable, mais le
# core-dump facilitera le debogage.

#    NoTrapSignals

# Valider cette ligne pour annuler la fonction arret du serveur
# de la combinaison de touches <Crtl><Alt><BackSpace>.

#    DontZap
EndSection
```

La section suivante, `Keyboard`, concerne le clavier :

```
#------------------#
# Section clavier. #
#------------------#


Section "Keyboard"
 Protocol     "Standard"
 AutoRepeat   500 5
 LeftAlt      Meta
 RightAlt     ModeShift
 RightCtl     Compose
 ScrollLock   ModeLock
 XkbKeymap    "xfree86(fr)"
 XkbKeycodes  "xfree86"
 XkbTypes     "default"
 XkbCompat    "default"
 XkbSymbols   "en\_US(pc101)+fr"
 XkbGeometry  "pc(pc101)"
EndSection
```

D'autres options sont disponibles. Les valeurs ci-dessus devraient convenir à la plupart des claviers (même francais).

La section suivante, `Pointer`, spécifie les paramètres nécessaires au bon fonctionnement de la souris :

```
#-----------------#
# Section souris. #
#-----------------#
```

```
Section "Pointer"

# Protocol (et non la marque) utilise par la souris.

    Protocol    "Microsoft"

# Peripherique utilise (/dev/mouse, lien symbolique vers /dev/ttyS0
# pour un port serie ou /dev/psaux pour un pilote de souris bus.
#
# Verifier que le noyau dispose du peripherique (recompiler le noyau
# si necessaire), et verifier que le peripherique indique existe et que
# les permissions permettent d'y acceder.

    Device      "/dev/mouse"

# Seules certaines souris Logitech requierent ces deux parametres.

#    BaudRate   9600
#    SampleRate 150

# Emulate3Buttons est une option pour les souris Microsoft ayant
# 2 boutons.
# Emulate3Timeout est le temps accorde en millisecondes
# (50 ms par defaut).

#    Emulate3Buttons
#    Emulate3Timeout    50

# ChordMiddle est une option pour certaines souris 3 boutons Logitech.

#    ChordMiddle
EndSection
```

La section suivante, `Monitor`, informe XFree des caractéristiques du moniteur :

```
#--------------------#
# Section moniteur. #
#--------------------#
```

```
# Il peut y avoir plusieurs sections moniteur.

Section "Monitor"

# Identifier permet d'attribuer un nom arbitraire a l'entree Monitor.
# Ce nom servira a designer le moniteur un peu plus loin dans le fichier.

    Identifier  "Generic Monitor"

# Bandwidth stipule la bande passante video maximum de l'appareil en MHz.
# C'est la vitesse maximum a laquelle la carte video peut envoyer des
# pixels au moniteur.

# HorizSync indique la frequence de synchronisation horizontale en kHz.
# HorizSync peut etre une liste separee par des virgules, ou une ou
# plusieurs gammes de valeurs.
#
# NOTE : LES VALEURS DONNEES ICI SONT SEULEMENT DES EXEMPLES.
# SE REFERER A LA NOTICE DU MONITOR POUR LES VALEURS CORRECTES.

# valeur typique pour un moniteur a frequence fixe.

     HorizSync  31.5

# pour un appareil <<multisync>>.

#    HorizSync  30-64

# gamme de valeurs pour un moniteur a frequence fixe.

#    HorizSync  31.5, 35.2

# plusieurs gammes de valeurs de frequences de synchronisation.

#    HorizSync  15-25, 30-50

# VertRefresh indique les frequences de synchronisationverticales en Hz.
# Memes remarques que pour HorizSync.

# valeur typique pour un moniteur a frequence fixe.
```

```
     VertRefresh 60

# pour un appareil <<multisync>>.

#    VertRefresh 50-100

# gamme de valeurs pour un moniteur a frequence fixe.

#    VertRefresh 60, 65

# plusieurs gammes de valeurs de frequences de synchronisation.

#    VertRefresh 40-50, 80-100

# ModeLine sert a specifier une resolution d'affichage. La syntaxe est :
#
#    ModeLine <nom> <horloge> <horizontal> <vertical>
#
# ou <nom> represente une chaine arbitraire (qui servira plus tard);
# <horloge> symbolise la frequence, exprimee en MHz, a laquelle la
# carte video peut envoyer des pixels au moniteur pour une resolution
# donnee;
# <horizontal> et <vertical> : a quel moment le canon du tube cathodique
# doit envoyer des electrons (s'allumer), et quand les impulsions
# horizontales et verticales doivent apparaitre.

# Un mode VGA standard 640x480 (hsync = 31.5kHz, refresh = 60Hz).

# Les deux syntaxes sont equivalentes.

#   ModeLine "640x480" 25.175 640 664 760 800 480 491 493 525

    Mode "640x480"
        DotClock        25.175
        HTimings        640 664 760 800
        VTimings        480 491 493 525
    EndMode

# Les deux syntaxes sont equivalentes.
```

```
    ModeLine "1024x768i" 45 1024 1048 1208 1264 768 776 784 817 Interlace

#    Mode "1024x768i"
#        DotClock       45
#        HTimings       1024 1048 1208 1264
#        VTimings       768 776 784 817
#        Flags          "Interlace"
#    EndMode
EndSection
```

Les fichiers du répertoire `/usr/X11R6/lib/X11/doc` peuvent donner des informations concernant les caractéristiques du moniteur, en particulier le calcul de chaque résolution désirée dans les fichiers suivants :

- VideoModes.doc
- modeDB.txt
- Monitors

Les lignes `ModeLine` prévues pour le modèle de moniteur vidéo seront sans doute dans ces fichiers. Si rien n'est indiqué, le fichier `VideoModes.doc` permet de réaliser manuellement cette configuration.

La section suivante, `Device`, indique les paramètres de la carte vidéo :

```
#----------------#
# Section device #
#----------------#


Section "Device"
    Identifier  "Generic VGA"

# Le reste sera remplit plus tard. Le serveur X testera cette partie.

EndSection
```

La dernière section, `Screen`, spécifie la combinaison moniteur/carte vidéo d'un serveur particulier :

```
#---------------#
# Section ecran #
#---------------#


# Le serveur couleur SVGA.
```

```
Section "Screen"

# Driver specifie le type du serveur X mis en oeuvre.
# Valeurs possibles : Accel, SVGA, VGA16, VGA2 ou Mono.
    Driver      "svga"

# Device specifie l'identificateur de la section Device, a la ligne
# Identifier.

    Device      "Generic SVGA"

# Monitor specifie l'identificateur de la section Monitor, a la ligne
# Identifier.

    Monitor     "Generic Monitor"

# Sous section Display. Les options sont :
#
# Depth : nombre de plans de couleurs (nombre de bits par pixels).
# Les valeurs sont 8, 4 pour VGA16, 1 pour monochrome. Les valeurs
# 16, 24 ou 32 pour cartes rapides et possedant suffisament de memoire.
#
# Modes : Noms definits a la section Monitor directive ModeLine.
#
# Virtual : Initialise la taille du bureau virtuel.
#
# ViewPort : indique les coordonnees du coin superieur gauche a afficher
# au demarage de XFree86 (fenetre centree par defaut, pas pratique !).

    Subsection "Display"
        Depth       8
        Modes       "1024x768i" "640x480"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection

EndSection
```

2.1.4 Caractéristiques de la carte vidéo

Dans le fichier *XF86Config* précédement crée, il manque les informations concernant la carte vidéo. Le serveur X peut tester cette partie, et afficher les paramètres qu'on pourra ensuite intégrer au fichier de configuration. Les fichiers suivants du répertoire `/usr/X11R6/lib/X11/doc` contiennent des informations pouvant éviter cette opération :

- modeDB.txt ;
- AccelCards ;
- Devices ;
- le fichier README.xx correspondant à la carte vidéo employée (README.S3, README.Mach32...) ;

En premier lieu, il convient de déterminer le circuit vidéo employé par la carte vidéo, avec la commande `SuperProbe` :

```
/usr/X11R6/bin/SuperProbe
```

Le résultat apparait après quelque secondes :

```
SuperProbe Version 2.15 (4 August 1997)
        (c) Copyright 1993,1994 by David Wexelblat <dwex@xfree86.org>

        This work is derived from the 'vgadoc2.zip' and
        'vgadoc3.zip' documentation packages produced by Finn
        Thoegersen, and released with all appropriate permissions
        having been obtained.  Additional information obtained from
        'Programmer's Guide to the EGA and VGA, 2nd ed', by Richard
        Ferraro, and from manufacturer's data books

The author welcomes bug reports and other comments mailed to
the electronic mail address above.  In particular, reports of
chipsets that this program fails to correctly detect are
appreciated.

Before submitting a report, please make sure that you have the
latest version of SuperProbe (see http://www.xfree86.org/FAQ).

WARNING - THIS SOFTWARE COULD HANG YOUR MACHINE.
          READ THE SuperProbe.1 MANUAL PAGE BEFORE
          RUNNING THIS PROGRAM.

          INTERRUPT WITHIN FIVE SECONDS TO ABORT!
```

```
First video: Super-VGA
        Chipset: S3 ViRGE/DX (PCI Probed)
        Memory:  2048 Kbytes
        RAMDAC:  Generic 8-bit pseudo-color DAC
                 (with 6-bit wide lookup tables (or in 6-bit mode))
```

Visiblement, l'exemple précédent montre que la carte vidéo employée ici est une carte `S3 ViRGE/DX`, avec 2048 Kbytes de mémoire. Pour savoir sous quel nom le serveur X reconnait ce processeur, taper la commande suivante (la page de manuel associée au serveur peut aussi le mentionner) :

```
X -showconfig > /tmp/showconfig.txt 2>&1
```

Le résultat peut être lu dans le fichier `/tmp/showconfig.txt` avec n'importe quel éditeur (`emacs, vi`...) :

```
XFree86 Version 3.3.1 / X Window System
(protocol Version 11, revision 0, vendor release 6300)
Release Date: August 4 1997
        If the server is older than 6-12 months, or if your card is newer
        than the above date, look for a newer version before reporting
        problems.  (see http://www.XFree86.Org/FAQ)
Operating System: Linux 2.0.32 i686 [ELF]
Configured drivers:
  SVGA: server for SVGA graphics adaptors (Patchlevel 0):
      NV1, STG2000, ET4000, ET4000W32, ET4000W32i, ET4000W32i_rev_b,
      ET4000W32i_rev_c, ET4000W32p, ET4000W32p_rev_a, ET4000W32p_rev_b,
      ET4000W32p_rev_c, ET4000W32p_rev_d, ET6000, et3000, pvga1, wd90c00,
      wd90c10, wd90c30, wd90c24, wd90c31, wd90c33, gvga, ati, sis86c201,
      sis86c202, sis86c205, tvga8200lx, tvga8800cs, tvga8900b, tvga8900c,
      tvga8900cl, tvga8900d, tvga9000, tvga9000i, tvga9100b, tvga9200cxr,
      tgui9320lcd, tgui9400cxi, tgui9420, tgui9420dgi, tgui9430dgi,
      tgui9440agi, tgui96xx, cyber938x, clgd5420, clgd5422, clgd5424,
      clgd5426, clgd5428, clgd5429, clgd5430, clgd5434, clgd5436, clgd5446,
      clgd5480, clgd5462, clgd5464, clgd5465, clgd6205, clgd6215, clgd6225,
      clgd6235, clgd7541, clgd7542, clgd7543, clgd7548, clgd7555, ncr77c22,
      ncr77c22e, cpq_avga, mga2064w, mga1064sg, mga2164w, oti067, oti077,
      oti087, oti037c, al2101, ali2228, ali2301, ali2302, ali2308, ali2401,
      cl6410, cl6412, cl6420, cl6440, video7, ct65520, ct65525, ct65530,
      ct65535, ct65540, ct65545, ct65546, ct65548, ct65550, ct65554,
      ct65555, ct68554, ct64200, ct64300, ark1000vl, ark1000pv, ark2000pv,
      ark2000mt, mx, realtek, AP6422, AT24, s3_virge, generic
```

Le serveur utilisé sera donc `s3_virge`. Le test suivant permet de s'en assurer. Il faut le faire lorsque rien n'est chargé par le système, sinon certaines mesures seront faussées :

```
X -probeonly > /tmp/probeonly.txt 2>&1
```

ou, si le shell est **csh** :

```
X -probeonly >& /tmp/probeonly.txt
```

Voici un exemple de contenu de fichier `/tmp/probeonly.txt` ainsi obtenu :

```
XFree86 Version 3.3.1 / X Window System
(protocol Version 11, revision 0, vendor release 6300)
Release Date: August 4 1997
        If the server is older than 6-12 months, or if your card is newer
        than the above date, look for a newer version before reporting
        problems.  (see http://www.XFree86.Org/FAQ)
Operating System: Linux 2.0.32 i686 [ELF]
Configured drivers:
  SVGA: server for SVGA graphics adaptors (Patchlevel 0):
      NV1, STG2000, ET4000, ET4000W32, ET4000W32i, ET4000W32i_rev_b,
      ET4000W32i_rev_c, ET4000W32p, ET4000W32p_rev_a, ET4000W32p_rev_b,
      ET4000W32p_rev_c, ET4000W32p_rev_d, ET6000, et3000, pvga1, wd90c00,
      wd90c10, wd90c30, wd90c24, wd90c31, wd90c33, gvga, ati, sis86c201,
      sis86c202, sis86c205, tvga8200lx, tvga8800cs, tvga8900b, tvga8900c,
      tvga8900cl, tvga8900d, tvga9000, tvga9000i, tvga9100b, tvga9200cxr,
      tgui9320lcd, tgui9400cxi, tgui9420, tgui9420dgi, tgui9430dgi,
      tgui9440agi, tgui96xx, cyber938x, clgd5420, clgd5422, clgd5424,
      clgd5426, clgd5428, clgd5429, clgd5430, clgd5434, clgd5436, clgd5446,
      clgd5480, clgd5462, clgd5464, clgd5465, clgd6205, clgd6215, clgd6225,
      clgd6235, clgd7541, clgd7542, clgd7543, clgd7548, clgd7555, ncr77c22,
      ncr77c22e, cpq_avga, mga2064w, mga1064sg, mga2164w, oti067, oti077,
      oti087, oti037c, al2101, ali2228, ali2301, ali2302, ali2308, ali2401,
      cl6410, cl6412, cl6420, cl6440, video7, ct65520, ct65525, ct65530,
      ct65535, ct65540, ct65545, ct65546, ct65548, ct65550, ct65554,
      ct65555, ct68554, ct64200, ct64300, ark1000vl, ark1000pv, ark2000pv,
      ark2000mt, mx, realtek, AP6422, AT24, s3_virge, generic
(using VT number 7)

XF86Config: /etc/XF86Config
(**) stands for supplied, (--) stands for probed/default values
(**) XKB: keymap: "xfree86(fr)" (overrides other XKB settings)
```

```
(**) Mouse: type: Microsoft, device: /dev/mouse, baudrate: 1200,
       Chorded middle button

********** Plusieurs lignes supprimees... **********

(**) SVGA: Graphics device ID: "DSV3325"
(**) SVGA: Monitor ID: "My Monitor"
(--) SVGA: PCI: S3 ViRGE/DX or /GX rev 1, Memory @ 0xe0000000
(--) SVGA: S3V: ViRGE/DXGX rev 1, Linear FB @ 0xe0000000
(--) SVGA: Detected S3 ViRGE/DXGX
(--) SVGA: using driver for chipset "s3_virge"
(--) SVGA: videoram:  2048k
(--) SVGA: Ramdac speed: 170 MHz
(--) SVGA: Detected current MCLK value of 69.801 MHz
(--) SVGA: chipset:  s3_virge
(--) SVGA: videoram: 2048k

********** Plusieurs lignes supprimees... **********
```

On peut alors rajouter une ligne `Chipset` dans la section `Device` du fichier de configuration *XF86Config* :

```
Section "Device"
    Identifier  "Generic VGA"
    Chipset "s3_virge"
EndSection
```

Il reste maintenant à déterminer les fréquences de travail de la carte vidéo (vitesse à laquelle la carte vidéo envoit des pixels au moniteur). L'un des fichiers pré-cités peut donner ces renseignements. Il s'agit d'une ligne du type :

```
Clocks  25.0 28.0 40.0 0.0 50.0 77.0 36.0 45.0
```

qu'il ne reste plus qu'à insérer dans le fichier de configuration *XF86Config* :

```
Section "Device"
    Identifier  "Generic VGA"
    Chipset "s3_virge"
    Clocks  25.0 28.0 40.0 0.0 50.0 77.0 36.0 45.0
EndSection
```

Il est très important de recopier les valeurs **telles quelles**, et de **ne pas les trier** ou d'eliminer les doublons. La commande `X -probeonly` peut déterminer ces valeurs, si elles n'apparaissent pas dans le fichier *XF86Config* (sans quoi le serveur appliquera ces valeurs sans effectuer le test).

Il se peut que la carte vidéo se dote d'un générateur d'horloge programmable. Il faudra alors insérer une ligne `ClockChip`, à la place de la ligne `Clocks` dans la section `Device` du fichier de configuration *XF86Config* (l'un des fichiers de documentation peut préciser quel générateur utilise la carte vidéo). Par exemple, la ligne : `ClockChip "s3gendac"` pourra être insérée pour les cartes vidéo utilisant un générateur “S3 GENDAC”.

Plusieurs options peuvent être insérées dans cette section, afin d'optimiser les performances. Les fichiers de documentation ou les pages de manuel préciseront quelles options sont nécessaires pour la carte vidéo employée.

2.1.5 Démarrer XFree86

La commande `startx` permet de lancer un serveur X. Cette commande lance à son tour la commande `xinit`. Pour sortir de X, il faut presser `Ctrl-Alt-Backspace`.

Si ca ne marche pas, il faut vérifier le fichier de configuration *XF86Config*, ou s'assurer que c'est le bon serveur qui est appellé. Sinon, lancer un serveur X “brut” par la commande :

```
X > /tmp/x.out 2>&1
```

On peut ensuite l'arrêter par la commande `Ctrl-Alt-Backspace`, et examiner le fichier `/tmp/x.out`.

2.2 Pour commencer à s'amuser avec du X

2.2.1 Lancement normal

Pour initialiser le serveur X, taper la commande `startx`, qui en principe fait référence au script `/usr/X11R6/bin/startx` qui lui même fait appel à la commande `xinit`.

La commande `xinit` fait appel à deux fichiers, `.xinitrc` et `.Xclients`, qu'il cherche dans le répértoire personnel de l'utilisateur (définit par la variable **$HOME**, et accessible par la commande `cd` sans aucun arguments). Si ces fichiers n'existent pas, les fichiers utilisés sont les suivants :

```
/usr/X11R6/lib/X11/xinit/xinitrc
/usr/X11R6/lib/X11/xinit/xserverrc
```

Sans ces fichiers, le serveur X "brut" est lancé. Pour le rendre utilisable, il faut faire appel à un **window-manager** (gestionnaire de fenêtres). Le plus simple est **twm**, auquel il faut faire appel dans le fichier `.xinitrc`.

Voici un exemple simple de fichier `.xinitrc` qui initialise un serveur X et qui appelle quelque applications au démarrage :

```
xterm -ge 64x20+200+0 -sb -fn 9x15bold &
oclock -ge +20+250 &
exec twm
```

On peut lancer `xinit` à des fins de tests, avec ou sans arguments. Le premier argument spécifie le premier "client" lancé (en géneral un fenêtre **xterm**, car sans elle l'environnement graphique est inutilisable...), avec des options, et le deuxième argument spécifie le serveur utilisé (monochrome, SVGA, S3...). Voici un exemple[1] :

```
xinit /usr/X11R6/bin/xterm -ge 72x32+100+50 -bg pink
```

ou

```
xinit -ge 72x32+100+50 -bg pink
```

On peut toujours spécifier un autre premier client, comme une horloge `xclock`, par exemple, mais ce n'est pas très intéressant (le serveur est inutilisable, puisqu'on ne peut passer aucune commande, et on est obligé de sortir de manière pas très "propre" en appuyant sur `Ctrl-Alt-Backspace)` :

```
xinit /usr/X11R6/bin/xclock -bg yellow -fg red -hd green
```

Pour lancer un autre serveur que celui spécifié par défaut, il faut faire précéder le nom par deux tirets ("`--`"). Par exemple, pour lancer le serveur monochrome, taper :

```
xinit -- /usr/X11R6/bin/XF86_Mono
```

et on peut combiner les deux options comme suit :

```
xinit /usr/X11R6/bin/xclock -bg yellow -fg red -hd green --
/usr/X11R6/bin/XF86_Mono
```

ou

```
xinit -ge 72x32+100+50 -bg pink -- /usr/X11R6/bin/XF86_Mono
```

[1]Par défaut, **xinit** lance comme premier client **xterm**, l'émulateur de consoles qui permet de passer des commandes au shell (*xterm -ge 80x24+1+1 -n login*). Par conséquent, les deux lignes suivantes sont équivalentes.

De plus, les mêmes résultats peuvent être obtenus en spécifiant des noms de fichiers contenant les options. Par exemple, si le fichier `pxinitrc` contient la ligne :

```
xclock -bg yellow -fg red -hd green
```

et le fichier `pxserverrc` contient la ligne :

```
/usr/X11R6/bin/XF86_Mono
```

on peut lancer le serveur X par la commande suivante :

```
xinit pxinitrc -- pxserverrc
```

En pratique, c'est le script `/usr/bin/startx` qui se charge de spécifier les fichiers d'options pour la commande `xinit`. Ce script contient en dernière ligne :

```
xinit $clientargs -- $serverargs
```

où la variable **$clientargs** contient le nom le premier client, et **$serverargs** le nom de serveur.

Parmi les applications lancées par le fichier `.xinitrc`, il y a `xterm`, l'émulation d'un terminal texte (qui permet de passer des commandes) et `oclock` qui affiche une horloge. Chaque commande est lancée en tâche de fond (le nom de la commande est suivit par '&') pour pouvoir garder la main pour lancer d'autres commandes depuis le terminal (sans quoi on ne peut plus lancer aucune commande jusqu'à la fin de l'exécution de la commande, bonjour le multi-tâches qui a fait la réputation d'UNIX !).

Ces commandes lancées au démarrage sont suivies de nombreuses options. Les sections suivantes en présentent les principales pour quelque applications courantes.

2.2.2 Démarrage automatique de X

Lorsqu'on se logue, on peut automatiquement lancer la commande `startx`. Pour cela, il suffit de l'insérer dans le dernier fichier de configuration lancé au moment du login, le fichier personnel `.bash_profile`. Cependant, si on se connecte sur un système distant, il ne faut pas lancer X (X fonctionne là où se trouve le moniteur). Dans ce cas précis, la variable **$TERM** vaut "*console*". Il faut donc insérer dans le fichier `.bash_profile` les lignes suivantes :

```
if [ ''$TERM'' = ''console'' ]; then
startx
fi
```

2.2.3 Configuration d'un login graphique

Le programme `xdm` permet d'afficher un login graphique. Son fichier de configuration est le fichier `/usr/lib/X11/xdm/xdm.config`. On peut l'essayer en lancant `xdm` en tant que root. Pour le lancer automatiquement au démarrage du système, il faut enlever le commentaire devant la ligne faisant référence à `xdm` dans le fichier `/etc/inittab` et mettre le niveau par défaut :

```
# default runlevel
id:2:initdefault:
```

égal à celui indiqué par la ligne `xdm` (si `xdm` se lance au niveau d'exécution 4, il faut que le niveau par défaut soit 4 et non 2). Pour plus de précisions, voir section 9.12 page 237 ou taper `man inittab`.

Comme la modification du fichier `/etc/inittab` est dangereuse, on peut passer directement au niveau souhaité, ce qui aurra pour effet de lancer les commandes spécifiées pour le niveau. Ainsi, si `xdm` se lance en mode 4, la commande `init 4` terminera les processus en cours et lancera `xdm`. Comme `init` attend 20 secondes avant d'arrêter tous les processus, on peut indiquer qu'on veut y passer tout de suite :

```
init -t0 4
```

Attention ! Dans cet exemple, on a pris un niveau 4 pour `xdm`. Il se peut que ce ne soit pas le niveau 4 qui soit spécifié dans le fichier `/etc/inittab`. Il faut donc bien **spécifier le niveau d'exécution indiqué dans le fichier** `/etc/inittab`...

2.3 L'émulateur de terminal xterm

Taper sur `Ctrl` et l'un des boutons de la souris permet d'afficher les menus de la fenêtre. On peut par exemple activer le mode *Secure Keyboard* pour entrer un mot de passe, enlever la barre d'ascenseur...

Taper `Ctrl-D` lorsque le pointeur de la souris se trouve sur la fenêtre la ferme. De plus, noter que l'option *-title* est commune à la plupart des applications X, ainsi que l'option *-help*. La commande `showrgb` affiche la liste des couleurs disponibles (`showrgb | less` est plus "praticable"...). Cette commande affiche le contenu du fichier `/usr/X11R6/lib/X11/rgb.txt`.

Option	Signification
-ge 64x20+200+0	Taille de 64x20 affiché en (200,0).
-fg violet	Couleur de devant violet (charactères pour une fenetre xterm).
-bg yellow	Couleur de fond jaune.
-cr blue	Curseur bleu.
-fn 7x13bold	Police de taille 7x13 en gras.
-sb	Barre d'ascenseur sur le coté gauche.
-bd blue	Couleur de contour bleue.
-bw 100	Largeur du contour de 100 pixels.
-iconic	Démarre comme une icône.
-name Application	Associe le nom “Application” à la tâche.
-title Application	Associe le titre “Application” à la fenêtre.
-ls	Option *login shell*. Taper `logout` pour sortir de la fenêtre. Le fichier de configuration sera exécuté à l'ouverture de la fenêtre.

TAB. 2.1 – Options courantes de la fenêtre **xterm**.

2.4 L'horloge xclock

Option	Signification
-digital	Affiche une horloge numérique.
-analog	Affiche une horloge analogique (par défaut).
-padding 10	Distance de 10 pixels entre l'horloge et la fênetre qui lui est affectée (5 pixels par défaut).
-chime	Emet un bip toutes les demi-heures et deux bips toutes les heures.
-hd red	Affiche en rouge les aiguilles.
-hl blue	Affiche en bleu le contour des aiguilles.
-up 1	Délai en secondes entre chaque changement d'affichage d'aiguille (une troteuse s'affiche en dessous de 30 s).

TAB. 2.2 – Options courantes de l'horloge **xclock**.

2.5 L'horloge oclock

Option	Signification
-minute red	Affiche en couleur rouge l'aiguille des minutes.
-hour blue	Affiche en couleur bleue l'aiguille des heures.

TAB. 2.3 – Options courantes de l'horloge **oclock**.

2.6 La calculatrice xcalc

Option	Signification
-rpn	Notation en polonaise inverse.
-stipple	Affiche en mêmes couleurs que la fenêtre de base.

TAB. 2.4 – Options courantes de la calculatrice **xcalc**.

Les fichiers suivants sont les fichiers de configuration de `xcalc` :
`/usr/X11R6/lib/X11/app-defaults/XCalc` et
`/usr/X11R6/lib/X11/app-defaults/XCalc-color`
On peut bien-entendu les modifier comme on le souhaite. D'une manière générale, les fichiers de configuration d'une application X sont situés dans le répertoire :
`/usr/X11R6/lib/X11/app-defaults/`

2.7 Le vérouilleur d'écran xlock

Option	Signification
-mode bounce	Affiche le mode "bounce".
-nolock	Pas de demande de mot de passe.
-delay 8000	Vitesse d'animation à 8000 microsecondes.
-batchcount 3	Affiche 3 objets au maximun sur l'écran.

TAB. 2.5 – Options courantes du vérouilleur d'écran **xlock**.

Parmis les modes d'affichage, les suivants méritent dêtre essayés[2] : *bounce, bat, cartoon, clock, eyes, galaxy, hyper, image, nose, pyro, world.*

[2]Il y en a plus de 40 au total.

Pour avoir la liste des autres modes, on peut invoquer `xlock -help`, que l'on peut visualiser avec la commande `less` (`xlock -help | less`) ou qu'on peut rediriger vers un fichier (`xlock -help >& xlock.help`, attention à la syntaxe particuière de cette redirection).

2.8 La personnalisation du fond d'écran avec xsetroot

La commande `xsetroot` permet de définir la couleur du fond de l'écran (*root*) avec l'option *-solid*. Par exemple, pour une couleur "SteelBlue" (définie dans le fichier `rgb.txt` du répertoire `/usr/lib/X11`) :

```
xsetroot -solid SteelBlue &
```

Pour avoir un écran en gris simple, utiliser l'option *-gray* :

```
xsetroot -gray &
```

et pour afficher une image au format bitmap (*.xbm), utiliser l'option *-bitmap*

```
xsetroot -bitmap /usr/include/X11/bitmaps/xlogo64 &
```

La commande `xv` peut afficher des images de plusieurs formats en fond d'écran :

```
xv -root -quit -max image1.gif
```

Enfin, on peut spécifier la forme du curseur lorsqu'il se trouve sur le fond de l'écran, avec l'option *-cursor* (pour y associer une image bitmap) ou *-cursor_name* (pour y associer une des formes prédéfinies par un fichier bitmap, voir la page de manuel de **fvwm**, par exemple). Pour avoir un curseur en forme de flèche blanche pointant vers le coin supérieur gauche de l'écran :

```
xsetroot -cursor_name left_ptr -fg white -bg black &
```

La commande suivante affiche la ligne de la page de manuel de **fvwm** contenant l'information sur l'image "left_ptr" ainsi que son numéro, les autres noms de formes prédéfinies devraient être contenues dans les lignes alentours[3] :

```
man fvwm | grep -n left_ptr
```

[3]Normalement, ces noms sont spécifiés dans le fichier */usr/include/X11/cursorfont.h*, noms auquels il faut enlever du nom le préfixe "XC_".

2.9 L'aboyeur d'arrivée de mail xbiff

Le tableau 2.6 présente les principales options de `xbiff`.

Option	Signification
emptyPixmap	Permet de définir le dessin à afficher lorsque la boîte aux lettres est vide.
fullPixmap	Permet de définir le dessin à afficher lorsque la boîte aux lettres est remplie.
volume	Permet de définir le volume du signal sonore lorsque un courrier arrive (30 par défaut).

TAB. 2.6 – Options courantes de l'aboyeur d'arrivée de mail **xbiff**.

Parmis les options de `xbiff`, il y a celles qui concernent le dessin affiché, qu'on peut changer (voir quelles images sont disponibles dans le fichier `/usr/include/X11/bitmaps/` ou `/usr/local/include/X11/bitmaps/`). Il peut être très intéressant de le spécifier dans le fichier `$HOME/.Xdefaults` comme suit (voir la section 2.13 page 60 concernant les ressources) :

```
xbiff*emptyPixmap : mailempty
xbiff*fullPixmap : mailfull
xbiff*volume : 20
```

2.10 Le lecteur de boîte aux lettres xmh

`xmh` est une interface graphique du programme `mh`. Ce dernier doit donc être installé. Les spécifications suivantes dans le fichier `$HOME/.Xdefaults` (voir la section 2.13 page 60 concernant les ressources) peuvent s'avérer être utiles :

```
Xmh*geometry : 675x700
Xmh.clip*compGeometry : 675x500
```

Pour lire un message, sélectionner l'option `Incorporate New Mail` dans le menu `Table of Contents` (ou appuyer sur `Shift-Alt-I`). Pour effacer ce message, sélectionner l'option `Delete` dans le menu `Message` (ou appuyer sur `Alt-d`). Cette supression doit être validée par l'option `Commit Changes` dans le menu `Table of Contents` (ou `Shift-Alt-C`). Pour lire le message suivant, sélectionner l'option `View Next Message` dans le menu `Message` (ou appuyer sur `Alt-Barre d'espacement-n`). Pour répondre à un message, sélectionner l'option `Reply` dans le menu `Message` (ou appuyer sur `Alt-r`).

Pour envoyer un nouveau message, sélectionner l'option `Compose Message` dans le menu `Message`.

On peut créer une nouvelle boîte aux lettres avec l'option `Create Folder` dans le menu `Folder`. Entrer un nom et valider, puis cliquer sur son nom de la nouvelle boîte aux lettres (dans l'ovale). L'avantage des plusieurs boîte aux lettres est que l'on peut y classer les courriers. Pour y déplacer un message, choisir `Move` puis `Commit Changes` dans le menu `Messages`. On peut vérifier que le message a bien été déplacé par l'option `Open Folder` dans le menu `Folder`.

Enfin, il est possible de filtrer les messages grâce à l'option `Pick` du menu `Sequence`. On fournit la chaine de caractères que doivent contenir les messages dans leurs sujets, leurs dates...et on indique un nom significatif (boîte "*Creating sequence*") pour retrouver facilement cette séquence par la suite.

2.11 La spécification des polices de caractères

La commande `xset -q` permet d'afficher où se situent les répertoires contenant les fichiers de configuration des polices. Ce sont les répertoires qui apparaissent après "Font Path :" :

```
Font Path:
/usr/X11R6/lib/X11/fonts/misc, /usr/X11R6/lib/X11/fonts/75dpi,
/usr/X11R6/lib/X11/fonts/100dpi
```

Dans chacun de ces répertoires se trouvent au moins deux fichiers : `fonts.alias` et `fonts.dir`.

Le fichier `fonts.alias` s'édite à la main. Il spécifie l'alias d'une police. Des jokers sont permis ('*') dans la syntaxe du nom de la police. Le nom d'une police contient plusieurs champs. L'ordre de ces champs est le suivant :
- nom de marque.
- famille, style ;
- largeur du trait.
- pente (**r** pour romain, **i** pour italique, **o** pour oblique).
- nombre moyen de caractères par ligne.
- dimension du pixel.
- dimension du point[4] (x10).
- résolution verticale et horizontale.

[4] Il y a 72 points par pouce.

– espacement, proportionnel ou constant (comme **c** ou **m**).
– largeur moyenne des caractères en pixels (x10).
– nom standard international.
– code.

Par exemple, la police :

```
-misc-fixed-medium-r-normal--7-50-100-100-c-50-iso8859-1
-misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
```

On peut définir un nouvel alias dans ce fichier pour pouvoir l'utiliser ensuite pour une quelconque application, par exemple :

```
foxapoil -misc-fixed-medium-r-normal--7-50-100-100-c-50-iso8859-1
-misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
```

Pour utiliser une telle police, on peut par exemple essayer :

```
xterm -fn foxapoil
```

L'autre fichier de configuration des polices, `fonts.dir`, permet de change toutes les polices avec la commande `mkfontdir`.

On peut tester une police grâce à l'application `xfontsel`, à laquelle il faudra donner les caractéristiques comme expliqué ci-dessus pour le fichier `fonts.alias`.

On peut également trouver la liste complète de toutes les polices à l'aide de la commande `xlsfonts`. La liste des polices disponibles étant élevée, il peut être utile d'utiliser des jockers ('*' ou '?') :

```
xlsfonts -fn *-helvetica-bold-*-24-*
```

Le résultat devrait ressembler à ceci :

```
-adobe-helvetica-bold-o-normal--24-240-75-75-p-138-iso8859-1
-adobe-helvetica-bold-o-normal--24-240-75-75-p-138-iso8859-1
-adobe-helvetica-bold-r-normal--24-240-75-75-p-138-iso8859-1
-adobe-helvetica-bold-r-normal--24-240-75-75-p-138-iso8859-1
```

Les polices fixes disponibles par des alias sont accessibles en appliquant un filtre à la sortie de `xlsfonts` :

```
xlsfonts | grep [0-9]x[0-9]
```

Le résultat devrait ressembler à ceci :

```
10x20
12x24
12x24kana
```

```
12x24romankana
5x7
5x8
6x10
6x12
6x13
6x13bold
6x9
7x13
7x13bold
7x14
7x14bold
8x13
8x13bold
8x16
8x16kana
8x16romankana
9x15
9x15bold
```

Enfin, la commande `xfd` permet d'afficher une police particulière, par exemple :

```
xfd -fn *-helvetica-*-24-*
```

Note sur le presse-papier : l'application `xclipboard` permet de copier et coller un texte sélectionné dans une boite, celle qui résulte de son appel. Pour l'utiliser, on rentre un texte qu'on sélectionne avec la souris (boutton gauche) et qu'on colle (boutton du milieu). On peut sauvegarder le contenu du presse-papier ou simplement afficher son contenu par le simple appel de cette application.

2.12 Configurer le gestionnaire de fenêtres twm

twm est le gestionnaire de fenêtres le plus simple, mais aussi le plus archaïque distibué avec Linux. Il est inclut dans toutes les distributions, mais il évolue moins que d'autres gestionnaires de fenêtres.

Le gestionnaire de fenêtres **twm** cherche les fichiers `.twmrc` (dans le répertoire personnel) ou `system.twmrc` (dans l'un des répertoires suivants : `/usr/lib/X11/twm`, `/usr/X11R6/lib/X11/twm` ou `/etc/X11/twm`). Pour l'utiliser, il faut l'appeler dans le fichier `.xinitrc` **à la dernière ligne**. Voici un exemple simple de fichier `.xinitrc` qui initialise un serveur X et qui appelle **twm** au démarrage :

```
xterm -ge 64x20+200+0 -sb -fn 9x15bold &
oclock -ge +20+250 &
exec twm
```

Il faut avant toute chose récuperer un fichier de configuration, par exemple le fichier `system.twmrc` qu'on peut copier dans son répertoire personnel et changer les droits d'écriture :

```
cp /usr/X11R6/lib/X11/twm/system.twmrc ~/.twmrc
chmod 644 .twmrc
```

En cliquant sur le boutton droit de la souris lorsque le serveur X est lancé, un menu apparaît, permettant de lancer plusieurs applications. Ce menu peut être configuré...

Quatre types de spécifications peuvent être données dans ce fichier, et dans un ordre **précis** : les variables, les fonctions, les associations de touches et les menus.

2.12.1 Les variables

Les variables peuvent de type *booléennes, numériques* ou *chaînes de caractères* :

– comme **variables booléenne**, il y a par exemple `NoGrabServer`, qui empêche le gestionnaire de fenêtres **twm** d'appeler le serveur X lorsqu'il manipule des boîtes de dialogue, ou `DecorateTransients` qui implique que les fenêtres de courte durée de vie (les boîtes de dialogue, par exemple) seront elles aussi décorées :

```
NoGrabServer
RestartPreviousState
DecorateTransients
```

– comme **variables numériques**, il y a par exemple `MoveDelta 3` qui reconnait un mouvement de la souris à partir du monent où le pointeur (de la souris) se déplace de plus de trois pixels :

```
MoveDelta 3
```

– comme **variables chaînes de caractères**, il y a par exemple le positionnement de quelques polices de caractères, comme `TitleFont` etc. Le contenu de la variable doit être entre guillemets :

```
TitleFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
ResizeFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
IconFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*"
```

– enfin il y a les variables définies par **groupes**, comme le groupe `Color`, par exemple. Le groupe est définit entre accolades :

```
Color
{
    BorderColor "slategrey"
    DefaultBackground "rgb:2/a/9"
    DefaultForeground "gray85"
    TitleBackground "rgb:2/a/9"
    TitleForeground "gray85"
    MenuBackground "rgb:2/a/9"
    MenuForeground "gray85"
    MenuTitleBackground "gray70"
    MenuTitleForeground "rgb:2/a/9"
    IconBackground "rgb:2/a/9"
    IconForeground "gray85"
    IconBorderColor "gray85"
    IconManagerBackground "rgb:2/a/9"
    IconManagerForeground "gray85"
}
```

2.12.2 Les fonctions utilisateur

Plusieurs fonctions internes sont définies (58 au total), comme `f.exec` qui exécute une commande Linux. On peut en définir de nouvelles par association de fonctions internes déjà existantes :

```
Function "move-or-lower" { f.move f.deltastop f.lower }
Function "move-or-raise" { f.move f.deltastop f.raise }
Function "move-or-iconify" { f.move f.deltastop f.iconify }
```

Pour `move-or-raise` par exemple, si le pointeur de la souris s'est déplacé de plus de `MoveDelta`, la fonction `f.move` sera exécutée et `f.deltastop` terminera la séquence (ici `f.raise` ne sera pas exécutée). Sinon, c'est `f.raise` qui sera éxécutée.

2.12.3 Les associations de touches

On peut associer le déclenchement d'une fonction avec la pression d'un bouton de souris ou une association de touches. La forme générale d'une telle association est la suivante :

```
Bouton ou Touche = modlist : context : fonction
```

modlist désigne la touche accompagnant le pression du bouton : **shift**, **control**, **lock**, **meta** (touche *Alt*), **mod1** à **mod5**. Ces touches peuvent être abrégées respectivement par **s, c, l, m, m1, m2...m5**. S'il y a plusieurs touches d'accompagnement, on les sépares par une barre verticale.

context désigne l'endroit où doit se trouver le pointeur de la souris pour que cette association soit valide : **window** pour la fenêtre, **title** pour la barre de titres, **icon** dans une icône, **root** sur l'arrière plan, **frame** sur le contour, **iconmgr** sur le gestionnaire d'icônes, **all** dans n'importe quel endroit possible.

Enfin, **fonction** désigne la fonction (interne ou définie par l'utilisateur) à exécuter.

```
Button1 = : root : f.menu "defops"

Button1 = m : window|icon : f.function "move-or-lower"
Button2 = m : window|icon : f.iconify
Button3 = m : window|icon : f.function "move-or-raise"

Button1 = : title : f.function "move-or-raise"
Button2 = : title : f.raiselower

Button1 = : icon : f.function "move-or-iconify"
Button2 = : icon : f.iconify

Button1 = : iconmgr : f.iconify
Button2 = : iconmgr : f.iconify
```

On peut par exemple programmer la conversion d'une fenêtre en icône ou vice-versa (`f.iconify`) par la pression de la touche F1 ou l'affichage du menu racine (`f.menu "defops"`) par la pression de la touche F2 lorsque le pointeur se trouve sur une fenêtre :

```
''F1'' =: all : f.iconify
''F2'' = : window : f.menu "defops"
```

2.12.4 Les menus

Le menu racine accessible en cliquant avec le bouton gauche de la souris sur le fond de l'écran peut être configuré et personnalisé, ainsi que les sous-menus. Le menu `"defops"` contient les options par défaut :

```
menu "defops"
{
"Twm"   f.title
"Iconify"       f.iconify
"Resize"        f.resize
"Move"          f.move
"Raise"         f.raise
"Lower"         f.lower
""              f.nop
"Focus"         f.focus
"Unfocus"       f.unfocus
"Show Iconmgr"  f.showiconmgr
"Hide Iconmgr"  f.hideiconmgr
""              f.nop
"Kill"          f.destroy
"Delete"        f.delete
""              f.nop
"Restart"       f.restart
"Exit"          f.quit
}
```

Chaque section contient le texte à afficher dans la ligne correspondante du menu racine et la fonction qui lui est associé. Pour les lignes vides, destinés à l'espacement dans l'affichage, une fonction `f.nop` est prévue.

On peut re-programmer ce menu, en y ajoutant une section personnelle par exemple, comme la section `"Programmes"~`:

```
menu "defops"
{
"Twm"   f.title
"Programmes"    f.menu "ProgrammesMenu"
""              f.nop
"Iconify"       f.iconify
"Resize"        f.resize
"Move"          f.move
"Raise"         f.raise
"Lower"         f.lower
```

```
""              f.nop
"Focus"         f.focus
"Unfocus"       f.unfocus
"Show Iconmgr"  f.showiconmgr
"Hide Iconmgr"  f.hideiconmgr
""              f.nop
"Kill"          f.destroy
"Delete"        f.delete
""              f.nop
"Restart"       f.restart
"Exit"          f.quit
}
```

La section `"Programmes"` est accessible en glissant le pointeur vers la droite sur le nom. C'est alors le sous-menu `"ProgrammesMenu"` (invoqué par la fonction `f.menu`) qui sera affiché, avec toutes les commandes définies par le menu `"ProgrammesMenu"` :

```
menu "ProgrammesMenu"
{
"xearth"          f.exec "xearth -label -grid -markerfile 9x15bold &"
"Mise en veille"  f.menu "Mise en veilleMenu"
"Jeux"            f.menu "JeuxMenu"

}
```

Le menu `"ProgrammesMenu"` peut à son tour contenir d'autre sous-menus, décrits par la suite :

```
menu "Mise en veilleMenu"
{
"bat"             f.exec "xlock -nolock -mod bat &"
"bounce"          f.exec "xlock -nolock -mod bounce &"
"world"           f.exec "xlock -nolock -mod world &"
}

menu "JeuxMenu"
{
"xabuse"          f.exec "xabuse &"
}
```

2.13 Les ressources

Les ressources permettent de spécifier un ensemble d'options communes à une application. Les caractéristiques communes à cette ressource particulière sont définies par des *classes* et spécifiées dans le fichier `$HOME/.Xdefaults` (ou le fichier de configuration général `/usr/X11R6/lib/xinit/.Xresources`, auquel cas les données s'ajoutent à celles du fichier `$HOME/.Xdefaults`).

2.13.1 Exemple d'utilisation

Par convention, le nom de la classe porte le nom du programme avec la première lettre en majuscule[5](les deux premières lettres pour un programme commencant par la lettre 'X'). Par exemple, si on veut définir la couleur des aiguilles et du fond de trois horloges, on peut par exemple mettre dans le fichier `$HOME/.Xdefaults` les lignes suivantes :

```
! Specification de la valeur concernant la classe XClock.
! Initialisation des classes de ressources Background, Foreground...
! pour l'ensemble de la classe XClock.

XClock*Background : yellow
XClock*Foreground : red
XClock*hands : green
XClock*Font : 7x13bold
```

On peut ensuite donner des spécifications spéciales à chacunes des horloges, par exemple leur positions et leurs tailles (toujours dans le fichier `$HOME/.Xdefaults`, à la suite) :

```
! Definitions specifiques a chaque xclock
! Le specifications de chaque xclock l'emportent sur celles de la classe

xclock-haut*Geometry : 160x170-0-0

xclock-milieu*Geometry : 160x170-200-0
xclock-milieu*Foreground : blue

xclock-3*Geometry : 160x170-400-0
```

Pour appeler ces horloges, on peut insérer les lignes suivantes dans le fichier d'initialisation `.xinitrc` :

[5]Une classe peut agir sur plusieurs programmes. Ainsi, la classe **Clock** permet de spécifier des options pour les horloges **xclock** et **oclock**.

```
.xinitrc+
```

Trois horloges apparaîtrons en bas de l'écran avec les couleurs, tailles et positions stipulées par les fichiers de configuration.

2.13.2 Pour avoir des informations sur les ressources

Pour vérifier que la base de données de ressources X a bien été modifiée, on peut utiliser la commande `xrdb` avec l'option *-query* :

```
xrdb -query
```

On obtient alors la modification des dernières ressources, dont celles concernant **XClock** :

```
XClock*Background:      yellow
XClock*Foreground:      red
XClock*hands:   green
xclock-haut*Geometry:   160x170-0-0
xclock-milieu*Geometry: 160x170-200-0
xclock-3*Geometry:      160x170-400-0
```

Sans argument, cette commande invite à cliquer sur le programme dont on veut obtenir des informations. On peut aussi obtenir des informations en indiquant l'index. Pour obtenir l'index, on lance la commande `xwininfo` et on clique sur le programme en question. On a alors l'index :

```
Window id: 0x100000a "xclock-haut"
```

On peut alors avoir des informations sur le programme :

```
xwininfo: Window id: 0x100000a "xclock-haut"

  Absolute upper-left X:  862
  Absolute upper-left Y:  596
  Relative upper-left X:  0
  Relative upper-left Y:  21
  Width: 160
  Height: 170
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x21 (installed)
  Bit Gravity State: NorthWestGravity
  Window Gravity State: NorthWestGravity
```

```
Backing Store State: NotUseful
Save Under State: no
Map State: IsViewable
Override Redirect State: no
Corners:  +862+596  -2+596  -2-2  +862-2
-geometry 160x170-0-0
```

2.13.3 L'éditeur de ressources editres

`editres` est utilisé pour le développement des applications X et permet d'afficher un arbre montrant les dépendances des *widgets* (menu, boîte de dialogue, bouton, ascenseur...). Pour l'utiliser, il suffit de l'appeller et d'aller dans le menu **Commands** et cliquer sur **Get Tree**, puis cliquer sur le programme dont on veut des informations. Un très bon exemple est `editres` lui même[6].

On pout avoir des informations sur l'application **xclock-haut** par exemple. L'arbre s'affiche alors, et pour changer les paramètres d'un widget, on clique dessus et on invoque la commande **Show Resource Box** du menu **Commands**. On a alors accès à toutes les ressources relatives au widgets en question, dont on peut modifier les valeurs et tester aussitôt les changements (bouton **"Apply"**) et même sauvegarder dans un fichier à préciser (normalement le fichier `$HOME/.Xdefaults`).

Par exemple, si on décide d'affecter une couleur violette aux aiguilles de l'horloge `xclock-haut`, on modifie la spécification de ressource `hands` et on sauvegarde le résultat dans le fichier `$HOME/.Xdefaults`. La modification sera alors écrite sous la forme :

```
.xclock-haut.clock.hands: violet
```

On peut également faire des modifications de ressources en appelant le programme avec l'option *-xrm* :

```
xclock -xrm 'XClock*hands : violet' &
```

La modification portera sur le classe `hands`, dont on pourra tester les modifications avant de décider du choix définitif.

2.13.4 Ressources X communes

Par convention, toutes les applications X ont en commun un ensemble standard de ressources. Celles-ci comprennent des paramètres tels que couleurs, taille et position des fenêtres et polices. En voici une liste :

[6] Ce programme est assez étendu, utiliser le rectangle panoramique avec la souris pour se déplacer dans l'arbre.

background couleur de fond (option dans la ligne de commande : **-bg** ou **-background**).

BorderColor couleur de bordure de la fenêtre (option dans la ligne de commande : **-bd** ou **-border**).

borderWidth largeur de la bordure en pixels (option dans la ligne de commande : **-bw** ou **-borderwidth**).

Display nom de l'affichage dans lequel doit apparaître la sortie d'une application (option dans la ligne de commande : **-d** ou **-display**).

foreground couleur de premier plan (option dans la ligne de commande : **-fg** ou **-foreground**).

Font nom de la police de caractères (option dans la ligne de commande : **-fn** ou **-font**).

Geometry taille et position de la fenêtre (option dans la ligne de commande : **-ge** ou **-geometry**).

Title chaîne du titre (option dans la ligne de commande : **-title**).

Pour l'option *-display* : par exemple, `xclock -display zecastor:0 &` si on veut lancer la commande **xclock** depuis le système ayant pour nom *zecastor* (affiché par la commande **hostname** ou variable d'environnement **$HOSTNAME**.

Pour l'option *-geometry* : ce qui suit est de la forme **80x25+10-10** par exemple, c'est à dire 80 pixels de largeur et 25 pixels de hauteur, affiché à 10 pixels du coté *haut* de l'écran (coté *bas* si négatif) et à 10 pixels du coté *droit* de l'écran (coté *gauche* si positif), pour une fenêtre **xterm** par exemple. Le point (0, 0) n'est pas situé en haut à gauche pour toutes les applications.

2.14 Configurer un autre gestionnaire de fenêtres : fvwm

fvwm est un gestionnaire de fenêtres beaucoup plus maniable que twm, plus fontionnel et surtout moins gourmand en mémoire. Le fichier de configuration est `$HOME/.fvwmrc`, et le fichier de base est le suivant :

```
usr/X11R6/lib/X11/fvwm2/system.fvwm2rc
```

Celui-ci est un bon point de départ.

2.14.1 Configuration générale

Le début de la configuration commence avec les couleurs et les polices de caractères :

```
#-------------------------------------------------#
# Choix des couleurs et des polices de caracteres #
#-------------------------------------------------#



# Couleur de forme (texte) et de fond (cadre de la fenetre)
# des fenetres inactives, des menus et du pageur.

# Les couleurs peuvent etre specifiees par leurs noms ou par leur
# nombre en hexadecimal.

# couleur des fenetres non actives

StdForeColor       white
#000000

StdBackColor       midnightblue
#60a0c0

# couleurs des fenetres actives

HiForeColor        Black

HiBackColor        red
#c06077

# couleurs du pager

StickyForeColor Black
StickyBackColor #60c0a0

# couleurs du menu

MenuForeColor           Black
MenuBackColor           grey

# Police utilisee dans la barre des menus
```

```
Font               -adobe-helvetica-medium-r-normal-*-*-120-*

# Police utilisee dans la barre de titres

WindowFont         -adobe-helvetica-bold-r-normal-*-*-120-*

# Police utilisee dans les icones

IconFont        fixed
```

On peut ensuite configurer le bureau virtuel :

```
#-----------------------------#
# Parametres de configuration #
#-----------------------------#


# Pour faire passer en haut de la pile la fenetre rendue active
# apres 150 ms.

AutoRaise 150

# Cliquer sur la fenetre au lieu de deplacer la souris sur la fenetre
# pour l'activer (c'est a dire pour qu'elle puisse recevoir des entrees
# du clavier).
# En commentaires : c'est le deplacement du curseur sur la fenetre qui
# determine la selection d'une fenetre, au lieu d'un clic sur le boutton
# gauche pour l'activer.

# ClickToFocus

# Emulation de mwm

# MWMFunctionHints
# MWMHintOverride
# MWMDecorHints
# OpaqueMove 0

# les fichiers tels que 4Dwm.fvwmrc ou mwm.fvwmrc emule un gestionnaire
de fenetre comme 4Dwm ou mwm. Il sont en general situes dans le
```

```
# repertoire /usr/X11R6/lib/X11/fvwm. Pour les utiliser, il faut les
# copier dans ce fichier (les remplacer par celui-ci), puis relancer
# fvwm...

# Epaisseur en pixels du cadre entourant la fenetre
# (6 par defaut)

BoundaryWidth      3

# Configuration du bureau virtuel

# Six ecrans de large sur six ecrans de haut

DeskTopSize  6x6

# Echelle de reduction dans la vue d'ensemble, le pageur
# (ici 1/50eme)

DeskTopScale 50

# Mise en service du pageur dans le coin inferieur droit, a 10 pixels
# de chaque bord

Pager        5 -5

# Pourcentage de defilement lorsque la souris a atteint le bord
# de l'ecran (100 100 = ecran entier)

EdgeScroll 10 10

# Temps et profondeur avant deplacement (en ms)

EdgeResistance 10 10

# Placement aleatoire des fenetres dont la position n'est pas specifiee

# RandomPlacement

# Pour eviter les maux de tete

NoPPosition
```

```
# Taille maximum (en pourcentage de la taille du bureau) des fenetres qui
# seront deplacees en solide.
# 0 : toutes les fenetres seront avec des contours elastiques
# 100 : toutes les fenetres seront deplacees en fenetres solides

OpaqueMove 0
```

Viennent ensuite les définition des caractéristiques de chaque fenêtre :

```
# Definitions des caracteristiques de chaque fenetre

# pour que Xbiff reste toujours visible au dessus des autres fenetres

Style "XBiff" StaysOnTop
```

Les options de `Style` sont les suivantes (tableau 2.7) :

Option	Signification
NoTitle	Supprime le titre (utilisé pour **xclock** ou **xbiff**, par exemple).
NoBorder	Elimine le cadre autour de la fenêtre.
Sticky	Colle la fenêtre à l'écran (elle apparaîtra toujours à la même place, quelque soit le bureau virtuel).
BoundaryWidth	Spécifie l'épaisseur en pixels du cadre entourant la fenêtre (6 par défaut).

TAB. 2.7 – Principales options de **Style**.

Il faut fournir en argument à `Style` des titres de fenêtres ou des classes d'applications.

Les fenêtres peuvent être mises sous la forme d'icônes. En voici les spécifications :

```
# Region de l'ecran ou les icones seront rangees
# coin-haut-gauche coin-bas-droite

IconBox -150 90 -5 -140

# Police de caracteres employee dans les icones

IconFont -adobe-helvetica-medium-r-*-*-*-120-*
```

```
#-----------------#
# Chemins d'acces #
#-----------------#


# Chemin d'acces des images bitmaps (XBM)
# On peut specifier plusieurs repertoires, comme
#
#    $HOME/bitmaps:/usr/X11R6/include/X11/bitmaps
#
# par exemple

IconPath /usr/X11R6/include/X11/bitmaps

# Chemin d'acces des images pixmaps (XPM)

PixmapPath /usr/X11R6/include/X11/pixmaps

# Chemin d'acces des modules

ModulePath /usr/lib/X11/fvwm/


#--------#
# Styles #
#--------#


# Image par defaut. Le chemin est relatif a IconPath ou PixmapPath,
# il peut etre donne en absolu

Style "*" BorderWidth 5, HandleWidth 5, Color Black/#60a0c0, Icon unknown.xpm
Style "Fvwm*" NoTitle, Sticky, WindowListSkip
Style "Fvwm Pager" StaysOnTop
Style "GoodStuff" NoTitle, NoHandles, Sticky, WindowListSkip, BorderWidth 0
Style "*lock" NoTitle, NoHandles, Sticky, WindowListSkip
Style "Maker" StartsOnDesk 1
Style "signal" StartsOnDesk 3
Style "rxvt" Icon term.xpm
```

```
Style "ddd" StartsOnDesk 2


# Images attribuees a differentes applications

Style "XTerm" Icon xterm.xpm, Color black/grey
# Style "Mosaic" Icon mosaic.xpm

Style "GoodStuff" Icon toolbox.xpm

# Dessin des boutons des fenetres

# Bouton 3 : losange.

# Bouton 5 : pentagone.

# Les coordonnees sont en pixels.

# @1 : couleur plus lumineuse   |
# | --> relief
# @0 : couleur plus sombre |

ButtonStyle : 3 50x35@1 65x50@0 50x65@0 35x50@1 50x35@1
ButtonStyle : 5 50x35@1 65x50@0 65x65@0 50x65@0 35x50@1 50x35@1
```

Si la version de `fvwm` est trop ancienne, `Style` n'est pas reconnu et la section `Icon` doit être modifiée comme suit :

```
Icon '''' unknown.xpm
Icon ''Xterm'' xterm.xpm
Icon ''Mosaic'' mosaic.xpm
```

La section suivante définit les fonctions. Chacune doit ensuite être associée à un bouton ou une touche du clavier dans la section *Association de boutons et de touches* plus loin dans le fichier.

```
#-----------#
# Fonctions #
#-----------#


# L'evenement associe a la fonction peut etre :
```

```
# * "I" : execute la fonction des que le gestionnaire de
#         fenetres est initialise;
# * "Click" : un simple click;
# * "DoubleClick" : un double click;
# * "Motion" : click en bougant la souris;

Function "InitFunction"
  Exec "I" exec rxvt -geometry +1200+10 &
        Module  "I"     GoodStuff
   Module "I" FvwmPager 0 3
  Module "I" FvwmWinList
EndFunction

Function "Move-or-Raise"
Move "Motion"
# Raise "Motion"
Raise "Click"
# RaiseLower "DoubleClick"
EndFunction

# Agrandit la taille de la fenetre :
# * de 80% en hauteur si on clique une fois
# * de 100% en hauteur si on clique en bougant le souris
# * de 100% en hauteur et 100% en largeur si on clique deux fois

Function "maxi_fenetre"
Maximize "Click" 0 80
Maximize "Motion" 0 100
Maximize "DoubleClick" 100 100
EndFunction
```

2.14.2 Configurer les menus

Un menu est délimité par `Popup` et `EndPopup`. Le nom du menu est associé à `Title`. Voici des exemples de menu :

```
#---------------------#
# Menus et sous-menus #
#---------------------#
```

```
# Sous-menus...

Popup "Xclients"
   Title "Xclients"
   Exec "xterm"    exec xterm &
   Nop  ""
   Exec "piou"     exec xterm -e rlogin $HOSTNAME -l piou &
   Nop  ""
   Exec "xcalc"    exec xcalc &
   Exec "xman"     exec xman &
   Exec "top"    exec xterm -ge 72x36 -font 7x14 -T Top -n Top -e top &
EndPopup

Popup "Module"
Title "Modules"
  Module "GoodStuff" GoodStuff
Module  "Identify"  FvwmIdent
  Module "Save" FvwmSave
  Module "Debug" FvwmDebug
   Module "Pager" FvwmPager 0 3
  Module "FvwmWinList" FvwmWinList
EndPopup

# Menu de manipulation des fenetres.

Popup "Fenetre_Opt"
   Title   "Options de la fenetre"
   Function "Deplacer" Move-or-Raise
   Function "Changer la taille" Resize-or-Raise
   Raise   "Au dessus"
   Lower   "Au dessous"
   Iconify "Iconifier"
   Stick   "Coller"
   Function "Maximum" maxi_fenetre
   Nop     ""
   Destroy "Detruire"
   Delete  "Supprimer"
   Nop     ""
   Refresh "Rafraichir l'ecran"
EndPopup
```

```
# Affiche un menu :
# * Xclients si on clique une fois
# * Module si on clique en bougant le souris
# Ferme la fenetre si on clique deux fois

Function "Opt_fenetre"
Popup "Click" Xclients
Popup "Motion" Module
Delete "DoubleClick"
EndFunction
```

Ces menus peuvent être insérés dans un autre (avec la fonction `Popup`), comme par exemple le menu principal :

```
# Menu principal (ou menu racine)

Popup "Fvwm"
   Title   "Fenetres"
   Move    "Deplacer"
   Resize  "Changer la taille"
   Raise   "Au dessus"
   Lower   "Au dessous"
   Iconify "Iconifier"
   Stick   "Coller"
   Nop     ""
   Popup   "Xclients" Xclients
   Nop     ""
   Popup   "Module" Module
   Nop     ""
   Destroy "Detruire"
   Delete  "Supprimer"
   Nop     ""
   Refresh "Rafraichir l'ecran"
   Exec    "Charger Xdefaults"  exec xrdb -load $HOME/.Xdefaults
   Restart "Relancer Fvwm"  fvwm
   Restart "Lancer twm"   twm
   Quit    "Quitter Fvwm"
EndPopup
```

2.14.3 Configurer les raccourcis

La commande `Mouse` permet d'associer des actions aux bouttons de la souris. En voici la syntaxe :

`Mouse` *bouton contexte modificateurs fonction*

où
- *bouton* vaut 1, 2, 3 (pour les boutons de gauche, du milieu et de droite) ou 0 (pour n'importe quel bouton).
- *contexte* spécifie la région dans laquelle ces actions prendront effet. En voici une liste :
  - **R** : fenêtre principale (en dehors de toute fenêtre ou icône).
  - **W** : toute fenêtre d'une application.
  - **S** : contour d'une icône.
  - **F** : coin d'un cadre.
  - **T** : barre de titres (en dehors des boutons).
  - **I** : fenêtre iconifiée.
  - **chiffre** : spécifie un bouton particulier de la barre de titres (entre 0 et 9, voir plus loin).
  - **A** : tout contexte (sauf le boutton de barre de titres).

  On peut combiner les possibilités ci-dessus, comme **TSIF**, par exemple. Les boutons apparaissent en haut à gauche et à droite de chaque fenêtre et portent un numéro impair pour ceux de gauche (1 3 5 7 9) et pair pour ceux situés sur la droite (0 8 6 4 2).
- *modificateurs* spécifie les combinaisons de touches associées aux boutons de la souris. Les valeurs sont :
  - **C** : `Ctrl` ;
  - **M** : `Meta` (touche `Alt` , en général) ;
  - **S** : `Shift` ;
  - **N** : aucune touche ;
  - **A** : n'importe laquelle de ces touches ;

En pratique, cela donne :

```
#-----------------------------------#
# Association de boutons et de clefs #
#-----------------------------------#


# Fond d'ecran (root)

# Bouton Contexte Modificateur Fonction
```

```
Mouse 1       R          N      PopUp "Fvwm"
Mouse 2       R          N      PopUp "Xclients"
Mouse 3       R          N      WindowList

# Boutons de barres de titres
# 1 3 5 7 9  0 8 6 4 2

Mouse 1       1          N      PopUp "Fvwm"
Mouse 1       3          N      Iconify
Mouse 1       4          N      Destroy
Mouse 1       2          N      Resize
Mouse 0       2          A      Function "maxi_fenetre"
Mouse 0       0  C Function "Opt_fenetre"
Mouse 1       T  N Function "Move-or-Raise"
Mouse 1       5          N      WindowList
```

Enfin, on peut associer des actions à certaines touches du clavier grâce à la commande `Key` (similaire à `Mouse`). En voici la syntaxe :

`Key` *touche contexte modificateurs fonction*

Par exemple, si on veut se déplacer d'un bureau virtuel avec la combinaison $\boxed{\texttt{Ctrl-}\textit{touche fléchée}}$:

```
# Racourcis clavier

Key   Up       A         C      Scroll +0 -100
Key   Down     A         C      Scroll +0 +100
Key   Left     A         C      Scroll -100 +0
Key   Right    A         C      Scroll +100 +0
```

On peut aussi utiliser les touches **F1, F2. . .**.

2.14.4 Configurer les modules

Un module est un programme très particulier qui ne peut être exécuté que sur un appel depuis le gestionnaire de fenêtres et qui tranmet une commande à exécuter. Une ligne concernant un module commence par une étoile (*). Cette ligne peut définir un attribut du module (couleur, géométrie, police de caractères. . .) ou valider un programme à exécuter. Dans l'exemple qui suit, des fonctions et des menus utilisés par les modules sont d'abord définits :

```
#---------#
# Modules #
#---------#


# Fonctions et menus utilisees par les modules

Function "barthiswmpopupfunc"
Popup "I" "Xclients"
EndFunction

Popup "bargraphicpopup"
Title "Graphics"
  Exec "XFig" xfig
  Exec "Gimp" gimp
   Exec "XPaint" xpaint
  Exec "TGif" tgif
  Exec "ImageMagick" xterm -iconic -e display
EndPopup

Popup "bartoolspopup"
Title "Tools"
 # take knews, if not trn, if not, nothing
 # user installed software
  Exec "TkDesk" tkdesk
   Exec "TkMan" tkman
  Exec "Xman" xman

  Nop ""
  Function "Netscape" NSFunc
EndPopup

Popup "barshellpopup"
Title "Shells"
  Exec "Shell in XTerm" xterm -e bash -login
  Function "Root shell" RootShell
EndPopup

# Cadre de GoodStuff (une sorte de bouton a barre).

*GoodStuffFont 6x13
```

```
*GoodStuffFore Black
*GoodStuffBack grey67
*GoodStuffGeometry +0+0

# Attention : specifier rows (lignes) ou columns (colones),
# pas les deux.

*GoodStuffRows 1

# Boutons places en ligne. Si on clique dessus, la commande associee
# s'execute.

*GoodStuff Kill rbomb.xpm Destroy
*GoodStuff Xcalc rcalc.xpm Exec "Calculatrice" xcalc &

# Identification de la fenetre des modules

*FvwmIdentBack MidnightBlue
*FvwmIdentFore Yellow
*FvwmIdentFont -misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1

# Bureau virtuels multiples.

*FvwmPagerBack pink
*FvwmPagerFore #908090
*FvwmPagerFont -misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
*FvwmPagerHilight #cab3ca
*FvwmPagerGeometry -1+380
*FvwmPagerLabel 0 "Bureau Principal"
*FvwmPagerLabel 1 Mail
*FvwmPagerLabel 2 Programmation
*FvwmPagerSmallFont -misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1

# Liste des fenetres ou icones actuellement ouvertes sur toutes les pages
# sur tous les bureaux virtuels.

# La liste est affichee par ordre chronologique.

*FvwmWinListBack #908090
*FvwmWinListFore Black
*FvwmWinListFont -misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
```

```
*FvwmWinListAction Click1 Iconify -1,Focus
*FvwmWinListAction Click2 Iconify
*FvwmWinListAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmWinListUseSkipList
*FvwmWinListGeometry +0-150

# Fonctions et menus utilisees par les modules

Function "bargraphicpopupfunc"
  Popup "I" bargraphicpopup
EndFunction


Function "bartoolspopupfunc"
  Popup "I" bartoolspopup
EndFunction


Function "barshellpopupfunc"
  Popup "I" barshellpopup
EndFunction


Function "MenuOrIconify"
  Popup "Click" WindowOps
  Raise "Motion"
  Move "Motion"
  Iconify "DoubleClick"
EndFunction

Popup "bargraphicpopup"
Title "Graphics"
  Exec "XFig" xfig
  Exec "Gimp" gimp
   Exec "XPaint" xpaint
  Exec "TGif" tgif
  Exec "ImageMagick" xterm -iconic -e display
EndPopup

Popup "bartoolspopup"
Title "Tools"
```

```
 # take knews, if not trn, if not, nothing
 # user installed software
  Exec "TkDesk" tkdesk
   Exec "TkMan" tkman
  Exec "Xman" xman

  Nop ""
  Function "Netscape" NSFunc
EndPopup

Popup "barshellpopup"
Title "Shells"
  Exec "Shell in XTerm" xterm -e bash -login
  Function "Root shell" RootShell
EndPopup

# Quelque boites affichant des menus...

*GoodStuff Graphics... palette3_3d.xpm Function "bargraphicpopupfunc"
*GoodStuff Tools... box_full_3d.xpm Function "bartoolspopupfunc"
*GoodStuff Shells... Shell.xpm Function "barshellpopupfunc"
```

2.15 Configurer openwin

Les gestionnaires de fenêtres **olwm** et **olvwm** sont concus pour les stations de travail Sun. **olvwm** est l'extension de **olwm**, avec des bureaux virtuels (comme l'indique le 'v').

Pour lancer ce gestionnaires de fenêtres, il faut invoquer le script `openwin`, dans le répertiore `$OPENWINHOME/bin`. Ce script lance à son tour le script `startx`, situé dans le répertoire `$OPENWINHOME/lib` (si ce script est absent, tout `startx` placé dans la variable d'environnement **$PATH** sera exécuté). L'argument passé à `startx` est le fichier de ressources `Xinitrc`, situé dans le répertoire `$OPENWINHOME/lib`. Le fichier `Xinitrc` appele à son tour les fichiers suivants :

1. le fichier `.Xresources`.
2. le fichier `.Xdefaults`.
3. le fichier `.Xmodmap`.
4. le fichier `.xinitrc`[7] ;

[7]Il y a un risque de confusion pour ce fichier *$HOME/.xinitrc*, utilisé pour initialiser

5. éventuellement le fichier d'initialisation de l'écran `.openwin-init` ;

Ce fichier script `Xinitrc` a pour tâche :
– l'affichage d'une première image.
– l'identification des fichiers de ressources.
– une éventuelle reprogrammation de certaines touches du clavier.
– le lancement éventuel des fichiers de configuration `.openwin-init` et `.xinitrc`.
– le lancement du gestionnaire de fenêtres `olwm` ou `olvwm`.

Seuls l'aspect général de l'écran (par le fichier `.openwin-init`, fichier de sauvegarde du bureau virtuel) et le menu émergent (par `.openwin-menu-*`) sont configurables. Le reste ne peut être configuré qu'avec les ressources X. Le programme `properties` permet de modifier de facon interactive (par le biais d'une interface graphique) les caractéristiques du bureau (couleurs, curseur, déplacement des icônes. . .).

On peut appeller simplement ce gestionnaire de fenêtres par la commande `openwin` ou par la commande `startx`, après avoir édité un fichier `.xinitrc` comme suit (ce fichier sert à lancer avec `startx`) :

```
#!/bin/sh

xterm -bg pink -fg red -sb -fn 9x15bold -ge -2-2 &
xclock -ge -2-2 -bg yellow -hd green &
exec olvwm
```

La commande `openwin` exporte et/ou positionne six variables d'environnement :
– les chemins d'accès pour les ressources des applications X : **$XAPPLRESDIR**.
– les fichiers du manuel : **$MANPATH**.
– les fichiers d'aide : **$HELPPATH**.
– les fichiers de base d'Openwin : **$OPENWINHOME**.
– le système X : **$X11HOME**.
– le nom du gestionnaire de fenêtres à utiliser (qui peut être `olwm` ou `olvwm`) : **$WINDOWMANAGER**.

Le fichier de sauvegarde du bureau virtuel `.openwin-init` peut être édité manuellement ou crée par openwin (dans le menu *Workspace/Utilities* la commande *SaveWorkspace*). En voici un exemple :

```
$OPENWINHOME/bin/cmdtool -Wp 225 0 -Ws 590 77 -C &
```

un serveur X, et ici utilisé comme fichier de sauvegarde du bureau virtuel. Il vaut mieux n'utiliser que le fichier *.openwin-init* pour configurer le bureau.

```
$OPENWINHOME/bin/workman -Wp 0 150 -Ws 590 300 &
$X11HOME/bin/xeyes -display :0 -ge 100x100+500+600 &
$X11HOME/bin/xclock -display :0 -d -ge 180x40+830+0 -fn 9x15bold
-upda 1 &
```

Les arguments *-Wp* et *-Ws* sont spécifiques à openwin et précisent respectivement la position et la taille de la fenêtre. Pour avoir de l'aide, taper, par exemple (en provoquant volontairement une faute) :

```
shelltool -Wp
```

ou encore :

```
shelltool -WH > aide.openwin
```

Pour configurer les menus et sous-menus, olwm (ou olvwm) cherche le fichier `$HOME/.openwin-menu` ou, si celui-ci n'existe pas, le fichier système (`$OPENWINHOME/lib/openwin-menu`). En voici un exemple :

```
"Workspace"       TITLE
"Shells "         MENU  $OPENWINHOME/lib/openwin-menu-s
"Editors "        MENU  $OPENWINHOME/lib/openwin-menu-e
"Tools "          MENU  $OPENWINHOME/lib/openwin-menu-t
"Games "          MENU  $OPENWINHOME/lib/openwin-menu-g
"Utilities "      MENU  $OPENWINHOME/lib/openwin-menu-u
"Properties "     PROPERTIES
SEPARATOR
"X11 Programs "   DIRMENU  /usr/X11R6/bin
"XView Programs " DIRMENU  $OPENWINHOME/bin
"XV"              exec /usr/X11R6/bin/xv
"Window Menu "    WINMENU
SEPARATOR
"Screensaver "    MENU  $OPENWINHOME/lib/openwin-menu-screensave
"Lock Screen "    MENU  $OPENWINHOME/lib/openwin-menu-xlock
"Exit"            EXIT
```

Quelque remarques s'imposent :

– `PROPERTIES` appelle le programme `props` pour effectuer une modification interactive des caractéristiques du bureau (voir plus loin).
– `DIRMENU` fera apparaitre comme sous-menu la liste de tous les fichiers du répertoire spécifié.
– `WINMENU` appelle une sorte d'afficheur de fenêtres et d'icônes actives dans le bureau (dans le workspace).
– `EXIT` permet de sortir de l'interface graphique.

- `exec` exécute le programme comme pour les autres gestionnaires de fenêtres.
- `MENU` appelle un sous-menu définit dans le fichier spécifié.

Voici un exemple de sous-menu, le sous-menu d'outils (*Tools*), définit dans le fichier `$OPENWINHOME/lib/openwin-menu-t` :

```
"Tools" TITLE PIN

"Xfilemanager (File Manager)"    exec /usr/X11R6/bin/xfilemanager
"Xfm 1.2 (File Manager)"         exec /usr/X11R6/bin/xfm
"Xman (View Manual Pages)"       exec /usr/X11R6/bin/xman
"Seyon (Communications Package)" exec /usr/X11R6/bin/seyon
-modem /dev/modem
"Xcalc (Calculator)"             exec /usr/X11R6/bin/xcalc
"Xspread (Spreadsheet)"          exec /usr/X11R6/bin/xspread
"Xxgdb (Debugger)"               exec /usr/X11R6/bin/xxgdb
"Xconsole (Console messages)"    exec /usr/X11R6/bin/xconsole
"Xmag (Magnifying glass)"        exec /usr/X11R6/bin/xmag
"Clocks" MENU                    $OPENWINHOME/lib/openwin-menu-clocks
```

Le menu peut être équipé d'une punaise avec `TITLE PIN`. Ainsi, on peut le fixer au bureau où il peut rester indéfiniment.

Voici un exemple de fichier `$OPENWINHOME/lib/openwin-menu-clocks` dans lequel est définit le sous-menu Clocks (appellé par le sous-menu *Tools*) :

```
"Clocks" TITLE PIN
"Oclock (Contour invisible)" DEFAULT exec /usr/bin/X11/oclock
"Clock" (OpenLook)"                  exec $OPENWINHOME/bin/clock
"Xclock standart"                    exec usr/bin/X11/xclock
```

Pour personnaliser la configuration des sous-menus, on peut copier le fichier `$OPENWINHOME/lib/openwin-menu` dans `$HOME/openwin-menu`[8]. On est alors libre de le modifier, ainsi que les chemins des fichiers contenant les caractéristiques des sous-menus. Seuls les noms des fichiers du menu racine utilisateurs et système (c'est à dire `$OPENWINHOME/lib/openwin-menu` et `$HOME/openwin-menu`) sont imposés.

Enfin, pour configurer le bureau, on peut faire appel à l'utilitaire `props`, qui propose une interface graphique pour configurer les caractéristiques du bureau (*Workspace*), des fenêtres, et configurer aussi les icônes, les menus...Tous les résultats peuvent être sauvegardés en étant écrits dans le

[8] olwm/olvwm appelle en priorité le fichier **$HOME/openwin-menu**.

fichier de ressources `$HOME/Xdefaults` [9] en appuyant sur le bouton “Apply”. Cela permet de voir tout de suite les changements apportés. Il y a en tout 16 options que l'on peut configurer.

Les pages de manuel suivantes donneront plus de détails pour le gestionnaire de fenêtres olwm/olvwm :
- `openwin`.
- `olwm`.
- `olvwm`.
- `olvwmrc`.
- `owplaces`.
- `xview`.
- `props`.
- `setlocale`.

2.16 Configurer mwm

Le gestionnaire de fenêtres *Motif Window Manager* présente beaucoup de similitudes avec **fvwm** (qui d'ailleurs propose une émulation mwm) et **twm**. **mwm** cherche les fichiers de configuration dans l'ordre suivant :

1. le fichier spécifié par la ressource **configFile** (ce peut être `Xconfig` dans le répertoire `/usr/lib/X11` ou `/usr/bin/X11`, ou tout autre fichier).
2. le fichier `$HOME/$LANG/.mwmrc`.
3. le fichier `$HOME/.mwmrc`.
4. le fichier `/usr/lib/X11/$LANG/system.mwmrc`.
5. le fichier `/usr/lib/X11/system.mwmrc`.

Le fichier de configuration `$HOME/.mwmrc` ressemble au fichier de configuration de twm (fichier `$HOME/.twmrc`). Par exemple, pour configurer les menus et sous-menus :

```
#--------------------------------------#
# Programmation des menus et sous menus #
#--------------------------------------#


Menu "DefaultRootMenu"
{
```

[9] Si il existe déjà, il sera écrasé.

```
"Root Menu"     f.title
"Programmes"    f.menu "ProgrammesMenu"
no-label        f.separator
"Pack Icons"    f.pack_icons
"Shuffle Up"    f.circle_up
"Shuffle Down"  f.circle_down
no-label        f.separator
"Restart"       f.restart
"Exit..."       f.quit_mwm
}

menu "ProgrammesMenu"
{
"xearth"            f.exec "xearth -label -grid -markerfile 9x15bold &"
"Mise en veille"    f.menu "MiseEnVeilleMenu"
"Jeux"              f.menu "JeuxMenu"

}

menu "MiseEnVeilleMenu"
{
"bat"       f.exec "xlock -nolock -mod bat &"
"bounce"    f.exec "xlock -nolock -mod bounce &"
"world"     f.exec "xlock -nolock -mod world &"
}

menu "JeuxMenu"
{
"abuse"         f.exec "abuse &"
}
```

Ou encore pour la programmation de l'association de fonctions avec les boutons de la souris et les touches spéciales du clavier :

```
#-------------------------------------------------------------#
# Programmation des associations de fontions avec les boutons #
# de la souris et les touches.                                #
#-------------------------------------------------------------#


Button DefaultButtonBindings
```

```
{
<Btn1Down> icon|frame f.raise
<Btn3Down> icon|frame f.post_wmenu
<Btn3Down> root       f.menu RootMenu
<Btn1Up>   icon       f.normalize
}
```

Une autre manière de modifier l'initialisation du gestionnaire *mwm* consiste à programmer les ressources au niveau local, dans le fichier `.Xdefaults` ou `.Xresources`, suivant le système. Par exemple :

```
!------------------------------------!
! Initialisation du gestionnaire mwm !
!------------------------------------!


! Remplace un clic sur la fenetre par la simple presence du
! pointeur sur cette fenetre.

Mwm*keyboardFocusPolicy : pointer

! Place les icones de haut en bas a droite de l'ecran
! (de gauche a droite en bas de l'ecran par defaut).

Mwm*iconPlacement : top right

! Ouvre une boite a icones dans le coin superieur droit de l'ecran.

Mwm*useIconBox : True
Mwm*iconBoxGeometry : -0+0

! Pour remplacer des icones pour des clients

Mwm*useClientIcon : False

! Remplace l'icone du client xterm (ici /home/xterm.icon).

Mwm*xterm*iconImage : /home/xterm.icon
```

Chapitre 3

Applications graphiques sous XFree86

Les images comme les icônes par exemple, sont sous forme de fichiers **bitmap** (avec l'extension *.xbm*) ou **pixmap** (avec l'extension *.xpm*). Les sections suivantes présentent ces deux types de fichiers images, ainsi que les différentes facons de les construires ou de les modifier.

3.1 Les fichiers bitmap

Les fichiers *bitmap* représentent des images construites point par point, chaque point pouvant avoir une des deux valeurs logique 0 ou 1. En noir et blanc, la couleur blanche sera associée à la valeur 0 et la couleur noire associée à la valeur 1. Ces fichiers sont rangés dans le répertoire suivant :

```
/usr/X11R6/include/X11/bitmaps
```

Un fichier bitmap est un fichier texte écrit en langage C. Ce fichier contient la taille de l'image, ainsi qu'un tableau de valeurs exprimées en hexadécimal spécifiant les bits (convertis en binaires, cela donnera les "0" ou "1" qui seront traduits en blanc ou noir). Par exemple, le fichier suivant :

```
/usr/X11R6/include/X11/bitmaps/xlogo11
```

peut être édité avec l'éditeur bitmap appellé `bitmap` (voir figure 3.1), et visualisé par une commande (`more, less, cat`...) ou un éditeur (`emacs, vi`...). Voici ce que contient ce fichier (réarrangé ligne par ligne de l'image pour qu'il soit plus clair) :

```
#define xlogo11_width 11
#define xlogo11_height 11
```

```
static char xlogo11_bits[] = {
0x0f, 0x04,
0x0f, 0x02,
0x1e, 0x01,
0x3c, 0x01,
0xb8, 0x00,
0x58, 0x00,
0xe8, 0x00,
0xe4, 0x01,
0xc4, 0x03,
0xc2, 0x03,
0x81, 0x07 };
```

Le début du fichier contient le nombre de pixels sur l'horizontale et la verticale (ici respectivement 11 et 11). Ensuite vient le tableau qui contient les valeurs en octal. Pour la première, par exemple, 0f se traduit en binaire par 0000 1111[1] qui, inversé, donne 1111 0000 (le poids le plus faible est à droite, le "f" ici). On peut vérifier grâce à l'editeur bitmat qu'il y a bien pour la première ligne 4 pixels noirs suivits de 4 pixels blancs. La seconde valeur, 04, se traduit en binaire par 0100 0000 qui, inversé, donne 0010 0000. Comme la ligne contient 11 pixels, seuls les trois premiers sont utilisés (001 ici, qui correspond bien à 2 pixels blancs suivis d'un pixel noir) et les cinq autres pixels sont perdus. Les deux nombres suivants sont utilisés pour la seconde ligne, et ainsi de suite...

On peut appeller la commande `bitmap` avec des options, comme la taille de l'image (*-size*) ou la taille d'un carreau élémentaire (*-sh* ou *-sw*) :

```
bitmap -size 17x25 -sh 6 -sw 6 image.xbm
```

L'image aura une taille de 17 carreaux en largeur et 25 carreaux en hauteur, avec chaque carreau ayant pour taille 6 pixels sur 6. Il y aura 3 colonnes dans le fichier `image.xbm` ($3 \times 8 = 24$, valeur supérieure à 17 la plus proche) et 7 bits seront perdus ($24 - 17 = 7$). Le tableau aura au total $3 \times 25 = 75$ valeurs.

L'éditeur bitmap permet de modifier une image bitmap déjà existente, ou bien d'en créer une nouvelle, avec tous les outils nécessaires (copier, coller, zoom...). Il existe deux commandes qui permettent de visualiser en ASCII

[1] 0f en hexadécimal vaut 0 15 en décimal, qui vaut bien 0000 1111 en binaire (voir en annexe A pour le détail de comptage en base 2 et en base 16).

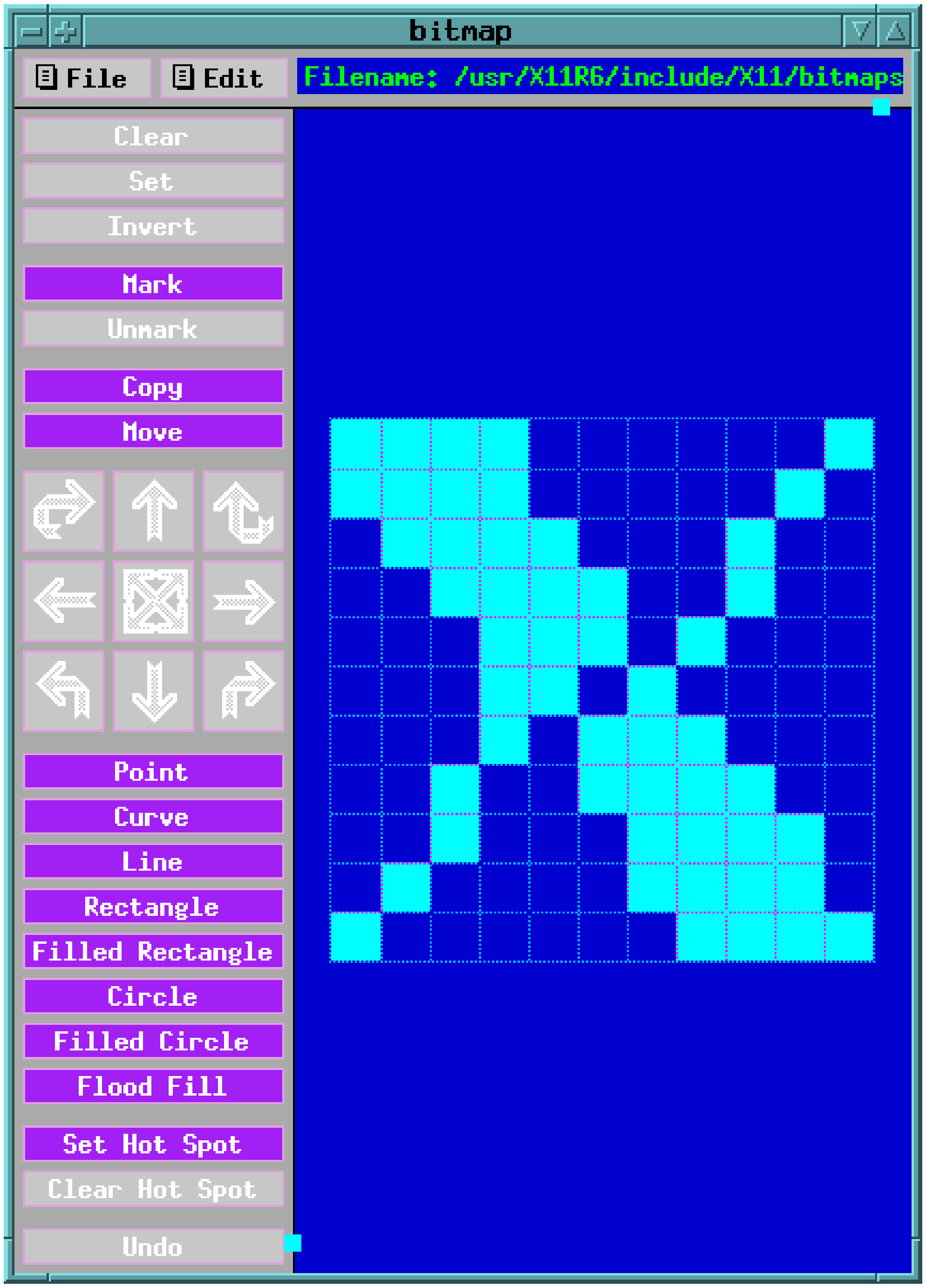

FIG. 3.1 – L'éditeur **bitmap** et le logo xlogo11.

un fichier bitmap (`bmtoa`) et inversement (`atobm`). Ces commandes servent à visualiser rapidement une image, par exemple. Par exemple, pour le fichier `xlogo11`, la commande :

```
bmtoa /usr/X11R6/include/X11/bitmaps/xlogo11
```

donne le résultat illustré par la figure 3.2.

```
####------#
####-----#-
-####---#--
--####--#--
---###-#---
---##-#----
---#-###---
--#--####--
--#---####-
-#----####-
#------####
```

FIG. 3.2 – Résultat de la commande **bmtoa** sur l'image xlogo11.

3.2 Les fichiers pixmap

Les fichiers de type pixmap permettent d'afficher des images en couleurs, mais le principe est le même que pour les fichiers de type bitmap : dans un fichier texte écrit en C sont spécifiés la taille de l'image, le nombre de couleurs, le nombre de caractères par point ainsi qu'une représentation ASCII du dessin, chaque charactère correspondant à une couleur définie juste avant. Par exemple, le fichier suivant :

```
/usr/X11R6/include/X11/pixmaps/xterm.xpm
```

contient les spécifications suivantes :

```
/* XPM */
static char * image_name [] = {
/**/
"64 38 8 1",
/**/
"       s mask  c none",
".      c gray70",
"X      c gray85",
"o      c gray50",
"O      c red",
"+      c darkolivegreen",
"@      c white",
"#      c black",
"                                                                ",
```

```
"                                                                ",
"                          ...........................           ",
"                         .XXXXXXXXXXXXXXXXXXXXXXXXXXX.          ",
"                        .XXXXXXXXXXXXXXXXXXXXXXXXXXXXXoo        ",
" OOOOOOOOOO             .XXOOO++++++++++++++++++++XXXXoo        ",
"  OOOOOOOOOO            .XOOO++++++++++++++++++++++XXXooo       ",
"   OOOOOOOOOO           .OOO+@@+@++@+@@@@++@+++++++XXXooo       ",
"    OOOOOOOOOO          OOO++++++++++++++++++++++++XXXoooo      ",
"     OOOOOOOOOO        OOOX++@@@@+@@+@@@+++++++++++XXXoooo      ",
"      OOOOOOOOOO      OOOXX++++++++++++++++++++++++XXXooooo     ",
"       OOOOOOOOOO    OOO.XX++@@@@@@@@@@+@@@@@++++++XXXooooo     ",
"        OOOOOOOOOO  OOO .XX++++++++++++++++++++++++XXXooooo     ",
"         OOOOOOOOO OOO  .XX++@@@+@@@@+@@++@@@++++++XXXooooo     ",
"          OOOOOOO OOO   .XX++++++++++++++++++++++++XXXooooo     ",
"           OOOOO OOOO   .XX++@+++++++++++++++++++++XXXooooo     ",
"            OOO OOOOOO  .XX++++++++++++++++++++++++XXXooooo     ",
"           OOO OOOOOOOO .XX++@@@@+@+@@@+@++++++++++XXXooooo     ",
"          OOO OOOOOOOOOO.XX++++++++++++++++++++++++XXXooooo     ",
"         OOO   OOOOOOOOOOXX++@+@@@@++++++++++++++++XXXooooo     ",
"        OOO     OOOOOOOOOOXX++++++++++++++++++++++XXXXoooo      ",
"       OOO       OOOOOOOOOOXXXXXXXXXXXXXXXXXXXXXXXXXXXooo       ",
"      OOO         OOOOOOOOOOXXXXXXXXXXXXXXXXXXXXXXXXXooo        ",
"     OOO           OOOOOOOOOOoooooooooooooooooooooooo...oo      ",
"    OOO             OOOOOOOOOO.........................ooo      ",
"   OOO               OOOOOOOOOOooooooooooooooooooooooooooo      ",
"  OOO                 OOOOOOOOOOXXXXXXXXXXXXXXXXXXXXXXoooo.. .. ",
" OOO                      oXXXXXXXXXXXXXXXXXX#######XXoooo  .  .",
"                        oooooooooooooooooooooooooooXXXooo     . ",
"                       oXXXXXXXXXXXXXXXXXXXXXXXXXXooooo      .  ",
"                      oXXXXXXXXXXXXXXXXXXXXXXXXXXoo      oooooo ",
"                      oXX@@@@@@@@@@@@@@@@@@@XXXXXoo   ooooo...o ",
"                     oXXXXXXXXXXXXXXXXXXXXXXXXXXoo   ooXXXoo..o ",
"                    oXX@@@@@@@@@@@@@@@@@@@@XXXXoo    oXXXXX..o  ",
"                   oXXXXXXXXXXXXXXXXXXXXXXXXXXoo     o.....oo   ",
"                   oooooooooooooooooooooooooooo      ooooooo    ",
"                                                                ",
"                                                                "};
```

Les commentaires sont compris entre “/*” et “*/”. Ce fichier a une largeur de 64 points, une hauteur de 38 points, il a 8 couleurs et un caractère par point.

Pour visualiser rapidement une image pixmap, on peut utiliser la commande `sxpm` :

```
sxpm /usr/X11R6/include/X11/pixmaps/xterm.xpm &
```

comme le montre la figure 3.3.

FIG. 3.3 – Le logo xlogo11 affiché par la commande **sxpm**.

Pour éditer ou modifier des fichiers pixmap, on peut utiliser le programme `xpaint`, qui permet de faire à peu près tout ce qui est possible de faire sur un dessin, en couleurs en plus. On peut sauvegarder des zones spécifiques de l'image ou l'image toute entière, dans différents formats, utiliser la loupe pour les modifications, définir la taille d'une nouvelle image...Il existe même deux pallettes différentes, l'une primaire (pour les contours d'une forme géométrique) et l'autre secondaire (pour le remplissage de l'intérieur d'une forme géométrique). On peut également modifier une pallette et sauvegarder ces modifications.

Enfin, il existe plusieurs options de traitements algorithmique de l'image : bosses, relief, inversion des couleurs, aiguisé....

3.3 Les formats de fichiers graphiques

Il existe plus d'une dizaine de formats de fichiers graphiques, chacun ayant ses avantages et ses inconvénients. Par exemple, les fichiers de type bitmap sont de gros fichiers car ils contiennent l'information de chaque point de l'image (et suivant le nombre de couleurs, un point de l'image peut être codé sur un octet pour 256 couleurs ou même 3 octets pour seize millions de couleurs !). La solution consiste à compresser pour diminuer la taille du fichier, mais en cas de zones abimées ou perdues, c'est toute l'image qui est perdue...

Pour identifier le format d'une image, on peut lancer la commande `identify`, appartenant à l'ensemble de programmes *ImageMagick*, (voir section 3.5.1 page 94). Par exemple, sur le fichier `tiger.ps` (normallement contenu dans le répertoire **ghostscript**) :

```
tiger.ps 546x568 PseudoClass 39c 76kb PS 2s
```

ou, avec l'option `-verbose`, qui donne plus de détails :

```
Image: /usr/share/ghostscript/4.03/examples/tiger.ps
  class: PseudoClass
  colors: 39
    0: (  0,  0,  0)  #000000  black
    1: ( 51, 51, 51)  #333333  gray20
    2: ( 76,  0,  0)  #4c0000
    3: (102,102,102)  #666666  gray40
    4: (102,153,  0)  #669900
    5: (153, 38,  0)  #992600
    6: (165, 25, 38)  #a51926
    7: (165, 38, 76)  #a5264c
    8: (178, 51, 89)  #b23359
    9: (178,102,102)  #b26666
    10: (204, 63, 76)  #cc3f4c
    11: (204,114, 38)  #cc7226
    12: (232,127, 58)  #e87f3a
    13: (255,114,127)  #ff727f
    14: (229,102,140)  #e5668c
    15: (153,204, 51)  #99cc33  ~OliveDrab3
    16: (234,140, 77)  #ea8c4d
    17: (234,142, 81)  #ea8e51
    18: (235,149, 92)  #eb955c
    19: (236,153, 97)  #ec9961
    20: (238,165,117)  #eea575
    21: (239,170,124)  #efaa7c
    22: (153,153,153)  #999999  gray60
    23: (178,178,178)  #b2b2b2  ~gray70
    24: (229,153,153)  #e59999
    25: (241,178,136)  #f1b288
    26: (242,184,146)  #f2b892
    27: (243,191,156)  #f3bf9c
    28: (244,198,168)  #f4c6a8
    29: (245,204,176)  #f5ccb0
    30: (229,229,178)  #e5e5b2
    31: (204,204,204)  #cccccc  gray80
    32: (248,216,196)  #f8d8c4
    33: (248,220,200)  #f8dcc8
    34: (249,226,211)  #f9e2d3
    35: (250,229,215)  #fae5d7
```

```
      36: (255,255,204)  #ffffcc
      37: (252,242,235)  #fcf2eb
      38: (255,255,255)  #ffffff  white
    matte: False
    runlength packets: 21041 of 310128
    geometry: 546x568
    depth: 8
    filesize: 76kb
    interlace: None
    format: PS
    comments:
     Image generated by Aladdin Ghostscript (device=pnmraw)
```

on obtient le nom du fichier, les dimensions de l'image en pixels (largeur x hauteur), la classe (*DirectClass* si les codes des couleurs sont directement associés aux couleurs ou *PseudoClass* si des nombres sont associés aux pixels), le nombre de couleurs, la taille du fichier en octets, l'extension en majuscule, et le temps en secondes qu'il a fallut pour obtebnir ces informations. Ici on a un fichier de type PostScript (PS), qui peut directement être inclus par le formateur de texte LaTeX par exemple.

Pour convertir un format de fichier graphique en un autre, on peut utiliser la commande `convert`, appartenant à l'ensemble de programmes *ImageMagick*. La conversion se fait d'après l'extension des noms des fichiers de départ et d'arrivée. Il existe aussi un programme très bien fait appellé `xv`, qui peut afficher les images, les convertires et même appliquer une image en fond d'écran.

3.4 La capture de fichiers image à l'écran

Il existe deux commandes permettant de capturer des images de l'écran :

– la commande `xwd`, qui est utilisée avec l'option *-out* :

```
xwd -out image1.xwd
```

il faut alors cliquer sur la fenêtre désirée pour avoir une image au format *.xwd*. Si on clique sur le fond de l'écran, on obtient une image de la totalité de l'écran. Pour obtenir également le contour de la fenêtre, il faut utiliser l'option *-frame* :

```
xwd -frame > image1.xwd
```

On peut également spécifier le nom de la fenêtre (tel qu'il est affiché sur la barre de titre) avec l'option *-name* en fin de commande, ou son code d'identification (tel qu'il est obtenu avec la commande `xwininfo`, par exemple) :

```
xwd -frame > image1.xwd -name xterm
```

Pour retarder la capture de l'écran (pour un menu qui n'est pas fixe ou si la fenêtre n'apparait pas complétement sur l'écran, par exemple), il faut utiliser la commande `sleep` juste avant :

```
sleep 5; xwd -frame  > image1.xwd -id 0xc00023
```

dans cet exemple, il y aura un délai de 5 secondes avant la capture de la fenêtre.
Pour voir le résultat, utiliser la commande `xwud` avec l'option *-in* :

```
xwud -in image1.xwd &
```

– la commande `import`, appartenant à l'ensemble de programmes *ImageMagick*. L'image de la capture peut être dans différents formats, déterminés par l'extension du nom du fichier (la plupart des principaux formats sont possibles, ce qui évite une conversion de format de fichier). Par exemple :

```
import image1.gif
```

On peut cliquer sur une fenêtre ou bien sélectionner la zone désirée : en cliquant sur un endroit, faire glisser la souris et lâcher le bouton ("appuyer-glisser-lâcher"). Comme pour `xwd`, on peut spécifier un nom de fenêtre ou un code d'identification avec la même option *-window* [2] :

```
import -window 0xc00023 image1.gif
```

Enfin, l'option *-delay* permet d'attendre un certain nombre de secondes avant la capture (5 secondes dans l'exemple qui suit) :

```
import -delay 5 -window 0xc00023 image1.gif
```

Pour les gestionnaires de fenêtres comme **mwm** ou **olwm**, il existe des commandes comme `snapshot` présentant un menu interactif.

3.5 Les autres programmes : ImageMagick, xv et xfig

3.5.1 ImageMagick

ImageMagick est un ensemble de programmes graphiques d'affichage et de manipulation interactive des fichiers image. Cet ensemble comprend les

[2] Pour la totalité de l'écran, indiquer le nom *root*.

programmes suivants :

import capture d'une partie ou de la totalité de l'image de l'écran

identify identification et description du format d'un ou de plusieurs fichiers image

convert conversion d'un fichier image d'un format dans un autre

display affichage, manipulation et traitement des fichiers image

animate affichage d'une séquence d'image

montage création d'un ensemble d'images, composé à partir de plusieurs autres images

mogrify traitement d'une image ou d'une séquence d'images

combine création d'une image par combinaison de plusieurs autres images

segment segmentation d'une image

xtp récupération, affichage ou impression de fichiers image en provenance d'un site éloigné, par l'intermédiaire d'un réseau, ou envoi dans le réseau de tels fichiers

La principale commande est donc `display`, qui permet l'affichage du contenu d'un fichier image. Par exemple :

```
display image1.gif &
```

En cliquant avec le bouton gauche de la souris sur l'image, on obtient le menu principal. Les options suivantes permettent de spécifier la position et la taille de la fenêtre :

```
display -geometry 420x315+720+133 image1.gif &
```

et les options suivantes permettent de spécifier la taille de la fenêtre en réduction et en agrandissement[3] :

```
display -geometry 80%x80% image1.gif &
```

Les commandes qui suivent permettent respectivement d'extraire une partie de l'image (avec l'option *-crop*) et d'ajouter une bordure autour de l'image (avec l'option *-border*, ici de 15 pixels sur les cotés verticaux et 25 pixels sur les cotés horizontaux), de couleur rouge (spécifiée par l'option *-bordercolor*) :

```
display -crop 150x120+12+30 image1.gif &
display -border 15x25 -bordercolor red image1.gif &
```

[3] On aurait pu spécifier *-geometry 80%*, qui est équivalent à l'option *-geometry 80%x80%*, puisque les deux spécifications sont identiques.

Les commandes qui suivent permettent respectivement d'afficher l'image avec une rotation d'un angle de 45° (avec l'option *-rotate*), de faire un "diaporama" avec un arrêt de 1 seconde sur chaque image, de deux facons différentes en utilisant les commandes `display` (avec l'option *-delay*) et `animate` (avec l'option *-delay* aussi) :

```
display -rotate 45 image1.gif &
display -delay 1 *.xwd &
animate -delay 1000 *.xwd &
```

Pour cette dernière commande, il vaut mieux lancer avant la commande `mogrify`, qui prépare les images[4] (l'option *-colors* force le nombre maximum de couleurs, et le point d'exclamation "!" force à adopter le format spécifié) :

```
mogrify -geometry 400x400! -colors 8 *.xwd &
```

Pour obtenir un répertoire visuel, on peut passer l'une des deux commandes suivantes :

```
display 'vid:*.xwd' &
```

qui peut être affichée avec l'option *Visual_Directory* du menu *File*, ou comme autre commande :

```
montage -title 4x3 *.xwd out.xwd &
```

qui affiche les images sur 4 colonnes et 3 lignes. Il faut spécifier un nom de fichier inexistant à la fin de la commande, qui sera crée (si il existe déjà, il sera écrasé...).

Le menu *Image_Edit/Draw* permet de dessiner sur l'image avec des outils sommaires. Enfin, la commande `combine` permet de construire une image en combinant deux fichiers existants :

```
combine -ge +20+30 image1.gif image2.gif imageout.gif
```

L'image `imageout.gif` sera la supperposition des images `image1.gif` et `image2.gif`.

3.5.2 xv

Le programme `xv` permet d'afficher des images, de convertir des formats...Pour obtenir le menu cliquer sur le bouton droit de la souris.

Une fois une image affichée, on peut connaitre les coordonnées du point et la couleur (affichée en hexadécimal) ou le niveau de gris (pour les images en

[4] **Attention !** Les fichiers sont écrasés ! Toute modification est donc irréversible. Faire des sauvegardes des fichiers avant de les passer à cette commande...

noir et blanc). Pour afficher une image en fond d'écran, sélectionner l'une des options *root* du menu *display*. Ce logiciel est un *shareware*, il faut l'enregistrer pour une utilisation courante (à titre commercial).

3.5.3 xfig

Le programme `xfig` permet de charger des fichiers au format spécial *xfig* (*.fig), au format LaTeX, GIF, EPS...Ce programme permet de réaliser toute sorte de figures, diagrammes, shémas, dessins, et de les convertires dant un format utilisable par LaTeX. De nombreux exemples (*.fig) montrent qu'avec un peu d'experience, on peut réaliser des dessins impressionnants (figure 3.4).

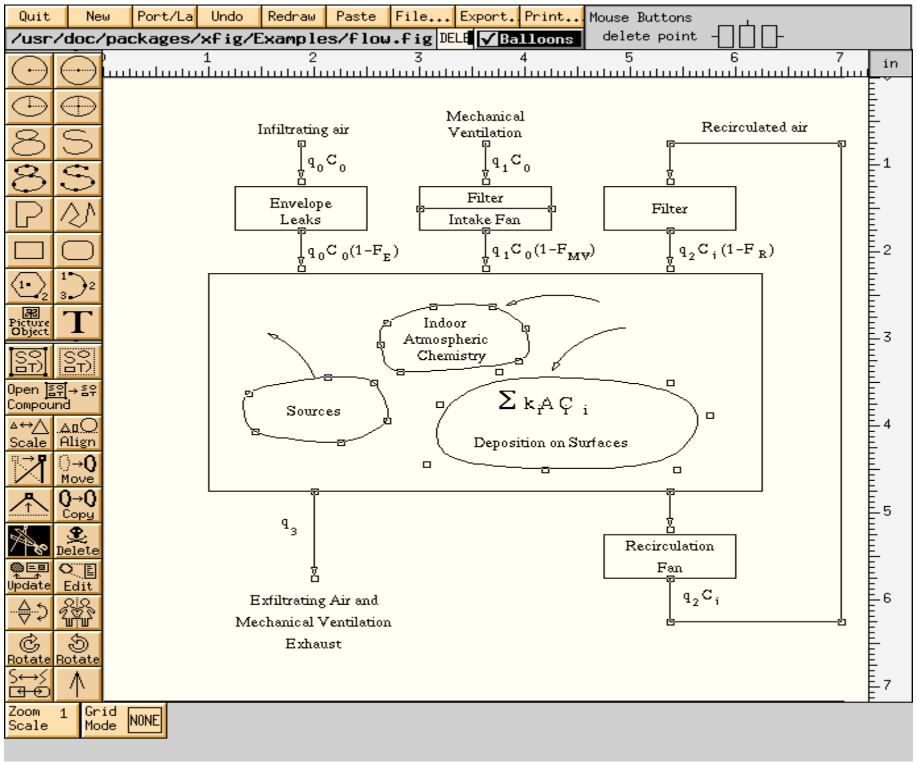

FIG. 3.4 – Un exemple d'utilisation de **xfig**.

3.6 Autres choses utiles quand on fait du X

– **Formatage de textes :**

Pour `gs` et `xdvi`, il faut utiliser des options spéciales (on peut les définires comme des alias). En effet, ils sont souvent configurés pour un papier de taille anglo-saxonne. Ces options sont les suivantes :

```
gs -sPAPERSIZE=a4
xdvi -paper a4
ghostview -a4
```

Et pour que `dvips` convertisse les documents dans un format papier A4, il faut spécifier dans le fichier `config.ps` :

```
@ a4 210mm 297mm
@+ ! %%DocumentPaperSizes: a4
@+ %%PaperSize: a4
@+ %%BeginPaperSize: a4
@+ a4
```

– Le programme `xearth` affiche une zolie Terre en fond d'écran. Voici comment la lancer encore plus belle :

```
xearth -label -grid -wait 60 +pos ``sunrel 20 -30'' &
```

– Le programme `xload` affiche la charge du système. Voici quelque options qui devraient ravire grands et petits :

```
xload -update 5 -scale 2 -jumpscroll 8 &
```

où *-update* indique l'intervalle en secondes entre chaque remise à jour (10 par défaut), *-scale* indique l'échelle initiale de charge moyenne (1 unité par défaut), et *-jumpscroll* indique de combien de pixels il doit être déplacé à chaque fois qu'il touche la limite droite de la fenêtre (la moitié de la largeur de la fenêtre par défaut).

Chapitre 4

Outils

4.1 Emacs

Dans la documentation d'Emacs, la touche `Ctrl` est remplacée par la touche `C`, et la touche `Alt` par la touche `M`. De plus appuyer en même temps sur la touche `Ctrl` et `c` est symbolisé par `C-c`. C'est cette convention qui sera utilisée dans ce document.

Pour ouvrir un fichier, taper `C-x C-f` et saisir le nom du fichier (la complétion fonctionne, c'est à dire que si on appuie sur la touche `Tab`, le nom du fichier sera complété comme dans le shell). Si plusieurs fichiers sont ouverts en même temps, pour passer de l'un à l'autre taper `C-x b` et saisir le nom du fichier. `C-x C-b` permet d'afficher la liste de tous les fichiers ouverts. `C-x o` permet de passer d'une fenêtre à l'autre, et `C-x 1` permet de n'afficher plus qu'une fenêtre, celle dans laquelle se trouve le curseur. `C-x k` permet de supprimer un tampon de la mémoire.

Le tutorial permet d'obtenir de précieuses informations (`C-h-t`) et dans chaque mode, `C-h m` affiche un résumé des commandes disponibles. Les séquences de touches `C-h k`, `C-h f` et `C-h v` permettent respectivement d'obtenir de l'aide sur une *séquence de touches* spécifiée, une *fonction* spécifiée et une *variable* spécifiée.

4.1.1 Copier et coller

Pour copier/couper et coller, il faut :

1. se mettre au début du morceau puis taper `C-SPACE`.
2. aller à la fin de la région (partie du texte sélectionnée).
3. appuyer sur `M-w` pour **copier** ou sur `C-w` pour **effacer** la région.

4. se positionner à l'endroit d'insertion et taper `C-y`.

La région précédement sélectionnée peut être remise à l'aide de `M-y`. On peut sauvegarder un tampon sous un nom particulier, un **registre** (*a* dans l'exemple qui suit) :

1. se mettre au début du morceau puis taper `C-SPACE`.
2. aller à la fin de la région (partie du texte sélectionnée).
3. appuyer sur `C-x x` puis saisir le nom du registre (ici `a`).
4. se positionner à l'endroit d'insertion et taper `C-x g` puis le nom du registre (en l'occurrence `a`).

4.1.2 Recherche et remplacement

`C-s` permet de rechercher une chaîne de caractères, et si on veut effectuer une nouvelle recherche, il suffit de taper à nouveau sur `C-s`. `C-r` permet d'inverser le sens de recherche et `M-C-s` met à contribution les expressions rationnelles (sans distinction de majuscule ou minuscule).

`M-%` lance la substitution interactive. Taper sur la barre d'espacement pour accepter le remplacement de la chaîne, sur la touche de suppression pour éluder cette occurrence, sur $\boxed{!}$ pour tout remplacer et sur $\boxed{.}$ pour arrêter la procédure. Pour tout remplacer sans aucune demande de confirmation, utiliser la fonction `replace-string` : `M-x replace-string`. La fonction `regexp` sert à remplacer une expression rationnelle.

4.1.3 Macro-instructions

On peut définir des **macro-instructions**, comme pour mettre une majuscule en début de ligne (avec `M-c`). Pour cela, il faut :

- presser `C-x (`
- se placer en début de ligne.
- taper `M-c`
- passer à la ligne suivante.
- presser `C-x )`

Pour appeller cette macro, taper `C-x e`.

4.1.4 Indentation

Pour indenter un fichier source par exemple, il faut :

1. selectionne la région par `C-SPACE`
2. taper `M-w` puis `M-C-\`

Ca indente le listing. Il y a plusieurs style d'indentation au choix.

4.1.5 Lancer des commandes

On peut envoyer un mail, en tapant sur `C-x m` puis sur `C-c C-s` pour envoyer le message écrit. RMAIL est une interface permettant de lire un courrier électronique : `M-x rmail`. Les touches `n` et `p` permettent de lire le message suivant et précédent, et `r` permet d'y répondre (`C-c C-y` permet d'y insérer le message original). La touche `h` affiche un résumé des messages.

On peut aussi lire des news sur l'un des forums Usenet (`M-x gnus`) ou compiler un programme C (`M-x compile`, pour aller à l'erreur indiquée taper `C-c C-c`) et même utiliser le débogueur `gdb` (voir section 8.7.2 page 209). Une commande peut être lancée avec `M-!`, et une fenêtre shell peut être ouverte avec `M-x shell`. `M-|` permet de transmettre le contenu de la région courante par un tube à l'entrée standard d'un programme.

On peut également consulter des pages info avec `C-h-i`. Les touches utiles pour naviguer sont :

- `m` suivie du nom du noeud pour lire la partie correspondante.
- `n` pour passer au noeud suivant.
- `p` pour rejoindre le noeud précédent.
- `l` pour rejoindre le dernier noeud consulté.
- `?` pour afficher une liste de commandes.
- `h` pour avoir une courte explication sur le fonctionnement de ce système.

4.1.6 Configurer Emacs

Configurer Emacs consiste à écrire un fichier `.emacs` se composant de code LISP. Par exemple, pour créer un racourcis clavier, comme lancer RMAIL lorsqu'on tape `C-c r` :

```
; Lecture du courrier.

(global-set-key "\C-cr" 'rmail)
```

Cette définition est faite pour tous les modes. On peut définir une nouvelle fonction et l'affecter à un racourcis clavier (en mode C dans l'exemple qui suit) :

```
; Insere dans un fichier le bloc suivant ("\n" passe a la ligne) :
;
```

```
; if () {
; }
;
; et met le curseur entre les parantheses ("backward-char 6", revient
; 6 caracteres en arriere).

(defun start-if-block()
(interactive)
(insert "if () {\n}\n")
(backward-char 6)
)

; Affectation d'une sequence de touches a cette fonction.

(define-key c-mode-map "\C-ci" 'start-if-block)
```

On peut aussi reconfigurer le clavier :

```
; Reconfiguration du clavier.


; Touche Backspace = retour arriere.
;
; "?" signifie "le code ASCII de"
; "?\C-h" signifie donc "le code ASCII de Ctrl enfonce avec h".
; Ce code (8) est le meme que "retour arriere".
; "\C-?" represente la touche supprime.

(keyboard-translate ?\C-h ?\C-?)

; Ctrl-\ ("?\C-\\") = Ctrl-h.

(keyboard-translate ?\C-\\ ?\C-h)

; Encore des modifications de touches...

(global-set-key "\C-a" 'scroll-up)
(global-set-key "\C-q" 'scroll-down)

; "global-unset-key" libere une sequence de touche.
; "overwrite-mode" provoque la bascule du mode insertion/remplacement.
```

```
(global-unset-key "\C-r")
(global-set-key "\C-r" 'overwrite-mode)

; La touche Entree indente la ligne suivante.

(define-key indented-text-mode-map "\C-m" 'newline-and-indent)
```

Enfin, voici quelque modifications succinctes qui permettent d'améliorer l'apect visuel du texte :

```
; Mode par defaut pour edition de texte qui indente les lignes.

(setq default-major-mode 'indented-text-mode)



; Validation de la saisie au kilometre par "text-mode-hook".
; Longueur maximum de chaque ligne a 72 caracteres.

(setq text-mode-hook 'turn-on-auto-fill)
(setq fill-column 72)
```

Ce ne sont ici que quelque exemples de base qui peuvent largement être améliorés.

```
;;
;; Fichier .emacs: initialisation d'emacs
;;   Fichier de base : Guide du Rootard
;;

(display-time)  ;; Pour avoir l'heure dans la barre d'etat
(setq display-time-24hr-format t)  ;; Format 24 heures

;; Nouveaux modes
(autoload 'c++-mode   "cplus-md" "C++ Editing Mode" t)
(autoload 'perl-mode   "perl-mode" "Perl Editing Mode" t)
(autoload 'c-mode "c-mode" "C Editing Mode" t)
; mieux vaudrait utiliser le "cc-mode"

;; Auto-Mode Settings : positionne le mode selon l'extension
(setq auto-mode-alist
(append '(("\.c$"  . c-mode)    ;; utilise le mode C++ meme pour C
("\.h$"  . c-mode)
("\.C$"  . c++-mode)
```

```
("\.H$"  . c++-mode)
("\.cc$" . c++-mode)
("\.C$"  . c++-mode)
("\.pl$" . perl-mode)           ;; Perl
("/tmp/snd\.[0-9]*" . text-mode);; Text (pour le mail)
("[Rr][Ee][0-9]*" . text-mode)
("\.ada$"  . ada-mode)          ;; Ada
("\.spec$"  . ada-mode)
("\.body$"  . ada-mode))
auto-mode-alist))

# Remapes varies
(global-set-key "\eg" 'goto-line)       ;; ESC G = Goto line
(global-set-key "\eo" 'overwrite-mode)

(put 'eval-expression 'disabled nil)

;; Accents...
(standard-display-european 1)
(load-library "iso-syntax")

;; Sous X-Window, textes en couleurs (C/C++/Shell/Makefile,etc)
(cond (window-system
(setq hilit-mode-enable-list  '(not text-mode)
hilit-background-mode   'light
hilit-inhibit-hooks     nil
hilit-inhibit-rebinding nil)
(require 'hilit19)
))
(if (not (equal window-system ""))
(global-set-key "\C-?" 'delete-char))
))
;; mieux vaudrait employer font-lock
```

4.1.7 Emacs et X window

Bien sûr, on peut effectuer les opérations classiques de copier/coller entre deux fenêtres Emacs ou entre une fenêtre Emacs et une fenêtre xterm, par exemple. Pour lancer Emacs, on peut définir des couleurs de fond (avec l'option *-bg*), de texte (avec l'option *-fg*), de curseur (avec l'option *-cr*), et de pointeur de la souris (avec l'option *-ms*). Par exemple :

```
emacs -bg ivory -fg slateblue -ms orange -cr brown memo.tex &
```

Pour lancer `xemacs`, la version “souris” de Emacs, il faut utiliser l'option *-mc* pour la couleur du curseur :

```
xemacs -bg ivory -fg slateblue -mc orange -cr brown memo.tex &
```

La souris fonctionne d'une facon particulière avec le bouton droit : on sélectionne d'abord le début de la zone avec un clic sur le bouton gauche, ensuite on peut sélectionner une zone de texte par un simple clic sur le bouton droit de la souris à tout autre endroit du texte, *puis* couper avec un second clic sur le bouton droit. On peut ensuite, comme dans une fenêtre xterm, coller par un clic sur le bouton du milieu de la souris.

La configuration des couleurs, de la présence ou non du menu, de l'ascenseur. . .peut se faire dans le fichier `.emacs`. Chaque fois que Emacs sera lancé, ces paramètres (un code Lisp, en fait) seront pris en compte. Par exemple, pour ne pas avoir de barre de menus ni d'ascenseur, insérer dans le fichier `.emacs` le code suivant :

```
(if (getenv "DISPLAY")
    (progn (menu-bar-mode -1)
     (scroll-bar-mode -1))
)
```

La correspondance entre la souris et une action peut également être définie dans le fichier `.emacs`. Par exemple, pour éditer un message électronique en appuyant en même temps sur le bouton gauche de la souris et la touche `Shift` , entrer le code suivant :

```
(define-key global-map [S-mouse-1] 'mail)
```

La touche S est équivalente à la touche `Shift` , les touches `Alt` et `Ctrl` étant respectivement équivalentes à “M” et “C”[1]. Une combinaison de touches telle que appuyer en même temps sur la touche `Ctrl` et la touche “x”, par exemple, est symbolisée en général (dans la documentation de Emacs ou d'autres programmes) par `C-x`. Bien entendu, `C-S-x` symbolise l'appui en même temps sur les touches `Ctrl` , `Shift` et `x` . Il en est de même pour le fichier `.emacs`. Ainsi, on peut changer le code précédent par :

```
(define-key global-map [S-C-mouse-1] 'mail)
```

[1]La touche “Echap” peut parfois être équivalente à la touche “M”. Lorsqu'une touche doit être appuyée en même temps avec **Alt**, avec **Echap** on devra appuyer sur **Echap** *puis* sur la touche voulue.

On peut travailler dans plusieurs fenêtres en même temps. Toute modification dans l'une sera apportée aux autres. Pour ouvrir une autre fenêtre, presser `C-x 5 2` et pour n'en fermer qu'une, presser `C-x 5 0` . Si on ferme une fenêtre par la commande "classique" `C-x C-c`, les deux fenêtres seront fermées en même temps.

Les couleurs peuvent être modifiées au cours d'une session Emacs, ce qui facilite les essais. La liste des couleurs disponibles peut être obtenue avec la commande `showrgb` :

```
showrgb | less
```

De même, on peut facilement trouver toute sorte de couleurs différentes portant un nom similaire (et donc une teinte semblable). Par exemple, pour trouver tous les oranges disponibles, taper :

```
showrgb | grep orange
```

Le résultat devrait ressembler à ca :

```
255 165   0            orange
255 140   0            dark orange
255  69   0            orange red
255 165   0            orange1
238 154   0            orange2
205 133   0            orange3
139  90   0            orange4
```

On peut définir de nouvelles couleurs, voir section ?? page ??. Par exemple, pour accéder aux changements de couleurs, il faut presser `M-x` puis taper `set-background-color` pour changer la couleur de fond et valider par `Enter`. Attribuer alors une nouvelle couleur. On peut tout aussi bien entrer `set-foreground-color`, `set-cursor-color` ou `set-mouse-color` à la place de `set-background-color`. Les couleurs peuvent différencier les mots-cléfs en programmation C par exemple. Ainsi si on met dans le fichier `.emacs` le code suivant :

```
(add-hook 'c-mode-hook '(lambda () (font-lock-mode 1)))
(add-hook 'c++-mode-hook '(lambda () (font-lock-mode 1)))
(add-hook 'lisp-mode-hook '(lambda () (font-lock-mode 1)))
(add-hook 'emacs-lisp-mode-hook '(lambda () (font-lock-mode 1)))
```

en C les commentaires, les fonctions. . .n'apparaitront pas de la même couleur. Chaque fichier édité par Emacs se terminant par le suffixe *.c* aura droit à ce traitement.

Pour changer les modes d'affichages des caractères, presser $\boxed{\texttt{M-x}}$ puis taper `list-faces-display` pour faire apparaitre la liste des différentes teintes utilisées. Pour en changer, procéder comme pour l'exemple qui suit :

1. Entrer $\boxed{\texttt{M-x}}$ `set-face-background`
2. taper `modeline`
3. taper `lemonchiffon`

Le résultat devrait apparaitre immédiatement. Les autres modes d'affichages des caractères peuvent être changés de la même facon.

Pour que ces modifications perdurent, on peut les indiquer dans le fichier `.Xdefaults` en utilisant les ressources (voir section 2.13 page 60), mais c'est plus simple de l'éditer dans le fichier `.emacs`. Par exemple :

```
(set-background-color "ivory")
(set-foreground-color "slateblue")
(set-cursor-color "brown")
(set-mouse-color "orangered")

(set-face-foreground 'bold "black")
(set-face-background 'bold "lightpink")
(set-face-foreground 'bold-italic "red")
(set-face-background 'bold-italic "wheat")
(set-face-foreground 'italic "darkolivegreen")
(set-face-background 'modeline "lemonchiffon")
(set-face-foreground 'modeline "maroon")
(set-face-foreground 'underline "violet")
```

Le tableau 4.1 résume les principales options de Emacs.

Commande	**Signification**
C-a	Déplacement au début de la ligne.
C-e	Déplacement à la fin de la ligne.
C-x C-f	Ouvre un fichier.
C-x C-v	find-alternate-file.
C-x i	insert-file (après le point courant).
C-x C-s	Sauvegarde le tampon (fichier).
C-x C-w	Sauvegarde le tampon dans un fichier ("save as").
C-x C-c	Quitter Emacs.
C-h	Aide.
C-h i	Info.
C-f	forward-char.
C-b	backward-char.
C-d	Efface le caratère pointé.
C-k	Efface jusqu'à la fin de la ligne.
M-d	Efface jusqu'au début du mot.
M-Backspace	Efface jusqu'à la fin du mot.
C-w	Efface et stocke le texte sélectionné.
C-y	Récupère le dernier effacement du texte.
M-f	forward-word.
M-b	backward-word.
C-p	previous-line.
C-n	next-line.
M-e	end-of-sentence (une phrase peut s'étendre sur plusieurs lignes).
M-a	begining-of-sentence.
M-}	forward-paragraph.
M-{	backward-paragraph.
M-<	Déplacement au début du tampon.
M->	Déplacement à la fin du tampon.
C-v	Recule d'un écran.
M-v	Avance d'un écran.
C-I	Recenter (rafraîchissement et recentrage de l'ecran).
M-n	digit-argument (répétition n fois : **M-3 C-p** pour remonter de 3 lignes).

Commande	Signification
C-u n	universal-argument (4 par défaut, essayer *C-u 75 #*).
C-_	Annule la dernière modification.
C-s	Recherche une chaîne de caractère vers la fin du tampon. Recherche incrémentale, puis **C-s** nouvelle occurence.
C-r	Recherche une chaîne de caractère vers le début du tampon.
M-%	Recherche et remplacement de texte.
C-x u	undo.
M-x	revert-buffer : annule tout depuis la dernière sauvegarde.
C-t	Inverser (comme “toggle”).
M-t	Permuter deux mots (positionner le curseur entre les deux mots).
C-x C-t	Permute les lignes.
M-u	upcase-word.
M-c	capitalize-word.
M-l	lowercase-word.
RETURN	Sortie de la recherche.
C-s C-w	Recherche avec le mot sous le curseur.
C-s C-y	Recherche avec la fin de la ligne.
C-x 3	Fractionne la fenêtre verticalement.
C-x 2	Fractionne la fenêtre horizontalement.
C-x 1	Ferme toutes les fenêtres Emacs sauf la première.

TAB. 4.1 – Principaux raccourcis sous **Emacs**.

4.2 Utilisation de TeX et LaTeX

TEX est un *formateur de texte*, qui permet d’écrire un livre, un article, un formulaire mathématique sans *trop* se soucier de la mise en page, ni de l’index ou de la table des matières. Il suffit d’indiquer dans le document les rudiments, et en compilant, tout est bien mis dans le document final. Mais TEX peut être utilisé avec les macros qui font un programme encore plus facile à utiliser : LATEX. Les fichiers TEX se terminent en général par *.tex*. Pour formater un fichier, il faut entrer l’une des deux commandes `tex` ou `latex` suivies du nom du fichier, sans besoin de préciser l’extension si celle-ci est *.tex*. Il peut être parfois nécessaire de compiler deux ou trois fois de suite le document pour la bibliographie, par exemple.

Un index peut être contenu dans le fichier, auquel cas il faudra appeler la commande `makeindex` et relancer `latex` :

```
latex memo.tex
makeindex memo.idx
latex memo.tex
```

Le résultat de la commande produit un fichier dont l'extension est *.dvi*, comme `memo.dvi`. Ce fichier peut être lu à l'aide de la commande `xdvi` ou convertit en fichier PostScript à l'aide de la commande `dvips`.

Des programmes tels que `tktex` ou `lyx` permettent une édition aisée des fichiers sources pour LaTeX.

Pour plus de renseignement, consulter le document [1].

4.3 Réaliser des pages de manuel à l'aide de groff

La rédaction d'une page de manuel consiste en l'écriture d'un fichier qui, formaté *via* la commande `groff`, affiche la page de manuel[2]. Pour réaliser une page de manuel d'une commande imaginaire `coffee`, par exemple, il faut d'abord éditer le fichier `coffee.man`. Ce fichier peut ressembler à ceci :

```
.TH COFFEE 1 "29 Juillet 94"

.SH NOM
coffee \- Controle la machine a cafe

.SH SYNOPSIS
\fBcoffee\fP [ -c | -b ] [ -t \fItype\fP ] \fInombre\fP

.SH DESCRIPTION
\fBcoffee\fP demande a la machine connectee sur \fB/dev/cf0\fR de faire du
cafe. Le parametre \fInombre\fP specifie le nombre de tasses.

.SS Options

.TP
\fB-c\fP
```

[2]La page de manuel est re-formatée par la commande **man** à son appel.

```
caffe chaud

.TP
\fB-t \fItype\fR
specifie le type de caffe ou \fItype\fP peut etre \fBColombie\fP

.SH FICHIERS
.TP
\fC/dev/cf0\fR
Le periferique de controle de la machine (j'ai fait espray pour la fote)

.SH "VOIR AUSSI"
lait(5), sucre(5), cognac(5)

.SH BOGUES
Connait pas les beugues moi, Mosieur !
```

La séquence `.TH` affecte le titre de la page de manuel, et `.SH` un début de section. La syntaxe :

```
coffee \- Controle la machine a cafe
```

doit être respectée pour que la page de manuel puisse intégrer la base de données accessible par la commande `man -k` ou `apropos`[3].

Les séquences `\fB`, `\fI` et `\fR` permettent respectivement d'afficher le texte en **gras**, en *italique* et de revenir à la police précédente. La séquence `.SS` permet quand à elle de débuter une sous-section et la séquence `.TP` permet d'afficher les options en retrait.

Il faut maintenant formatter cette page de manuel, à l'aide de `groff` :

```
groff -Tascii -man coffee.man | less
```

L'option -*Tascii* précise que le format devra être de type ASCII (l'option -*Tps* permettrait de formater une page de manuel au format PostScript), et la page de manuel est affichée à l'écran. Le résultat ressemble à ceci :

```
COFFEE(1)                                                 COFFEE(1)


NOM
```

[3]Ne pas oublier de lancer la commande **makewhatis** pour mettre à jour cette base de données.

```
        coffee - Controle la machine a cafe


SYNOPSIS
        coffee [ -c | -b ] [ -t type ] nombre


DESCRIPTION
        coffee  demande  a  la  machine  connectee sur /dev/cf0 de
        faire du cafe. Le parametre nombre specifie le  nombre  de
        tasses.


   Options
        -c     caffe chaud


        -t type
               specifie  le type de caffe ou type peut etre Colom
               bie


FICHIERS
        /dev/cf0
               Le periferique de controle de la machine (j'ai fait
               espray pour la fote)


VOIR AUSSI
        lait(5), sucre(5), cognac(5)


BOGUES
        Connait pas les beugues moi, Mosieur !
```

Il ne reste plus qu'à copier ce fichier dans le répertoire contenant les pages de manuel (en changant l'extension *.man* en *.1*) :

```
cp coffee.man /usr/man/man1/coffee.1
```

Pour installer la page de manuel dans un autre répertoire (par exemple `$HOME/man`), il faut inclure ce répertoire dans la variable d'environnement

$MANPATH :

```
export MANPATH=$MANPATH:$HOME/man
```

Cette commande devra être incluse dans un fichier de démarrage (comme le fichier `.bashrc`, par exemple) pour qu'il soit tenu compte de cette modification à chaque session.

Il ne reste plus qu'à lancer la commande `man` :

```
man coffee
```

La page de manuel devrait alors s'afficher.

4.4 Réaliser des pages d'info à l'aide de Texinfo

La réalisition d'une page Info consiste en l'écriture d'un fichier source, qui sera formaté à l'aide de la commande `makeinfo`. Ce qui suit montre un exemple de fichier source, le fichier `vide.texi`, qui documente une commande imaginaire `vide`. Voici l'en-tête du fichier Texinfo :

```
\input texinfo @c -*-texinfo-*-
@c %**start of header
@setfilename vide.info
@settitle Le plein de vide
@setchapternewpage odd
@c %** end of header
```

La commande `\input texinfo` servira à la version formatée par TEX en vue d'une éventuelle impression papier. Les commentaires sont précédés de la commande `@c`. Les commandes `@setfilename` et `@settitle` indiquent respectivement le nom du fichier Info à réaliser, le titre et `@setchapternewpage` précise que chaque nouveau chapitre doit commencer sur une page impaire (`odd`).

La suite initialise le titre de la page, utilisé lors du formatage par TEX :

```
@titlepage
@title Vide
@subtitle Le plein de vide
@author par Mathieu DECORE
@end titlepage
```

Voici maintenant les *noeuds* du document (les sections, en quelque sorte, accessibles en tapant les touches $\boxed{\texttt{m}}$, $\boxed{\texttt{n}}$, $\boxed{\texttt{p}}$ et $\boxed{\texttt{l}}$). Le premier noeud s'appelle *Top* et est définit par la commande `@node` :

```
@c    Node, Next        , Previous, Up
@node Top , Description ,          , (dir)

@ifinfo
Ce fichier ne documente rien (vide).
@end ifinfo

@menu
* Description :: Description du vide
* Appel :: Comment utiliser le vide
* Index :: Index de ce document
@end menu
```

A la suite de la commande `@node` est indiqué le noeud suivant, le précédent et le parent ("`(dir)`" désigne la page d'info générale de tout le système). Un résumé de ce fichier est compris entre `@ifinfo` et `@end ifinfo`, et n'apparaîtra que dans la page Info (et pas dans le document formaté par TEX).

Le noeud *Description* est la première page (ou le premier chapitre) :

```
@c    Node       , Next , Previous, Up
@node Description, Appel, Top     , Top
@chapter Description du @code{vide}

@cindex Le non etat
@cindex Description

@section Premiere section
@subsection Premiere sous section

@section Seconde section

Le @code{vide} n'est rien. Voir aussi @xref{Appel} pour la syntaxe
du @code{vide}.
```

Les commandes `@chapter`, `@section` et `@subsection` ne seront exploitées que par TEX. La commande `@cindex` permet d'insérer une ligne dans l'index en fin de document, `@code` permet d'imprimer en mode `verbatim`, et `@xref` permet de faire référence à un autre noeud ou un autre document Texinfo (qu'on peut consulter en tapant sur $\boxed{\texttt{f}}$).

Le noeud suivant, appellé *Appel*, montre un exemple grâce à la commande `@example` :

```
@node Appel, Index, Description, Top
@chapter Executer @code{vide}

@cindex Execution du @code{vide}
@code{vide} s'execute comme ca :

@example
vide @var{options} @dots{}
@end example
```

Cet exemple apparaîtra de la facon suivante pour un texte formaté par TEX :

`vide` *options* ...

et comme ca pour une page Info :

```
vide OPTIONS ...
```

La suite du noeud *Appel* montre un tableau à deux colonnes (`@table`) dont chaque option sera affichée de manière spéciale (`@samp`) :

```
@cindex Options
@cindex Arguments
Les options suivantes sont supportees :

@cindex Obtenir de l'aide
@table @samp
@item -help
Affiche l'aide.

@item -version
Affiche la version de @code{vide}.

@end table
```

Il reste ensuite à afficher l'index à cet endroit précis du document (`@printindex cp`) :

```
@node Index, , Appel, Top
@unnumbered Index

@printindex cp
```

et le document se termine par un court sommaire (`@shortcontents`), une table des matières (`@contents`) et le travail est ainsi terminé (`@bye`) :

```
@shortcontents
@contents
@bye
```

La commande `makeinfo` réalise la page Info et crée le fichier indiqué (`vide.info`, en l'occurence) :

```
makeinfo vide.texi
```

Avec Emacs, on peut taper `M-x makeinfo-region` et `M-x makeinfo-buffer`.

On peut maintenant lire ce fichier (`C-h i` puis $\boxed{\text{g}}$ suivit du chemin avec Emacs) :

```
info -f vide.info
```

Pour rendre cette page accessible par tous, créer un lien vers elle dans le fichier `dir` dans le répertoire `info`. Pour trouver ce fichier, taper :

```
locate info | grep dir
```

ce qui donne sans doute le fichier `/usr/info/dir`.

Pour formater ce fichier à l'aide de TEX, il faut d'abord vérifier que le fichier `texinfo.tex` est bien présent (normalement dans le répertoire `inputs` de TEX). Il faut alors lancer la commande `tex` :

```
tex vide.texi
```

Ensuite, il faut générer l'index :

```
texindex vide.??
```

Puis relancer `tex` :

```
tex vide.texi
```

Le fichier `vide.dvi` [3] devrait alors apparaître. Ce fichier peut être lu à l'aide de la commande `xdvi` ou convertit en fichier PostScript à l'aide de la commande `dvips`.

Chapitre 5

Programmation du shell

Le shell est un *interpréteur de commandes*. C'est un programme spécial qui joue le rôle d'intermédiaire en interprétant les commandes passées par l'utilisateur.

5.1 Le shell, un interpréteur de commandes

Il existe plusieurs types de shells, les plus connus depuis Unix ayant une version améliorée sous Linux. Le fichier `/etc/shells` contient une liste de tous les shells disponibles :

```
/bin/ash
/bin/bash
/bin/bash1
/bin/csh
/bin/false
/bin/passwd
/bin/sh
/bin/tcsh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/tcsh
/usr/bin/zsh
```

Les plus connus sont **bash** (version améliorée du shell Bourne sous Unix), **ksh** (version améliorée du shell Korn sous Unix) et **tcsh** (version améliorée du shell C sous Unix). La commande `help` affiche la liste des commandes internes du shell. Par défaut, c'est le shell Bash qui est installé avec Linux. C'est aussi le plus puissant et le plus utilisé, c'est pourquoi c'est celui-ci qui sera utilisé dans les sections suivantes.

L'initialisation du shell bash, à son ouverture, se fait à l'aide de plusieurs scripts, qui représentent autant de possibilités de le personnaliser. Dans l'ordre, les fichiers suivants sont lus et exécutés :

1. le fichier `/etc/profile`, s'il existe.
2. le fichier `$HOME/.bash_profile`, s'il existe
3. sinon le fichier `$HOME/.bash_login`, s'il existe.
4. sinon le fichier `$HOME/.profile`, s'il existe, et si le fichier `$HOME/.bash_profile` n'existe pas. Dans ce cas, le fichier `.bashrc` n'est pas pris en considération, même s'il existe.
5. le fichier système `/etc/bashrc`.
6. le fichier de ressources `.bashrc`, s'il existe.

Dans le cas où bash est invoqué en tant que shell, il n'exécute que les fichiers `/etc/profile` et `$HOME/.profile` s'ils existent.

Pour personnaliser le shell, il faut donc modifier les fichiers `$HOME/.bash_profile` et/ou `$HOME/.bashrc`.

5.2 Les scripts shell

Les scripts shell sont des fichiers exécutables permettant de lancer successivement plusieurs commandes. Pour créer un script shell, il faut éditer un ficher, y entrer les commandes et le rendre exécutable par une commde du type :

```
chmod 755 premier.sh
```

Il faut bien vérifier que le répertoire dans lequel se trouve le script shell est contenu dans la variable d'environnement **$PATH** (voir section 1.10 page 28).

Comme premier fichier, on peut faire, par exemple le fichier `premier.sh` :

```
# Fichier premier.sh.
# Un premier script shell.


echo moi, $USER, faire sur $HOSTNAME mon premier script shell,
echo -n "aujoud'hui, le "
date
echo
```

et l'appeller :

```
premier.sh
```

Un script shell peut en appeller un autre, le niveau du shell change alors comme le montre l'exemple suivant (on parle alors de *sous-shell*) : le fichier `s1.sh` contient :

```
# Script s1.sh.
#
# Exemple d'appel de fichier script (s2.sh, ici) et
# illustration des sous-shells.
#
# Un script shell peut en apeller un autre, avec des parametres,
# y compris lui meme (script recursif).

echo entree s1.sh, niveau du shell = $SHLVL
s2.sh
echo retour s1.sh, niveau du shell = $SHLVL
```

et le fichier `s2.sh` contient :

```
# Script s2.sh.
# Exemple de fichier script appelle (par s.sh, ici) et
# illustration des sous shells.


echo entree-sortie s2.sh, niveau du shell = $SHLVL
```

Lorsqu'on appelle le script `s1.sh`, on obtient alors :

```
entree s1.sh, niveau du shell = 5
entree-sortie s2.sh, niveau du shell = 6
retour s1.sh, niveau du shell = 5
```

5.3 Passons aux choses sérieuses

Voici ce que peut contenir un simple script shell. Ce script montre entre autres comment **lire les arguments** passés par l'utilisateur à la suite du script, **sélectionner les arguments** à traiter, **redéfinir les arguments** si il n'y en a pas qui ont été passés par l'utilisateur, **afficher une variable** dont le contenu a été définit par le script, **lire une entrée clavier** et l'affecter à une variable, **utiliser les** différentes **variables spécialement définies par le shell**, qui contiennent des informations sur les programmes ou sur les utilisateurs…

```
#!/bin/sh


# Exemples de programmation du shell.
# Pour plus d'informations, lire la page de manuel de bash (76 pages !).


echo Exemples de programmation du shell...
echo "Utilisation : simple.sh <fichier> <argument>"


#--------------------#
# lire les arguments #
#--------------------#


echo nom du script : $0
echo argument 1 : $1
echo argument 2 : $2
echo tous les arguments : $*
echo tous les arguments sous forme separee : $@
echo Il y a en tout $# arguments


#-------------------------#
# Redefinir les arguments #
#-------------------------#


a=1
b=2
c=3
d=4
e=5

set a b c d e

echo "Et maintenant, apres le passage de set..."
echo argument 1 : $1
echo argument 2 : $2
echo argument 3 : $3
```

```
echo "Tous les arguments : $* ($# arguments)"

shift

echo
echo "Et maintenant, apres le passage de shift..."

echo "Tous les arguments : $* ($# arguments)"
echo "\$2 devient \$1, \$3 devient \$2..."

set a b c d e

echo
echo "On refait un petit cout de set"
echo "Tous les arguments : $* ($# arguments)"

shift 3

echo
echo "Et maintenant, apres le passage de shift 3..."

echo "Tous les arguments : $* ($# arguments)"
echo "\$4 devient \$1, \$5 devient \$2..."


#--------------#
# echo et read #
#--------------#


echo -n "Entrer deux chaines de caracteres : "
read moi toi
echo "Vous avez saisi : moi=$moi toi=$toi"
echo "Longeur du contenu de \"moi\" : ${#moi}"


# nb: pas d'espace !
moi=10
echo $moi
```

```
echo derniere commande executee par le shell : $?
echo no. de processus du shell : $$
echo PID de la derniere commande executee en tache de fond : $!


echo "Fin du programme !"
echo "Vous me devez \$20 !...Non, je blague"

exit 0 # Valeur de retour du script (contenu dans la variable '$?'
       # a la fin de l'execution du script).

# Il est de bon ton de retourner la valeur 1 en cas d'abandon
# du a une erreur.
```

5.3.1 Utilisation de while

L'instruction **while** permet en autres de tester les arguments passés au programme, et de les traiter. Cette instruction trouvera donc tout naturellement sa place au début d'un script shell digne de ce nom.

```
# Fichier while.sh.
# Exemple d'utilisation de while.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
    echo "Normalement il faut fournir de parametres avec ou sans tiret (-)"
fi


# Exemple de traitement des caracteres. Une option commence par '-'.
# while commence a traiter les arguments a partir de $1 (si on avait
# specifie "while $3" on aurait commence le traitement des caracteres
# a partir de $3...) jusqu'au dernnier. Noter que while efface tous
# les arguments...
```

```
while [ -n "$1" ]
do
  case $1 in
    -*) echo "Option : $1" ;;
    *) echo "Argument : $1 " ;;
esac
shift
done
```

5.3.2 Utilisation de variables

Une variable peut être supprimée par la commande `unset`, et peut être verouillée par la commande `readonly` :

```
moi=mathieu
echo $moi
```

Le résultat s'affiche :

```
mathieu
```

Maintenant, si on vérrouille la variable **moi**, et qu'on essait de la supprimer :

```
readonly moi
unset moi
```

on obtient un message d'erreur :

```
bash: unset: moi: cannot unset: readonly variable
```

Pour qu'un script shell puisse utiliser une variable définie dans le shell par l'utilisateur (dans une fenêtre **xterm**, par exemple), il faut l'exporter avec la commande export :

```
export moi
```

On peut rechercher une chaîne de caractères dans le contenu d'une variable, en fournissant un critère de recherche. Par exemple, pour rechercher la plus petite occurrence du critère de recherche en début de texte de la variable "t" définie par :

```
t="Tentation donnant satisfaction"
```

on utilisera le signe "#" et on entrera la commande :

```
echo ${t#T*i}
```

ou "t" désigne la variable, et "T*i" le critère de recherche. Le résultat est :

```
on donnant satisfaction
```

De même, pour rechercher la plus grande occurrence du critère de recherche en début de texte de la variable "t", on utilisera le signe "##" et on entrera la commande :

```
echo ${t##T*i}
```

Le résultat est *on*. Pour rechercher en fin de texte, on utilisera respectivement "%" et "%%".

```
# Fichier varia.sh.
# Exemple d'utilisation de variables.


# Affectation, calculs...


variable1=5
echo variable1 $variable1

echo -n "variable2 ? "
read variable2
echo variable2 $variable2

variable3=$[variable1+variable2]

# equivalent a la ligne :
#       variable3=$((variable1+variable2))

echo variable3 $variable3


# "declare -i" permet de rendre les calculs plus rapides...

declare -i nombre
nombre=3*9
echo 3*9=$nombre
let entiere=nombre/5
echo partie entiere de $nombre/5 : $entiere

echo "fichier vide (2e argument, vide.txt par defaut) : ${2:-vide.txt}"
```

```
nom="mathieu"

if who | grep "^$nom" > /dev/null 2>&1
then
    echo "L'utilisateur \"$nom\" est connecte"
else
    echo "L'utilisateur \"$nom\" n'est pas connecte"
fi

somme=10
i=3

somme=`expr $somme + $i`

echo somme = $somme

espace=$(du -s)
echo $espace


#------#
# eval #
#------#


# eval sert a evaluer une expression. Ainsi, si :
#       $b contient a
# alors :
#       \$$b contient $a
# et
#       eval echo \$$b
# contient le contenu de la variable a. En fait, eval permet d'outre
# passer les signes "$".

a=Mathieu
echo "a=Mathieu"

b=a
echo "b=a"
echo "\$b contient $b"
```

```
echo "\\\$\$b contient \$$b"

echo "Maintenant avec 'eval' : "
echo -n "eval echo \\\$\$b = "
eval echo \$$b

echo "Parfois il faut combiner plusieurs 'eval' :"

a='$b'
b='$c'
c=fin

echo "a='\$b'"
echo "b='\$c'"
echo "c=fin"

echo "eval echo \$a"
eval echo $a
echo "eval eval echo \$a"
eval eval echo $a
```

Les autres commandes souvent utilisées par les scripts shells sont détaillées dans les sections qui suivent. Chacune d'elles traite d'une instruction particulière.

5.3.3 Utilisation de test

```
# Fichier test.sh.
# Exemple d'utilisation de test.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
    echo "Normalement il faut fournir en premier argument un nom"
    echo "de fichier, comme test.sh, par exemple, soit le numero"
    echo "d'un utilisateur comme indique dans le fichier /etc/passwd"
```

```
    echo "(troisieme champ <user:password:Numero>, 0 devrait marcher)..."
    echo
    echo "Essayer \"$0 $0 0\""
    echo
fi

# $? : valeur de sortie de la derniere commande executee.


#------#
# test #
#------#


# Quelque conditions de test :

# Fichiers...

#       -f : fichier normal.
#       -s : fichier non vide.
#       -d : repertoire.
#       -r : fichier accessible en lecture.
#       -x : fichier accessible en ecriture.

# Variables...

#       -z : variable vide.
#       -n : variable non vide.

# Comparaison d'une variable a un nombre...

#       -eq : egal.
#       -ne : different.
#       -lt : <.
#       -ge : >.
#       -le : <=.
#       -ge : >=.


if test -f "$1"
# forme abregee : if [ -f "$1" ]
```

```
then
    echo Le fichier \"$1\" est bien present.
    echo "Nous allons maintenant lire ce fichier grace a la commande less"
    echo 'Pour quitter less, taper "q"'
    echo "Voulez vous lire le fichier $1 (o pour l'editer) ?"
    read reponse
    if [ $reponse = "o" ]
    then
        less "$1"
    else
        echo "Ok, pas d'edition du fichier..."
    fi
else
    echo "Fichier \"$1\" absent"
fi
```

```
ligne=`grep "^[^:]*:[^:]*:$2:" /etc/passwd`
if [ -n "$ligne" ]
then
    echo "Numero $2 d'utilisateur trouve dans le fichier /etc/passwd"
    echo $ligne
else
    echo "Aucun numero d'utilisateur trouve dans le fichier /etc/passwd"
fi
```

```
fichier=$1

if [ $# = 0 ]
then
    echo "Pas de fichiers donnes a $0"
else
    test -e $1
    echo "Le fichier $fichier est present si 0 est affiche : $?"
fi
```

```
test -r $0 -a -x $0 && echo "$0 : fichier lisible et ecrivable"
```

```
if test \( -r $0 -a -x $0 \) -o \( -r $1 -a -x $1 \)
then
    echo "$0 ou $1 : fichier lisible et ecrivable"
fi


vide=$1
test -s $vide || echo "$vide : fichier vide"


test $vide = vide.txt

if [ $? = 0 ]
then
    echo "La variable vide contient le fichier $vide"
fi


[ -d "../shell" ]
if [ $? = 0 ]
then
    echo "le repertoire shell existe"
fi


L=25
l=7

test "$L" -eq 25 && echo -n "L=25 "
test "$l" -ne 10 && echo -n "l<>10 "
test "$l" -lt 10 && echo -n "l<10 "
test "$L" -ge 15 && echo -n "L>15 "
test "$l" -le 10 && echo -n "l<=7 "
test "$L" -ge 10 && echo -n "L>=26 "


if [ -z "$mathieu" ]
then
    echo "la variable mathieu est vide"
fi
```

```
if [ -n "$L" ]
then
    echo "la variable L est non vide"
fi


( [ -n "$l" ] && echo "la variable l est non vide" ) || echo "la variable
l est vide"
```

5.3.4 Utilisation de if

```
# Fichier if.sh.
# Exemple d'utilisation de if.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
    echo "Normalement il faut fournir le nom d'un fichier"
    echo "Conseil : lancer \"$0 $0\"..."
    echo
fi


# IMPORTANT : 0 est la condition VRAIE.


# Si une simple commande reussit, l'utilisateur en est informe.
# En cas d'erreur, le message du shell est aussi affiche...

if cp "$1" "$1%"
then
    echo "sauvegarde de $1 reussie"
    echo "nous allons maintenant l'editer grace a vi"
    echo 'Pour quitter vi, taper "<Esc>:q!"'
    echo "Voulez vous editer le fichier $1 (o pour le l'editer) ?"
```

```
    read reponse
    if [ $reponse = "o" ]
        then
            vi "$1"
        else
            echo "Ok, pas d'edition du fichier..."
    fi
else
    echo "sauvegarde du fichier $1 impossible"
fi


# Si une simple commande reussit, l'utilisateur en est informe.
# En cas d'erreur, la redirection supprime le  message du shell...
# Seul le message d'erreur du script sera affiche.

if grep "if" $0 >/dev/null 2>&1
then
    echo "if trouve dans $0"
else
    echo "if pas trouve dans $0"
fi
```

5.3.5 Utilisation de case

```
# Fichier case.sh.
# Exemple d'utilisation de case.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
    echo "Le premier argument doit etre le nom d'un utilisateur"
    echo "(root par defaut)..."
    echo
fi
```

```
#------#
# case #
#------#


# case sur une variable d'environnement.

case $LOGNAME in
  root) PS1="# ";;
  mathieu | piou) PS1="Salut $LOGNAME$ ";;
  *) PS1="\h:\w$";;
esac

# Le contenu de la variable d'environnement a ete change. Il faut
# l'exporter et la proteger en lecture.

export PS1
readonly PS1
echo $PS1


# case sur le nombre d'arguments fournis par l'utilisateur.

case $# in
  0) echo "aucun parametre"
     echo "Syntaxe :  $0 <nom d'utilisateur>";;
  1) echo "1 parametre passe au programme : $1";;
  2) echo "2 parametres passes au programme : $1 et $2";;
  *) echo "TROP DE PARAMETRES !"
esac


# case sur un lecture clavier.

echo "Voulez vous continuer le programme ?"
read reponse
case $reponse in
  [yYoO]*) echo "Ok, on continue";;
  [nN]*) echo "$0 arrete suite a la mauvaise volonte de l'utilisateur ;-)"
         exit 0;;
```

```
  *) echo "ERREUR de saisie"
     exit 1;;
esac


# case sur le premier argument fournit par l'utilisateur.

case $1 in
  *[!0-9]*) echo "$1 n'est pas un nombre";;
esac


# Sur quel utilisateur va porter le prochain case ?

if [ $# -lt 1 ]
then
    utilisateur="root"
    echo "utilisateur : $utilisateur"
else
    utilisateur=$1
    echo "utilisateur : $utilisateur"
fi

# Une commande qui cherche si l'utilisateur est logue ou pas.

who | grep "^$utilisateur" > /dev/null 2>&1

# La valeur de retour de la commande precedente ("$?") vaut :
#       0 si l'utilisateur est logue.
#       1 si l'utilisateur n'est pas logue.
#       2 si la syntaxe de la commande precedente est erronee.

case $? in
  0) echo Message envoye a $utilisateur
     mail $utilisateur << Fin
     Et voici $LOGNAME egalement !!
Fin
;;
  1) echo $utilisateur non connecte;;
  2) echo "ERREUR appel errone de la commande grep";;
esac
```

5.3.6 Utilisation de for

```
# Fichier for.sh.
# Exemple d'utilisation de for.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
    echo "Il faut fournir le nom d'un fichier"
    echo "Conseil : commencer par \"$0 $0\"..."
    echo
fi


# for sur des operations mathematiques.

somme=0

for i in 1 2 3 4 5 6 7 8 9 10
do
    somme=`expr $somme + $i`
done

echo "Somme 1->10 : $somme"


# for sur tous les fichiers se terminant par "sh" (scripts shell).

for fichier in *.sh
do
    echo -n $fichier
    echo -n " "

# Sauvegarde de tous les scripts shell.

#     cp $fichier $fichier%
```

```
done

echo


# Pour chaque utilisateur logue, on cherche son numero d'utilisateur
# et on l'affiche.

for nom in [ `who | cut -c1-9` ]
do
    No=$(grep "$nom" /etc/passwd 2> /dev/null | cut -d: -f3)
    if [ -z "$No" ]
    then
# L'utilisateur logue n'a pas de numero d'utilisateur !
        echo "Oops !"
    fi
    echo "$nom : $No"
done


# Affiche la liste des fichiers passes en argument et en fait une
# copie de sauvegarde.

for fichier # eq. a 'for fichier in $@ do' ($@=tous les arguments)
do
    echo -n $fichier
    echo -n " "
#     cp $fichier $fichier%
done

echo
```

5.3.7 Utilisation de set

```
# Fichier set.sh.
# Exemple d'utilisation de set.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
```

```
then
    echo Aucun argument recu !
    echo "Tant mieux, mais essayez a nouveau le script avec comme"
    echo "argument un nom d'utilisateur (si possible logue, comme"
    echo "le votre)..."
    echo
fi


# Si un argument est fournit (un nom d'utilisateur logue), la variable
# "nom" se voit affecter cet argument.

nom=$1


# Cree la liste des parametres si aucun argument n'a ete recu.

echo "Tant pis, \"set\" va en fabriquer..."
echo "Tous les fichiers se terminant par \".sh\" (des scripts shell"
echo "en general) seront les arguments..."

if [ $# = 0 ]
then
    set *.sh
fi


echo "Nombre d'arguments passes a $0 : $#"


# On peut utiliser set pour recuperer le resultat d'une commande.
# Les arguments $1, $2, $3... devient alors les champs de la commande,
# et non les arguments passes par l'utilisateur au script.

# Cherche si le premier argument passe par l'utilisateur au script (et
# sauvegarde dans la variable "nom" plus haut) correspond a un
# utilisateur logue.

set `who | grep "^$nom"`

if [ "$1" = "$nom" ]
```

```
then
    echo "L'utilisateur \"$nom\" est connecte sur $2 depuis le $4 $3 a $5"
else
    echo "L'utilisateur \"$nom\" n'est pas connecte"
fi


# D'autres exemples d'utilisation de "set".

set `date` && [ $2 = "Nov" ] && [ $3 = "19" ] \
&& echo "Bon anniverssaire, $USER "

set `grep "^$nom" /etc/passwd | cut -d: -f1,3,4,6 | tr ":" " "`

if [ "$1" = "$nom" ]
then
    echo "L'utilisateur \"$nom\" a un UID = $2 et un GID = $3"
    echo "son repertoire personnel est $4"
else
    echo "L'utilisateur \"$nom\" est inconnu"
fi


#-----#
# set #
#-----#


bizard=xzorglub
echo "bizard=xzorglub"

grep "^$bizard" /etc/passwd
echo "on ne voit rien a l'ecran quand il ne se passe rien..."

# 'set -x' peut servir au debogage d'un script.
# Les commandes sont alors affichees telles qu'elles seront executees.
# Taper 'sh -x simple.sh' dans le shell = 'set -x' pour tout le programme.

# Une option activee sera desactivee en remplacent - par + :
# ex. : set +x --> desactive set -x
```

```
set -x

grep "^$bizard" /etc/passwd
echo "maintenant si..."

# apres 'set +x', plus d'affichage de commandes !

set +x

# 'set -u' affiche un message d'erreur en cas de variable non definie
# et quitte le script. Enlever le "#" pour tester. Le script se terminera
# aussitot.

echo $contenu
# set -u
echo $contenu
contenu="-salut"
echo $contenu


# La variable "contenu" sera affectee du premier parametre de position...

set - $contenu
echo "Option \$1 : $1"
```

Les principales options de la commande `set` sont données dans le tableau 5.1.

Option	**Signification**
-o emacs/vi	Affiche l'historique des commandes éditées par `emacs/vi`.
ignoreeof	`<Ctrl-d>` sans effet.
noclobber	Lors d'une redirection avec "`>`", il n'y aura pas d'écrasement. Utiliser "`>/`" pour écraser quand même.
allexport	Exporte toutes les variables définies. Affiche la liste de tous les paramètres et leurs état (on/off).

TAB. 5.1 – Principales options de la commande **set**.

5.3.8 Utilisation de fonctions

```
# Fichier fonction.sh.
# Exemple d'utilisation de fonctions dans un script shell.


# Definition de la fonction moi. L'argument "$1" est celui qui sera
# fournit a la fonction, et nom au script...

moi() {
echo Aujourd\'hui nous sommes le `date`.
echo Bienvenu sur $1.
}


# Utilisation de la fonction moi.

echo Bonjour $USER !
moi $HOSTNAME
echo Salut !
```

5.3.9 Utilisation de select

```
# Fichier select.sh.
# Exemple d'utilisation de select.


# S'il n'y a pas de parametres fournis par l'utilisateur...

if [ $# = 0 ]
then
    echo Aucun argument recu !
    echo "$0 risque de ne pas bien marcher..."
    echo
fi


#--------#
# select #
#--------#
```

```
# Affichage du prompt definit par "PS3".

PS3="Numero (4 ou <Ctrl>-d pour continuer) : "

# Debut de la boucle select.

select nom in mathieu piou root continuer
do

# Si la variable "nom" contient quelquechose (saisie clavier de
# l'utilisateur non vide).

    if [ -n "$nom" ]
    then
        echo
        echo "$nom a ete choisit"
        echo

# Si "continuer" a ete selectionne par l'utilisateur, in sort de la
# boucle select.

        [ $nom = "continuer" ] && break
    fi
done


# Affichage du prompt.

PS3="Argument passes : "

# Sans precision de "in", on selectionne dans les arguments du script.

select argument
do

# Si il y a des arguments passes par l'utilisateur et que le choix de
# l'utilisateur est une saisie clavier non vide.

if [ $argument != "" ] && [ $# != 0 ]
then
```

```
    echo
    echo "Argument selectionne : $argument "
    echo
    break
else
    echo "Pas d'arguments passes au programme $0..."
fi
done
```

5.3.10 Utilisation de trap

```
# Fichier trap.sh.
# Exemple d'utilisation de trap.


# La sequence de touches <Ctrl-c> sera sans effet.

trap '' 2 3


# La sequence de touches <Ctrl-c> a retrouve son etat normal
# (quitter le script).

trap 2 3


# Pour le test de trap qui suit, ceci est necessaire.

echo "Creation du repertoire mathieu s'il n'existe pas deja"
echo
[ -d mathieu ] || mkdir mathieu


# La sequence de touches <Ctrl-c> fera la commande entre ' '
# Ca peut etre utile pour faire une action juste avant de quitter le script
# par exemple : trap 'rm -f  /tmp/fic_test; exit 1;' 2 3
# un peut de menage, meme en cas d'arret involontaire (<Ctrl-c>), ne peut
# pas faire de mal !

trap '[ -d mathieu ] && rmdir mathieu;' 2 3
```

```
echo "Test de trap, appuyer sur <Ctrl-c> pour le test."
echo "Ca effacera le repertoire mathieu."
echo "Sinon, le test de trap ne se fera pas, et le repertoire mathieu"
echo "ne sera pas efface (verifier qu'il existe dans ce cas la)."
echo
echo "Entrer ensuite une variable"
read variable
echo "Ok, variable saisie : $variable..."
```

5.3.11 Autres commandes

Les commandes qui suivent sont d'emploi déconseillé car nuisant à la compréhension du script :

- **break**, sort de la boucle, **break n** saute les n boucles suivantes.
- **continue**, refait la boucle, **continue n** refait la n^{eme} boucle précédente.
- la commande " **:**" retourne la valeur 0. Elle ne fait rien. En clair, on peut remplacer la syntaxe "*while true*" par "*while :*"...On peut faire une boucle infinie du type 'while true' et en sortir par la commande `exit`, `break` ou `continue`.

Chapitre 6

Programmation en Tcl/Tk

Les scripts **Tcl** ont pour principal intérêt de pouvoir être exécutés par **Tk**, qui à l'aide de quelque dizaines de commandes suplémentaires, permet d'offrir une **interface graphique**. La section qui suit donne un appercu des possibilités de programmation en Tcl. La section d'après donnera un appercu des possibilités de programmation en Tk. Il existe pour presque chaque commande une page de manuel, et en principe de la documentation située en général dans le répertoire `/usr/doc/`.

6.1 Programmation avec Tcl

Un exemple simple pourrait être un script qui compte les lignes d'un fichier, et affiche le résultat à l'écran. Cet exemple utilise des commandes très souvent utilisées : lecture des arguments, ouverture et lecture d'un fichier, calculs, affichage du résultat à l'écran...Cet exemple tient en très peu de lignes et montre la puissance d'un script shell.

```
#!/usr/bin/tclsh


# Fichier lc.tcl.
# Un exemple tres simple d'utilisation de Tcl.

# Essayer "lc.tcl lc.tcl"...


if {$argc != 1} {
error "lc <fichier>"
}
```

```
set lefichier [open [lindex $argv 0] r]
set compte 0

while {[gets $lefichier ligne] >= 0} {
    set compte [expr $compte + 1]
}

puts "Lu $compte lignes."
```

Allez, on passe au gros morceau…

```
#!/usr/bin/tclsh


# Fichier simple.tcl.
# Exemple de programmation en Tcl.


puts ""
puts "Salut a \
tout le monde !"

set long [string  length "Salut a \
           tout le monde !"]
puts "$long characteres dans la phrase precedente..."

puts "Il y a [string length bonjour] caracteres dans 'bonjour'"

# Les {} forment un seul argument...

puts {Il y a [string length bonjour] caracteres dans 'bonjour'}
puts ""


# else doit se trouver sur la meme ligne que l'accolade fermant le if.
# L'espace entre if/else et '{' est primordial !

set pi 3.1416
if {$pi==3.1416} {puts "pi=$pi"} else {puts "pi <> 3.1416"}
```

```
# L'exemple qui suit montre elseif et comment lire une entree.

puts ""
puts "entrer un chiffre 0<chiffre<10 :"
set chiffre [gets stdin]
if {$chiffre >=7} {
    puts "chiffre >=7"
} elseif {$chiffre >=4 && $chiffre < 7} {
    puts "4<=chiffre<7"
} elseif {$chiffre >=2 && $chiffre<4} {
    puts "2<=chiffre<4"
} else {
puts "chiffre <1 ou chiffre >10"
}
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Exemple d'utilisation de la boucle while :
# 1=vrai et utilisation de 'continue' et 'exit'.

set i 0
set somme 0

while {1} {
    incr i ;# i++
    if {$i==5} {continue}; # sauter si i=5
    if {$i>10} {break}; # fin de la boucle i
    set somme [expr $somme+$i];

puts $i
}

puts "Somme=$somme"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin
```

```
for {set i 0; set somme 0} {$i<=10} {incr i} {
    set somme [expr $somme+$i]
puts $i
}
puts "Somme $somme"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Exemple d'utilisation de la commande 'foreach'.

set somme 0


# La variable 'i' n'a pas besoin d'etre initialisee...

foreach i { 1 2 8 9 10 } {
    set somme [expr $somme+$i]
puts $i
}
puts "Somme $somme"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Autre exemple pour lire les mots d'une phrase.

set mathieu "root c'est sympa"
foreach mot $mathieu {
puts "$mot"
}
puts ""


# Exemple d'affichage d'un prompt.
```

```
set prompt "Entrer la commande (\"help\" ou \"quit\" pour sortir) :"

puts "$prompt"

# On peut aussi mettre :
#    puts -nonewline "$prompt"
# pour que les entrees clavier soient affichees sur la meme ligne...


while { [gets stdin commande] != -1} {
    switch -exact -- $commande {
        help {puts "help quit start draw"}
        quit {puts "Au revoir !"; break}
        start {puts "Demarrage"}
        draw {puts "Dessine"}
        default {puts "Commande inconnue : $commande"}
}
puts "$prompt"
}
puts ""


# Pour comparer avec une expression rationnelle, utiliser '-regexp'

puts "******************************************"
puts " Entrer une ligne commencant par q, s ou d "
puts "******************************************"
puts ""

puts "$prompt"

while { [gets stdin commande] != -1} {
    switch -regexp -- $commande {
        ^q.* {puts "Bye !"; break}
        ^s.* {puts "commance par \"s\""}
        ^d.* {puts "commance par \"d\""}
        default {puts "Commande inconnue : $commande"}
}
puts "$prompt"
}
puts ""
```

```
# Exemple d'utilisation d'une fonction.
# Definition de la fonction.

proc total elements {
set somme 0
    foreach i $elements {
        set somme [expr $somme+$i]
    }
return $somme
}

# Utilisation de la fonction avec 1 argument.

set comptes "5 4 3 5"
puts "Total=[total $comptes]"

# Utilisation de la fonction avec plusieurs arguments.

set somme1_10 [total {1 2 3 4 5 6 7 8 9 10}]
puts "et de 1->10 : $somme1_10"

# Le nom de l'argument de la fonction a peu d'importance...

proc total args {
set somme 0
    foreach i $args {
        set somme [expr $somme+$i]
    }
return $somme
}

set somme1_10 [total 1 2 3 4 5 6 7 8 9 10]
puts "et de 1->10 bis : $somme1_10"
puts ""


# Utilisation de la commande string.

puts "devinez mon prenom..."
```

```
gets stdin commande
if {[string compare $commande "mathieu"]==0} {
puts "gagne !"
} else {
puts "perdu !"
}
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Utilisation de tablaux.
# Le tableau partition contient le nom des partitions.
# Le tableau disque_utilise contient la taille de la partition.

set partition(1) "/Dos"
set partition(4) "/"
set partition(5) "/home"
set partition(6) "/usr"

set disque_utilise($partition(1)) 701
set disque_utilise($partition(4)) 94
set disque_utilise($partition(5)) 48
set disque_utilise($partition(6)) 756

foreach n {1 4 5 6} {
puts "Disque utilise par $partition($n) = $disque_utilise($partition($n))M"
}


# On peut aussi utiliser des variables d'environnement...

puts "la variable PATH a pour valeur $env(PATH)"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Exemple d'ecriture dans un fichier.
```

```
set fichier [open "fictest" w]
puts $fichier "hello !"
close $fichier


# Les deux formes suivantes sont equivalentes :
#     set line [gets stdin]
#     gets stdin line


# Il y a 3 facons de lire un fichier :

puts "Il y a 3 facons de lire un fichier :"
puts ""

puts "Facon 1 : ligne par ligne."
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin
puts "---------------------------------------------------------"

set pass [open "/etc/passwd" r]
while { [gets $pass line] != -1} {
puts $line
}
close $pass

puts ""
puts "Facon 2 : par packets de 20 octets."
puts "---------------------------------------------------------"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin

# Par packets de 20 octets.

set f1 [open "/etc/passwd" r]
while {![eof $f1]} {
```

```
    set buffer [read $f1 20]
puts $buffer
}
close $f1

# Par caractere ("\n", fin de ligne ici).

puts ""
puts "Facon 3 : par caractere (\"\\n\", fin de ligne ici)."
puts "-----------------------------------------------------"
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin

set f3 [open "/etc/passwd" r]
set buffer [read $f1]
split $buffer "\n"

foreach Axelle $buffer {
puts $Axelle
}
close $f3
puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Pour tous les fichiers se terminant par '.tcl'.

puts "Fichiers scripts Tcl :"
foreach fichier [glob *.tcl] {
puts $fichier

# On peut ensuite ouvrir chaque fichier en lecture par ex et le traiter...
}

puts ""

set outfile [open simple.out w];
```

```
# Execution d'une commande UNIX.
# On peut rediriger la sortie ou l'entree, utiliser des pipes...

exec ls

# Equivalant de "ls *.tcl".

eval exec ls [glob *.tcl];


puts ""
puts "Appuyer sur une touche pour continuer..."
puts ""
gets stdin


# Exemple de traitement des lignes d'un fichier.

set fid [open "/etc/passwd" r];

# Recuperation des informations ligne par ligne.
# Decoupage de chaque ligne en champs separes par ":".
# Extrait du premier champ (indice 0).

while {[gets $fid line] != -1} {
    set fields [split $line ":"];
    puts [lindex $fields 0];
}


# join rassemble plusieurs elements en un seul, en precisant le
# separateur de champ.
# Ici, les elements sont separes par un espace.

puts ""
set x {1 2 3 4 5 6}
set y [join $x ""]
puts $y
```

6.2 Programmation avec Tk

Le plus simple est de commencer avec le fameux “Salut” :

```
#!/usr/X11R6/bin/wish -f


# Fichier tksalut.
# Affiche le salut et permet de quitter a l'aide d'un boutton.


# On definit 2 "widgets" (boutton, menu, barre d'asenceur...) :
#    .msg est une etiquette ("label") et affiche un message
#    .bye est un boutton auquel est associe une commande si on
#    clique dessus.

label .msg -text "Salut !"
button .bye -text "Bye" -command {exit}


# Maintenant, on place les widgets...

pack .msg .bye
```

La figure 6.1 montre le résultat de ce script.

FIG. 6.1 – Le fameux **Salut** avec Tk.

Dans l'exemple qui suit, on peut même saisir un texte et exécuter une commande :

```
#!/usr/X11R6/bin/wish -f


# Fichier tkedit.
```

```
# Saisit du nom d'un fichier et edition de ce fichier.


# 3 widgets : Affichage d'un texte (.l), saisie d'un texte (.e) et
# boutton pour quitter (.bye).

label .l -text "Fichier :"

entry .e -relief sunken -width 30 -textvariable fname

button .bye -text "Bye" -command {exit}


# On place les widgets. L'option "-padx" precise l'espace horizontal
# libre a gauche et a droite du widget, et "-pady" precise l'espace
# vertical libre.

pack .l -side left
pack .e -side left -padx 1m -pady 1m
pack .bye -side left


# En cas de saisie se terminant par "<Control-c>", on quitte le
# script en affichant un message.

bind .e <Control-c> {puts "Merci d'avoir utilise tkedit.\n d'apres
un exemple de \"Le systeme Linux\", par Matt WELSH"; exit }


#  En cas de saisie se terminant par "<Return>", on lance une fenetre
# xterm et on edite le fichier.

bind .e <Return> {
exec xterm -e vi $fname
}
```

Le script qui suit montre quelque “ficelles”, comment placer les widgets, gérer les évenements souris ou clavier. . .La plupart des commandes devraient avoir leur place dans un script Tk digne de ce nom. . .

```
#!/usr/X11R6/bin/wish -f
```

```
# Fichier tkmin.
# Le minimum que devrait contenir un script digne de ce nom.


# Creation et configuration des widgets.


# Taille du widget "." (fenetre principale).

wm geometry . 600x400

# Creation d'un widget cadre pouvant contenir d'autres widgets.

frame .f

# Creation de deux boutons :
#    + le premier, contenu dans cadre .f, qui porte le texte "Fichier".
# Ce bouton se voit affecter des couleurs de fond et de devant.
#
#    + le second, .bh, qui porte le texte "Ouvrir un fichier"

button .f.b -text Fichier -fg blue -bg yellow
button .bh -text "Ouvrir un fichier"

# On peut configurer le bouton apres sa creation.

.bh config -bg yellow

# Creation d'un widget pouvant saisir du texte.

text .text


# Assignation des evenements claviers ou souris.


# L'exemple qui suit peut servir pour afficher une aide interactive.

# Si le pointeur de la souris arrive sur le widget .f.b ("<Enter>"),
```

```
# afficher le bouton .bh ("place").

bind .f.b <Enter> {place .bh -in .f.b -relx 0.5 -rely 1.0}

# Si le pointeur de la souris quitte le widget .f.b ("<Leave>"),
# effacer le bouton .bh ("place forget").

bind .f.b <Leave> {place forget .bh}

# Si on appuie sur <Control-c> dans le widget .text, quitter
# le programme.

bind .text <Control-c> exit

# Association de evenements clavier dans le widget .text.
#    %A designe le caractere imprimable que tape l'utilisateur.
#    { } designe un caractere non imprimable.
#    %W designe le widget recevant la frappe.

bind .text <KeyPress> {
if { "%A" != "{ }" && "%A" != "a"} {%W insert insert %A}
}

# Si on appuie sur le bouton gauche de la souris ("<ButtonPress-1>"),
# et si on le relache ("<ButtonRelease-1>") dans le widget .text.
# Pour le bouton du milieu -2 et celui de droite -3.
#
# %x et %y designent les coordonnees du pointeur de la souris.

bind .text <ButtonPress-1> {puts "pointeur en (%x, %y) sur le widget %W"}
bind .text <ButtonRelease-1> {puts "plus pointeur en (%x, %y) sur le widget
%W"}

# Racourcis : <ButtonPress-1>=<Button-1> et <KeyPress-q>=<q>.

# Le bouton pour quitter.

button .f.q -text Quitter -command {exit} -fg blue -bg yellow


# On place les widgets.
```

```
# Affiche contre le cote specifie ("left", "right", "top" ou "bottom").

pack .f.b .f.q -side left

# Si le widget conteneur est plus grand que le widget contenu, il
# peut s'etendre en "both", "none", "x" ou "y".

pack .f -fill x

# Autre facon de placer les widgets : place.
#
# "-relx" et "-rely" precisent la position du widget en fraction
# horizontale et verticale du widget conteneur (entre 0.0 et 1.0).
# "-anchor" precise la position du point d'ancrage de la fenetre.
# Options : "center", "e", "n", "ne", "nw", "s", "se", "sw", "w".
# L'option par defaut est "nw".

place .text -relx 0.5 -rely 0.15 -anchor n
```

Le script qui suit montre un exemple concret et utile : dessiner des ronds et des rectangles, en utilisant des menus et la souris.

```
#!/usr/X11R6/bin/wish -f


# Fichier tkdraw.
# Un script qui dessine.


# Initialisation des variables globales servant a stoquer les objets
# et leurs positions.

set oval_count 0
set rect_count 0
set orig_x 0
set orig_y 0


# Definition des fonctions.
```

```
# Fonction set_oval : dessine une ovale.

proc set_oval {} {

# Pour que Tcl ne croies pas que ces variables soient locales.

global oval_count orig_x orig_y

# Le . indique la fenetre principale, .c le widgets "canvas" contenu dans
# la fenetre principale. Il y a une hierarchie cf .mbar.file.menu : la
# barre horizontale contient Fichier qui contient un menu
# (voir plus loin).

# Lorque le bouton 1 est appuye, cree une ovale.

bind .c <ButtonPress-1> {

set orig_x %x
set orig_y %y
set oval_count [expr $oval_count + 1]

# -tags sauvegarde cette ovale sous un nom oval1, oval2...

.c create oval %x %y %x %y -tags "oval$oval_count" -fill red
}

# Bouton 1 de la souris + deplacement de la souris : efface l'ovale
# courante et remplace par une nouvelle.

bind .c <B1-Motion> {
.c delete "oval$oval_count"
.c create oval $orig_x $orig_y %x %y  -tags "oval$oval_count" -fill red
}
}
# Fin de la fonction set_oval.

# Fonction set_rect : dessine un rectangle.

proc set_rect {} {
```

```
global rect_count orig_x orig_y

bind .c <ButtonPress-1> {

set orig_x %x
set orig_y %y

set rect_count [expr $rect_count + 1]

.c create rectangle %x %y %x %y -tags "rect$rect_count" -fill blue
}

bind .c <B1-Motion> {
.c delete "rect$rect_count"
.c create rectangle $orig_x $orig_y %x %y  -tags "rect$rect_count" -fill
blue
}
}
# Fin de la fonction set_rect.

# Facon plus rapide :
#    set objtype ...
#    .c create $objtype %x %y %x %y -tags "obj$obj_count" -fill blue


# Creation de la barre des menus .mbar.
# -relief cree un sillon autour et -bd specifie l'epaisseur du cadre
# (sillon ici).

frame .mbar -relief groove -bd 3

# Positionnement de .mbar dans la fenetre principale.
# -fill x indique a pack qu'il devra occuper toute la largeur de la
# fenetre qui le contient et -expand qu'il devra grossir pour occuper
# cet espace (cf. page de manuel de pack).

pack .mbar -side top -expand yes -fill x

# Creation des menus Fichier et Objet fils du .mbar.
```

```
menubutton .mbar.file -text "Fichier" -menu .mbar.file.menu
menubutton .mbar.obj -text "Objets" -menu .mbar.obj.menu

# Positionnement par rapport au widget pere (ici .mbar).

pack .mbar.file .mbar.obj -side left

# Creation du menu Fichier et ajout de l'item Quitter.

menu .mbar.file.menu
.mbar.file.menu add command -label "Quitter" -command { exit }

# Creation du menu Objet et ajout des items Ovales et Rectangles.

menu .mbar.obj.menu

# Facon plus "classique".

#.mbar.obj.menu add command -label "Ovales" -command { set_oval }

# Avec des boutons coloriees si l'option est validee ou non.

.mbar.obj.menu add radiobutton -label "Ovales" -variable objtype \
    -command { set_oval }

.mbar.obj.menu add radiobutton -label "Rectangles" -variable objtype \
    -command { set_rect }

# Creation du widget canvas .c.

canvas .c
pack .c -side top

# Initialisation des ovales, en invoquant le 1er item du menu Objet.

.mbar.obj.menu invoke 1
```

Voila, une fois de plus en quelque ligne on a crée un super programme avec une non moins superbe interface graphique (figure 6.2)...

La plupart des commandes citées on une page de manuel très détaillée avec toutes les options disponibles, bien plus que celles présentées ici...

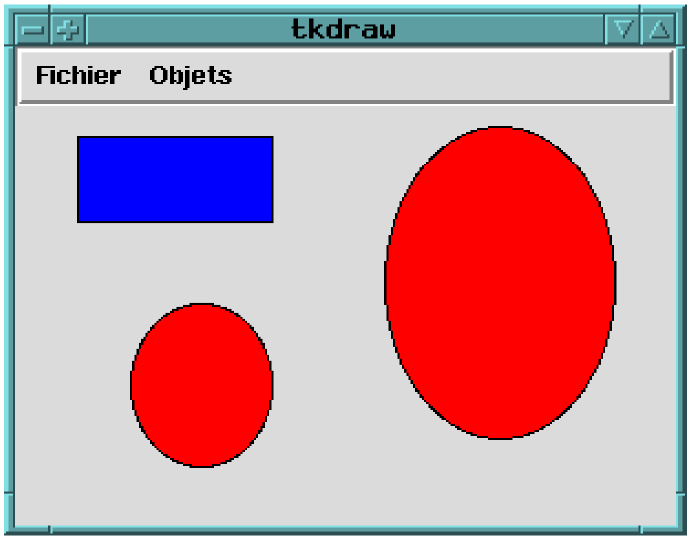

FIG. 6.2 – Un superbe programme Tk en quelque lignes.

Chapitre 7

Programmation en Perl

Les scripts **Perl** est souvent utilisés par les pages Web. Ils permettent de rendre interactive une simple page Web. Ce langage est très simple et très polyvalent, si bien qu'il existe souvent plusieurs facons d'effectuer la même tâche. Par exemple, les expressions :

```
while ($_ = <STDIN>) {print;}

while (<STDIN>) {print;}

for (;<STDIN>;) {print;}

print while $_ = <STDIN>;

print while <STDIN>;
```

sont toutes équivalentes. Cette polyvalence peut être génante, mais c'est à l'utilisateur de choisir la forme qui lui plaît.

Le script qui suit montre comment utiliser des variables, des tablaux, comment effectuer des opérations sur les variables, qu'elles soient des chiffres ou des chaînes de caractères...

Noter que le chemin indiqué au début du script peut aussi etre `/usr/local/bin/perl` sur certains systèmes. Pour s'en assurer, la commande `which perl` donnera le bon chemin.

```
#!/usr/bin/perl

# Script de demonstration de perl.
```

```
print "\n";


#-----------#
# variables #
#-----------#


# Scalaires.

$age=22;
$dents=28;
$total=$age+$dents;
$utilisateur="Mathieu";

print "$utilisateur a $age ans, $dents dents et $total age+dents\n";
print '$utilisateur a $age ans, $dents dents et $total age+dents\n';

undef $nom;

#$nom="moi";

if (!(defined $nom)) {
    print "variable nom non dfinie !\n" ;
}

# Matricielles.

@experience=(2, 5, 3);
@languages=("Perl", "Tcl/Tk", "Bash");
$nombre=@languages;

print "il connait $nombre languages Linux :\n";
for ($i=0; $i<$nombre; $i++) {
    print "$languages[$i] depuis $experience[$i] semaines\n";
}

print "dernier indice de la matrice \@languages : $#languages\n";
print "\@languages : (@languages)\n";
print "\n";
```

```
# Matricielles associatives.

%def_partitions=(1, "\/Dos", 4, "\/", 5, "\/home", 6, "\/usr");

foreach $i (1, 4, 5, 6) {
    print "partition $i point de montage $def_partitions{$i}\n";
}
print "\n";

%def_espace=("\/Dos", 701, "\/", 94, "\/home", 48, "\/usr", 756);

foreach $i ("\/Dos", "\/", "\/home", "\/usr") {
    print "partition $i espace disque occupe $def_espace{$i}M\n";
}

print "\n";
print "Appuyer sur une touche pour continuer...\n";

<STDIN>;

# Une matrice associative contenant les variables d'environnement est
# definie par Perl : %ENV.

#print "PATH=$ENV{PATH}\n";
print "\n";

# @ARGV contient les arguments -> $ARGV[0], ARGV[1]...
print "$ARGV[0] $ARGV[1]\n";

print "id. du script : $$\n";
print "id. de l'utilisateur utilisant le script : $<\n";
print "etat renvoye par le dernier appel system : $?\n";
print "argument par defaut : $_\n";
print "nom du script : $0\n";

# cf. (pages de manuel, par exemple) "delete", "each", "keys", "value"
# qui sont des operateurs sur les matrices.

# "shift" permet de passer a l'argument suivant.

$premier=shift @languages;
```

```
print "premier argument de \@languages : $premier\n";

$second=shift @languages;
print "second argument de \@languages : $second\n";


#------------#
# operateurs #
#------------#


# Pour les chaines de caracteres, les operateurs (==, !=, <, >, <=, >=)
# sont remplaces par (eq, ne, lt, gt, le, ge).

# Valeur de retour de "cmp" : -1 si <, 1 si >, 0 si =. C'est l'ordre
# alphabetique qui est pris en compte.

if (("ab" cmp "ac")==-1) {print "<\n";}
if (("ad" cmp "ac")==1) {print ">\n";}
if (("ad" cmp "ad")==0) {print "=\n";}

# "**" designe la puissance.

$x=2;
$y=3;
$z=$x**$y;

print "x=$x y=$y x^y=$z\n";

# Pour elever au carre...

$y **=2; print "y*y=$y\n";

# "()" initialise a 0.

print "\@experience : (@experience)\n";
@experience=();
print "\@experience : (@experience)\n";

# "." concatene 2 chaines.
```

```
$chaine1="Salut";
$chaine2=" le monde";
$message=$chaine1.$chaine2;

print "$message\n";

# "x=" repete un certain nombre de fois une operation.

$marqueur="*";
$marqueur x= 14;

# Affiche 14 '*'.

print "$marqueur\n";

# ".." designe un operateur de plage.

@chiffres=(1..9);
@alphabet=("a".."z");

print "@chiffres\n";
print "@alphabet\n";
print "\n";

$user="mathieu";
#$user="root";

if ($user eq "root") {
print "root c'est sympa...mais dangereux !\n";
}
else {
print "Desole $user, vous devez etre \"root\" !\n";
}
```

Le script qui suit montre comment utiliser les boucles, les écritures ou lectures dans un fichier, les appels systèmes, les expressions rationnelles…

```
#!/usr/bin/perl

# Script de demonstration de perl.

print "\n";
```

```
#----------------------------#
# if...else...elsif...unless #
#----------------------------#


# Lecture clavier.

print "Entrer un chiffre 0<chiffre<10 :\n";

$chiffre=<STDIN>;

# Supprime l'interligne apres la derniere ligne.

chop $chiffre;

if ($chiffre >=7) {
    print "chiffre >=7\n";
}
elsif ($chiffre >=4 && $chiffre < 7) {
    print "4<=chiffre<7\n";
}
elsif ($chiffre >=2 && $chiffre<4) {
    print "2<=chiffre<4\n";
}
else {
print "chiffre <1 ou chiffre >10\n";
}

# unless execute la commande si la condition est fausse...

unless ($user eq "root") {
    print "mais puisque je te dis que tu n'es pas root, $user !\n";
}


#-------------------#
# while for foreach #
#-------------------#
```

```
# 4 facons d'utiliser une boucle...

# Facon 1.
# "@chiffres" est un tableau.

@chiffres=(1..9);

while (@chiffres) {
    $i=shift @chiffres;
    print "$i\n";
}

# Facon 2.

$somme=0;
$i=0;

while (1) { # toujours verifie.
    $i++;
    if ($i==5) {next}; # saute a l'iteration suivante.
    if ($i>10) {last}; # fin de boucle.
    $somme+=$i;
}
print "somme 1 -> 10 sans 5 : $somme\n";

# Facon 3.

for ($i=0, $somme=0; $i<=20; $i++) {
    $somme+=$i;
}
print "somme 1 -> 20 : $somme\n";

# Facon 4.

$somme=0;
$i=0;

foreach $i(1..30) {$somme+=$i;}
print "somme 1 -> 30 : $somme\n";
```

```
$somme=0;
$i=0;

foreach $i(1,30, 120) {$somme+=$i;}
print "somme 1+30+120 : $somme\n\n";


#----------------#
# appels systeme #
#----------------#


$prompt="Commande (ou \"exit\") : ";

print "essai d'appel systeme 1\n";
print $prompt;

while (<STDIN>) {
    chop;

# "$_" designe la derniere entree (ici <STDIN>).

    if ($_ eq "exit") {last;}
    system $_;
unless ($?==0) {print "Erreur d'execution : $_\n";}

print $prompt;
}

# Pareil en utilisant fork et exec (cree un processus fils qui execute
# la commande demandee, puis ferme le processus fils et revient au
# processus pere).

print "essai d'appel systeme 2 (en utilisant fork et exec)\n";
print $prompt;

while (<STDIN>) {
    chop;
    if ($_ eq "exit") {last;}
    $statut=fork;
    if ($statut) {
```

```
        wait; # Attends dans le processus pere la fin du processus fils.
        print $prompt;
        next;
    }
    else
    {
        exec $_}
}

print "\n";

# Ouverture du fichier "/etc/passwd" en lecture.
# Ouverture du fichier "sortie" en ecriture( ">").

open (fichier, "/etc/passwd");
open (out, ">sortie");

# Tant qu'il y a une ligne dans le fichier ouvert...

while(<fichier>) {

# Si un utilisateur n'a pas de shell, on l'inscrit dans le fichier
# de sortie.

    if ($_ =~ /:\n/) {print out $_}
}

close fichier;
close sortie;


# Envoi d'un mail.
# Si le nom du fichier dans "open" est precede de "|", ce nom est
# considere comme une commande.

foreach ("root", "mathieu"){
    open (message, "| mail -s essai $_");
    print message "Ca marche...\n";
    close message;
}
```

```
# Si le nom du fichier dans "open" est suivit de "|", ce nom est
# considere comme une commande qui sera executee.

open (pl, "ls -l *.pl |");
while (<pl>) {
    print $_;
}

print "\n";

# Exemple de fonction.
# Les arguments passes a la fonction sont stoques dans la matrice "@_".

sub hello {

# Copies locales des arguments.

    local($premier, $dernier)=@_;
    print "Bonjour, $premier, $dernier\n";
}

$a=$ENV{USER};
$b=root;

# Appel de la fonction "hello".

&hello($a,$b);

# Appel du script "salut.pl" qui contient la definition de la fonction
# "salut".

require 'salut.pl';

# Appel de la fonction "salut".

&salut($a,$b);

print "\n";


#-------------------------#
```

```
# expressions rationelles #
#-------------------------#


# Les expressions rationelles sont rangees entre 2 "/".
# Le signe "=~" signifie : "si c'est equivalent du modele".
# Le signe "!~" signifie : "si c'est different du modele".


print "Essai expressions rationnelles...\n";
print "quit pour quitter\n";

while (<STDIN>) {
    print "le texte suivant doit comporter 3 espaces au debut
(quit pour sortir)\n";
    if ($_ =~ /^{3}\s/) {print "3 espaces au debut...\n\n"}
    if ($_ !~ /\)\n/) {print "ne se termine pas par ')'...\n\n"}
    if ($_ =~ "quit") {exit}
}
```

Le fichier `salut.pl` demandé par le script précédent pourrait ressembler à cela :

```
# Exemple de fichier annexe Perl.

# Ce fichier contient simplement une fonction.
# Pour utiliser cette fonction, mettre dans le script Perl :
#   require 'salut.pl';
# avant l'appel de la fonction "salut".

sub salut {
    local($premier, $dernier)=@_;
    print "Salut, $premier, $dernier\n";
}
1; # valeur de retour.
```

A propos des expressions rationnelles, beaucoup utilisées par Perl : celles ci permettent de rechercher une chaîne de caractère suivant un modèle précis. Des exemples de modèles ont déjà été évoqués à propos de la commande `grep` (voir section 1.3 page 14). Le tableau 7.1 présente d'autres modèles pouvant être utilisés.

Expression	Signification
`.`	Correspond à **tout caractère**, sauf le caractère interligne.
`x*`	Correspond à **0 ou plusieurs** occurrences du caractère "**x**".
`x+`	Correspond à **1 ou plusieurs** occurrences du caractère "**x**".
`x?`	Correspond à **0 ou 1 occurrences** du caractère "**x**".
`x[3]`	Correspond à **3** occurrences du caractère "**x**".
`x[3,]`	Correspond à **au moins 3** occurrences du caractère "**x**".
`x[,7]`	Correspond à **au plus 7** occurrences du caractère "**x**".
`x[3,7]`	Correspond à **au moins 3** et **au plus 7** occurrences du caractère "**x**".
`[...]`	Correspond à **n'importe lequel** des caractères mis entre crochet.
`$`	Correspond à la **fin d'une ligne**.
`\0`	Correspond au **caractère nul**.
`\b`	Correspond à un **retour arrière**.
`\B`	Correspond à **tout caractère** d'un mot, **sauf le premier et le dernier**.
`\b`	Correspond au **début ou à la fin d'un mot** (s'il n'est pas entre crochets).
`\cX`	Correspond à **Ctrl-x**.
`\d`	Correspond à **un seul chiffre**.
`\D`	Correspond à un caractère **non numérique**.
`\f`	Correspond à un **saut de page**.
`\n`	Correspond à un caractère **retour ligne**.
`\147`	Correspond à la **valeur octale** 147.
`\r`	Correspond à un **retour chariot**.
`\S`	Correspond à un caractère **d'espacement non blanc**.
`\s`	Correspond à un caractère **d'espacement blanc** (*espace, tabulation, interligne*).
`\t`	Correspond à une **tabulation**.
`\W`	Correspond à un caractère **non alphanumérique**.
`\w`	Correspond à un caractère **alphanumérique**.
`\18f`	Correspond à la **valeur hexadécimale** 18f.
`^`	Correspond au **début d'une ligne**.

TAB. 7.1 – Règles d'interprétation des caractères contenus dans une expression rationnelle.

Le script qui suit montre un exemple concret d'emploi d'expression rationnelles pour reformater la sortie d'une commande en n'affichant que certains éléments. La syntaxe pour utiliser ce script est :

```
ls -l | taille.pl
```

Voici le script :

```
#!/usr/bin/perl

# taille.pl : exemple d'emploi d'expression rationnelle.

# taille.pl met dans le tableau @champs le resultat de la commande ls -l.

# Syntaxe : ls -l | taille.pl.

# @champs est le tableau dans lequel les elements sont ranges (la
# numerotation commence a 0).

# La fonction split divise chaque ligne en champs.

# "/\s+/" designe un ou plusieurs espaces comme separateurs.

# "$_" represente la ligne d'entree derniere (ici <STDIN>).

while (<STDIN>) {
    @champs = split(/\s+/, $_);
    $taille = $champs[4];
    $nomfichier = $champs[8];
    printf " \"%s\" a une taille de %d octets\n", $nomfichier, $taille;
}
```

Le résultat pourrait ressembler à ca :

```
"entree" a une taille de 187 octets
"essai.pl" a une taille de 869 octets
"logintime" a une taille de 1221 octets
"salut.pl" a une taille de 320 octets
"simple.perl" a une taille de 4137 octets
"sortie" a une taille de 0 octets
"taille.pl" a une taille de 634 octets
"var.perl" a une taille de 3212 octets
```

C'est quand même plus zoli, non ?

Le script qui suit montre quelque possibilités de programmation avancée avec Perl. On peut malgrès tout facilement comprendre ce qu'il fait. Il montre à quel point il peut être facile de traiter un problème avec Perl.

```
#!/usr/bin/perl

# Ce script montre quelque possibilites de programmation avancee
# avec Perl...

print "Exemple d'emploi d'expressions rationnelles...\n";
print "---------------------------------------------\n";

print "'Mathieu' marche...\n";

# $name commence par Mathieu ("^"), en maj. ou min. ("i") avec un espace
# apres Mathieu ("\b").

$name=<STDIN>;
    if ($name =~ /^Mathieu\b/i) {
        print "Bonjour, Mathieu\n";
    } else {
        print "desole...\n";
    }

# Enleve ("s") tout apres (".*") un char different d'une lettre, d'un
# chiffre ou d'un chararactere souligne ("\W").

$name=~s/\W.*//;

# Transcrit en minuscules.

$name=~tr/A-Z/a-z/;

print "$name\n";

# Pour tous les fichiers se terminant par ".secret";
while ($filename=<*.secret>) {
    open (liste, $filename);

# S'il date de moins de 7 jours... on recupere les lignes

    if (-M liste<7) {
        while ($name=<liste>) {
            chop ($name);
            print "$filename -> $name\n";
```

```
        }
    }
    close(liste);
}

# "getpwuid" contient les champs du fichier /etc/passwd.

@pass=getpwuid($<);

foreach $i (0..8) {
    print "i=$i, $pass[$i]\n";
}
```

Le script qui suit montre comment reformater de facon soignée la sortie d'une commande comme `last`, qui indique pour chaque login le nom de l'utilisateur et le temps passé. Le script `logintime` aditionne le temps passé dans chacune de ces sessions et affiche le résultat sous forme d'un tableau.

```
#!/usr/bin/perl


# Fichier logintime.
# Ce script reformate la sortie de la commande last.

# Utilisation : last | logintime.

# La sortie s'affichera sous forme de tableau avec en tete.


# Tant qu'il y a quelque chose a lire...
# cherche une ligne et sauvegarde le nom et le temps.

while (<STDIN>)
  {
  if (/^(\S*)\s*.*\((.*):(.*)\)$/)
      { $heures{$1}+= $2;
        $minutes{$1} += $3;
        $logins{$1}++;
      }
  }

foreach $user (sort(keys %heures))
```

```
{
  $heures{$user} += int($minutes{$user} / 60);
  $minutes{$user} %= 60;

# Facon simple d'afficher la sortie :

#  print "Utilisateur $user, temps total ";
#  printf "%02d:%02d, ", $heures{$user}, $minutes{$user};
#  print "total des sessions $logins{$user}.\n";

# Facon avec formatage de page :

# Conversion de la chaine numerique en chaine de caractere.

$letemps = sprintf("%02d:%02d", $heures{$user}, $minutes{$user});

# Demande d'utilisation du format de sortie definit pour afficher
# une ligne.

write;

# Definition du format du haut de la page et du format de sortie.
# Le "." marque la fin de la definition de STDOUT_TOP et STDOUT.
#
# "@<<<<<" justifie le texte a gauche et "@#####" denote une
# valeur numerique.

format STDOUT_TOP =
 Utilisateur   Temps total   Nombre de sessions
------------- ------------- --------------------
.


format STDOUT =
@<<<<<<<<<<<< @<<<<<<<<<<<< @#######
$user,        $letemps,     $logins{$user}
.
}
```

Ainsi, le résultat pourrait ressembler à ceci :

```
 Utilisateur   Temps total   Nombre de sessions
```

```
------------- ------------- --------------------
mathieu       31:36                 28
root          00:43                 19
```

Une fois de plus, de facon très simple on a réussit à effectuer une tâche qui aurait pu être compliquée à réaliser dans un autre langage.

Chapitre 8

Programmation en C

8.1 Utiliser gcc

8.1.1 Compilation à partir d'un fichier source

Voici un simple fichier source :

```
/*
 * salou.c - Source du programme salou
 *
 * Ce programme montre comment traiter les arguments d'un programme
 * (boucle "for (i = 0; i < argc; i++)") et comment utiliser des
 * fonctions definies dans le meme fichier source.
 *
 */

#include <stdio.h>
#include <string.h>

main (int argc, char **argv)
{
  int i;
  printf ("\n");

  /* Traitement des arguments passes au programme */

  if (argc > 10)
    {
      printf ("Trop d'arguments ! Essayez %s -h pour avoir l'aide\n",
```

```
                                                               argv[0]);
      printf ("\n");
      printf ("********************************\n");
      printf ("Usage : %s <arguments>\n", argv[0]);
      printf ("**********************************\n");
      exit (1);
    }

  for (i = 0; i < argc; i++)
    {
      if ((strcmp ("-h", argv[i]) == 0) || (strcmp ("--help",
                                            argv[i]) == 0))
{
  printf ("aide demandee...\n");
  printf ("y'a qu'a demander !...\n");
  printf ("et faut pas hsiter !...\n");
  printf ("  -h, --help         affiche l'aide\n");
  printf ("  -n, --name         affiche le nom de l'auteur\n");
} /* if */
      if ((strcmp ("", argv[i]) == 0))
{
  printf ("%s programme par Mathieu DECORE, 1998\n", argv[0]);
} /* if */
    } /* for */
  printf ("\n");

  printf ("*********************************************\n");
  printf ("Bonjour  celui qui lira a \n");
  printf ("**********************************************\n");

/* Exemple d'utilisation d'une fonction definie dans le programme
 * Cette fonction, "min", renvoit le minimun de deux nombres :
 * min(a, b) renvoit a si a < b, et renvoit b sinon (si b < a)
 */


  printf ("min 3, 5 : %d\n", min (3, 5));
  printf ("min 8, 5 : %d\n", min (8, 5));

/* Affichage de tous les arguments passes au programme */
```

```
  for (i = 0; i < argc; i++)
    printf ("argc %d argv[%d] %s \n", argc, i, argv[i]);

}

/* Definition d'une fonction "min" pour l'exemple */

int
min (int a, int b)
{
  if (a < b)
    return a;
  else
    return b;
}
```

Pour compiler ce programme, il suffit d'entrer :

```
gcc -o salou salou.c
```

L'exécutable sera `salou`, comme précisé juste après l'option *-o*.

8.1.2 Compilation à partir de deux fichiers sources

On peut dans un premier fichier source définire une fonction qui sera appelée par un autre fichier source. Par exemple, le fichier `cercle.c` contient simplement la définition d'une fonction utilisée par le fichier `aire.c`. Voici le fichier cercle.c :

```
/*
 * cercle.c - Exemple de fichier source en C qui, compile avec
 * un autre source (aire.c), donne l'executable aire
 *
 */

#include <math.h>

#define SQUARE(x) ((x)*(x))

/* definition de la fonction "aire_du_cercle" utilisee par aire.c */

double
aire_du_cercle (double r)
```

```
{
  return M_PI * SQUARE (r);
}
```

et voici le fichier aire.c :

```
/*
 * aire.c - Exemple de fichier source en C qui, compile avec
 * un autre source (cercle.c), donne l'executable aire
 *
 */

#include <stdio.h>
#include <stdlib.h>

/*
 * fonction "aire_du_cercle" definie dans un autre fichier source
 * (ici cercle.c).
 *
 */

double aire_du_cercle (double r);

void
main (int argc, char **argv)
{
  double aire, rayon;
  if (argc < 2)
    {
      printf ("Usage : %s rayon\n", argv[0]);
      exit (1);
    }
  else
    {
      rayon = atof (argv[1]);
      aire = aire_du_cercle (rayon);
      printf ("Aire du cercle de rayon %f = %f\n", rayon, aire);
    }
}
```

Pour compiler un tel programme, il faut d'abord compiler *chaque* source avec l'option *-c* (le fichier obtenu sera un fichier *objet*, avec l'extension *.o*), **puis** compiler les deux fichiers objets pour faire l'exécutable `aire` :

```
gcc -c aire.c
gcc -c cercle.c
gcc  aire.o cercle.o -o aire
```

Pour compiler en utilisant des bibliothèques (personnelles ou non), voir section 8.2. Pour compiler en utilisant des bibliothèques partagées, voir section 8.5.

8.2 Créer des bibliothèques C

Le but de cette section est d'inclure à un fichier `salut.c` une bibliothèque personnelle `libmat.a` contenant des fonctions. Voici le fichier `salut.c` :

```
/*
 *
 * salut.c - Exemple d'utilisation de bibliotheques personneles.
 * La bibliotheque utilisee ici est "libmat.h" et les fonctions
 * appelees sont "moi" et "mathieu". De plus, une fonction est definie
 * dans ce fichier source ("min").
 *
 */

#include <stdio.h>
#include <string.h>
#include <libmat.h>
#include <stdlib.h>

main (int argc, char **argv)
{
  int i;
  double rayon, aire;
  printf ("\n");

  printf ("*******************************************\n");
  printf ("bonjour  celui qui lira a\n");
  printf ("********************************************\n");

  /* Appel des fonctions definies a la fin de "salut.c" */
```

```
  printf ("min 3, 5 : %d\n", min (3, 5));
  printf ("min 8, 5 : %d\n", min (8, 5));

  /* Appel des fonctions definies dans "libmat.h" */

  moi (1, 1);
  mathieu (22);

  /* Traitement des arguments passes au programme. */

  for (i = 0; i < argc; i++)
    printf ("argc %d argv[%d] %s \n", argc, i, argv[i]);

  for (i = 0; i < argc; i++)
    {
      if ((strcmp ("-h", argv[i]) == 0) || (strcmp ("--help",
argv[i]) == 0))
{
  printf ("aide demandee...\n");

  printf ("  -h, --help         affiche l'aide\n");
  printf ("  -t                 affiche un message\n");
}
      if ((strcmp ("-t", argv[i]) == 0))
{
  printf ("Message...\n");
}
    }
  printf ("\n"); /* for i */
}

/* Definition d'une fonction juste pour tester */

int
min (int a, int b)
{
  if (a < b)
    return a;
  else
    return b;
```

```
}
```

Ce qui suit traite des bibliothèques **statiques**, qui sont directement incluses dans le fichier binaire (exécutable) une fois la compilation faite. Ces bibliothèques ont une extension *.a* et se trouvent généralement dans le répertoire `/usr/lib/`. Pour la programmation en C, les plus utiles sont `libc.a` (bibliothèque C standard) et `libm.a` (bibliothèque mathématique).

Soient deux sources `moi.c` :

```
int moi(int nom, int prenom) {
printf ("J'ai %d prenom et %d nom.\n", prenom, nom);
return -1;
}
```

et `mathieu.c` :

```
int mathieu(int age) {
printf("Mathieu a %d ans...\n", age);
}
```

contenant des fonctions. Pour utiliser ces bibliothèques, il faut :

1. Compiler chaque source :

   ```
   gcc -c moi.c mathieu.c
   ```

2. Créer la bibliothèque `libmat.a` :

   ```
   ar r libmat.a moi.o mathieu.o
   ```

3. Génerer l'index :

   ```
   ranlib libmat.a
   ```

4. Ecrire l'en-tête dans le fichier `libmat.h` :

   ```
   extern int mathieu(int );
   extern int moi(int , int );
   ```

5. Placer la bibliothèque (`libmat.a`) dans le sous-répertoire `lib/` et l'en tête (`libmat.h`) dans le sous répertoire `include/`
6. Pour compiler un fichier source appellant cette bibliothèque, inclure les répertoires précedents, en demandant à l'éditeur de liens d'utiliser `libmat.a` par l'argument *-lmat* :

   ```
   gcc -I include -L lib -O salut.c -o salut -lmat
   ```

Le préfixe *lib* et le suffixe *.a* sont sous-entendus.

8.2.1 Compiler à partir de deux fichiers sources en utilisant des bibliothèques personnelles

La synthèse des section précédentes est contenue dans le fichier `salaire.c`, acronyme de **sal**ut et de **aire**, les programmes vus avant :

```
/*
 * salut.c - Exemple d'utilisation de bibliotheques personneles.
 * La bibliotheque utilisee ici est "libmat.h" et les fonctions
 * appelees sont "moi" et "mathieu". De plus, une fonction est definie
 * dans ce fichier source ("min"). Enfin, ce fichier source en C
 * appelle une fonctiondefinie dans un autre fichier source, "cercle.c".
 * Le tout donne l'executable salaire (SALut+AIRE).
 *
 */

#include <stdio.h>
#include <string.h>
#include <libmat.h>
#include <stdlib.h>

/* fonction "aire_du_cercle" definie dans un autre fichier source
 * (ici cercle.c). */

double aire_du_cercle (double r);

main (int argc, char **argv)
{
  int i;
  double rayon, aire;
  printf ("\n");
  if (argc < 2)
    {
      printf ("Syntaxe incorrecte, essayez %s -h pour avoir l'aide\n",
                                                         argv[0]);
      printf ("\n");
      printf ("**********************************\n");
      printf ("Usage : %s rayon\n", argv[0]);
      printf ("***********************************\n");
      exit (1);
    }
```

```
    printf ("********************************************\n");
    printf ("bonjour  celui qui lira a\n");
    printf ("*********************************************\n");

    /* Appel des fonctions definies a la fin de "salut.c" */

    printf ("min 3, 5 : %d\n", min (3, 5));
    printf ("min 8, 5 : %d\n", min (8, 5));

    /* Appel des fonctions definies dans "libmat.h" */

    moi (1, 1);
    mathieu (22);

    /* Appel de la fonction definie dans le fichier  "cercle.c" */

    rayon = atof (argv[1]);

    aire = aire_du_cercle (rayon);
    printf ("Aire du cercle de rayon %f = %f\n", rayon, aire);

    /* Traitement des arguments passes au programme. */

    for (i = 0; i < argc; i++)
      printf ("argc %d argv[%d] %s \n", argc, i, argv[i]);

    for (i = 0; i < argc; i++)
      {
        if ((strcmp ("-h", argv[i]) == 0) || (strcmp ("--help",
                                                    argv[i]) == 0))
  {
    printf ("aide demandee...\n");

    printf ("  -h, --help         affiche l'aide\n");
    printf ("  -t                 affiche un message\n");
  }
        if ((strcmp ("-t", argv[i]) == 0))
  {
    printf ("Message...\n");
  }
```

```
    }
  printf ("\n"); /* for i */
}

/* Definition d'une fonction juste pour tester */

int
min (int a, int b)
{
  if (a < b)
    return a;
  else
    return b;
}
```

Pour compiler un tel programme, il faut entrer les commandes suivantes :

```
gcc -c -I ../include salaire.c
gcc -c cercle.c
gcc  -L ../lib salaire.o cercle.o -o salaire -lmat
```

C'est un peu compliqué, et surtout long à taper, d'autant qu'on peut compiler des fichiers qui n'ont pas été modifiés (si seul le fichier `cercle.c` est modifié, ce n'est pas la peine de re-compiler le fichier `salaire.c`). Les **makefiles** vont nous aider pour nous simplifier la vie.

8.3 La programmation en C++

Le compilateur C++ s'appelle `g++`. Voici un exemple de fichier écrit en C++ (le fameux *salut*) :

```
#include <iostream.h>

void
main (void)
{
  cout << "Salut en C++ !" << endl;
}
```

Pour compiler un tel fichier, taper (la syntaxe est semblable à celle utilisée par `gcc`) :

```
g++ salut++.C -o salut++
```

L'exemple qui suit montre un exemple un peu plus élaboré de programme écrit en C++. Ce fichier, `str.C`, montre l'implémentation d'une classe pour faciliter les manipulations de chaînes de caractères (*strings*). Un début de définition de la classe string est ici présenté.

```
#include <iostream.h>

class string
{
  char *p;
  int longueur;
public:
  string(int l) { p = new char[longueur = l]; }
  ~string(void) { delete[] p; }
  string& operator=(string&);
};
string& string::operator=(string& s)
{
  p = s.p;
  longueur = s.longueur;
}

void main(void)
{
  string s1(10);
  string s2(20);
  cout << "Tout s'est bien passe !" << endl;
  s1 = s2;
}
```

8.4 Les makefiles

Les *makefiles* permettent de compiler des programmes ou plus généralement de lancer de facon simple une commande ou une suite de commandes. Dans un simple fichier ASCII, on donne les fichiers à construire, à partir desquels il faut les construires, les opérations qu'il faut exécuter...

8.4.1 Compilation simple d'un fichier C

Voici un makefile simple (c'est un fichier qui peut s'appeller `makefile` ou `Makefile`, auquel cas il apparaîtra en tête de l'arborescence) :

```
CC = gcc
CFLAGS = -ansi
LDFLAGS = -lm -lc  # bibliotheque math suivie de bibliotheque C

all : salou
```

Si on se contente de spécifier quel compilateur C il faut utiliser, et avec quelles options, la cible (le fichier `salou` qu'on cherche à construire) sera faite en compilant le fichier `salou.c` (évident puisqu'on compile avec le compilateur C, *CC*). Si le fichier `salou` existe déjà et qu'aucune modification n'a été apportée au fichier source, la compilation n'aura pas lieu. On aurait aussi pu mettre :

```
salou : salou.c
gcc -ansi -lm -lc -o salou salou.c
```

La ligne de dépendances `salou : salou.c` indique clairement que le fichier `salou` a besoin du fichier `salou.c` pour être construit et que si celui-ci n'a pas été modifié, la compilation du fichier `salou` n'aura pas lieu. Si le fichier `salou.c` n'existe pas mais qu'il y a dans la suite du makefile une ligne indiquant comment le construire, le fichier `salou.c` sera construit *puis* le fichier `salou` sera à son tour construit. Noter qu'il faut une tabulation à la suite de la ligne de dépendances (la ligne `gcc -ansi -lm -lc -o salou salou.c` dans l'exemple précédent).

Voici un exemple plus explicite :

```
# Exemple simple d'utilisation d'un Makefile

#
# Par defaut, le Makefile sait comment construire un fichier objet (*.o)
# a partir d'un fichier source (*.c) :
#
# aire.o : Makefile aire.c
#
# suffit a lancer la compilation a partir du fichier source (ici aire.c)
#

#
# La commande CC sera invoquee par defaut, c'est l'ancien compilateur C;
# c'est sur la plupart des systemes un lien vers le nouveau compilateur
# GNU gcc :
#
# aire : $(OBJS)
```

```
#
# suffit a lancer la compilation a partir des fichiers objets
#

#
# ANSI_FLAG peut etre definit lorsque on lance la commande make :
#
# make ANSI_FLAG=Ok
#

#
# ANSI_FLAG peut aussi etre definit dans le fichier .bashrc :
#
# export ANSI_FLAG = Ok
#

#
# ATTENTION ! ANSI_FLAG = non est aussi une definition, si bien que le
# programme sera egalement compile avec la norme ansi...
#

ANSI_FLAG =
# aucune definition, accepte

CFLAGS=-g -O
# options de compilation

OBJS = aire.o cercle.o
# fichiers objets


PROG = ../../
# chemin du repertoire prog

BIN = $(PROG)bin/
# chemin du repertoire bin

ifdef ANSI_FLAG
CFLAGS := $(CFLAGS) -ansi # faut il compiler suivant la norme ansi ?
endif
```

```
install : all
mv aire $(BIN)

all : aire

# Construire le fichier binaire (compile, donc executable) aire avec CC
# a partir des fichiers objets definits dans la macro OBJS
# (ici aire.o et cercle.o)

# Dependances de aire : aire.o et cercle.o

aire : $(OBJS)

# Construire le fichier objet avec CC et CFLAGS

aire.o : Makefile aire.c

cercle.o : Makefile cercle.c

# Nettoyage des fichiers objets (si on considere par exemple que le fichier
binaire n'a dorenavent plus besoin d'etre recompile)

clean :
rm -f `find . -name '*.o' -print`
```

8.4.2 La règle des modèles

Voici un exemple simple utilisant la règle des modèles :

```
# Exemple d'utilisation de regles implicites : les regles de modele

# Ces regles sont plus souples que les regles de suffixe

# '%' signifie "n'importe quelle chaine"

# '%.o : %.c' signifie : prendre un fichier se terminant par .c pour
construire un fichier se terminant par .o

# $@ represente le fichier de sortie

# $< represente le fichier d'entree
```

```
# Syntaxe pour lancer ce makefile : make aire.o

# On peut passer des arguments : make CFLAGS="-O -g" aire.o

OBJS = cercle.o aire.o

aire : $(OBJS)
$(CC) -o $@ $(OBJS)

%.o : %.c
$(CC) $(CFLAGS) $(CPPFLAGS) -c -o $@ $<
```

Et voici un exemple un peu plus compliqué (ce makefile permet de compiler deux versions du programme `aire` au choix : `aire`, le programme normal, et `aire_dbg` avec l'option *-g* pour déboguage) :

```
# Exemple d'utilisation de regles implicites : les regles de modele

# Ces regles sont plus souples que les regles de suffixe

# '%' signifie "n'importe quelle chaine"

# '%.o : %.c' signifie : prendre un fichier se terminant par .c pour
construire un fichier se terminant par .o

# $@ represente le fichier de sortie

# $< represente le fichier d'entree

# Syntaxe pour lancer ce makefile : make aire.o

# On peut passer des arguments : make CFLAGS="-O -g" aire.o

OBJS = cercle.o aire.o

DEBUG_OBJS = cercle_dbg.o aire_dbg.o

aire : $(OBJS)
$(CC) -o $@ $(OBJS)

aire_dbg : $(DEBUG_OBJS)
```

```
$(CC) -o $@ $(DEBUG_OBJS)

%.o : %.c
$(CC) $(CFLAGS) $(CPPFLAGS) -c -o $@ $<

%_dbg.o : %.c
$(CC) -c -g -O -o $@ $<

clean :
rm -f `find . -name '*.o' -print`
```

8.4.3 La règle des suffixes

Voici un exemple simple utilisant la règle des suffixes :

```
# Exemple d'utilisation de regles implicites : les regles de suffixe

# '.c .o' signifie : prendre un fichier se terminant par .c pour construire
un fichier se terminant par .o

# $(CC) est connu, il correspond a la commande cc qui est un lien vers gcc

# $@ represente le fichier de sortie

# $< represente le fichier d'entree

# Syntaxe pour lancer ce makefile : make aire.o

# On peut passer des arguments : make CFLAGS="-O -g" aire.o

OBJS = cercle.o aire.o

aire : $(OBJS)
$(CC) -o $@ $(OBJS)

.c .o :
$(CC) $(CFLAGS) $(CPPFLAGS) -c -o $@ $<
```

8.4.4 Utiliser plusieurs makefiles

Un makefile peut en appeller un autre, et même utiliser les variables définies dans le makefile, à condition que ces variables aient été exportées.

Dans l'exemple suivant, le makefile appelle le fichier `cache` :

```
all :
export HOST_NAME=$(shell uname -n); \
mkdir $$HOST_NAME;

INC_FILE = cache

include $(INC_FILE)
```

et le fichier cache contient :

```
cache_rm :
@rmdir $$HOST_NAME
```

8.5 Utilisation de bibliothèques partagées

Les bibliothèques partagées ne sont chargées que lorsqu'un programme les appellent, ce qui permet de gagner de la place en mémoire. Pour savoir à quelles bibliothèques partagées est lié un programme, utiliser la commande `ldd`. Par exemple pour le programme `/usr/bin/X11/xterm` :

```
ldd /usr/bin/X11/xterm
```

Le résultat devrait ressembler à ca :

```
libXaw.so.6 => /usr/X11R6/lib/libXaw.so.6 (0x4000b000)
libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0x40042000)
libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0x40054000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x40096000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x4009f000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x400b4000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x400bf000)
libc.so.5 => /lib/libc.so.5 (0x4015e000)
```

La ligne `libc.so.5 => /lib/libc.so.5 (0x4015e000)` par exemple indique la bibliothèque C standard de version 5.

L'éditeur de liens dynamiques `ld.so` recherchera les bibliothèques dans les répertoires `/lib`, `/usr/lib`, et dans les répertoires spécifiés dans le fichier `/etc/ld.so.conf` et dans la variable d'environement `$LD_LIBRARY_PATH` (utile pour ses propres bibliothèques confidentielles, par exemple). Il faut par ailleurs invoquer la commande `ldconfig` après chaque modification du fichier `/etc/ld.so.conf`.

Les bibliothèques partagées ont une extension *.so* et se trouvent en général dans le répertoire `/lib`. Le système recherche toujours sur le numéro de version majeur les noms de bibliothèques : `/lib/libc.so.5.4.33` a un numéro majeur de version 5 et un numéro mineur de version 4, par exemple. La bibliothèque recherchée sera `/lib/libc.so.5`, qui est un lien symbolique vers `/lib/libc.so.5.4.33` comme l'indique la commande `ls -l` :

```
lrwxrwxrwx 1 root root 13 Nov 3 17:58 /lib/libc.so.5 ->
libc.so.5.4.33*
-rwxr-xr-x 1 root root 634880 Apr 29 1996 /lib/libc.so.5.4.33*
```

8.5.1 Mettre à jour les bibliothèques

Pour mettre à jour les bibliothèques statiques, il suffit de copier la nouvelle bibliothèque dans le répertoire `/usr/lib`. L'ancienne sera écrasée, et remplacée par la nouvelle. Pour mettre à jour les bibliothèques dynamique, c'est plus compliqué car elles doivent en permanence être accessibles. Il faut donc d'abord copier la nouvelle bibliothèque dans le répertoire `/lib` (`/lib/libc.so.5.5` par exemple) *puis* modifier le lien symbolique en une seule étape :

```
ln -sf /lib/libc.so.5.5 /lib/libc.so.5
```

La nouvelle bibliothèque sera `/lib/libc.so.5.5`, et l'ancienne peut maintenant être enlevée en toute sécurité non sans avoir vérifie que le nouveau lien pointe bien vers la nouvelle bibliothèque gâce à la commande `ls -l` :

```
lrwxrwxrwx 1 root root 13 Nov 3 17:58 /lib/libc.so.5 ->
libc.so.5.5*
-rwxr-xr-x 1 root root 634880 Apr 29 1996 /lib/libc.so.5.4.33*
-rwxr-xr-x 1 root root 684680 Sep 14 1996 /lib/libc.so.5.5*
```

Une fois de plus, il faut lancer la commande `ldconfig` pour tout mettre à jour.

8.5.2 Création d'une bibliothèque partagée

Voici un objet simple, mis en place en tant que bibliothèque partagée :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct OBJDATA
```

```
{
char *name;
int version;
} OBJDATA;

void *init(char *name)
{
OBJDATA *data=(OBJDATA*)calloc(1,sizeof(OBJDATA));
if(name)
data->name=malloc(strlen(name)+1);
strcpy(data->name, name);
printf("Cree : %s\n", name);
return data;
}

void montre(void *data)
{
OBJDATA *d=(OBJDATA*)data;
printf("Montre : %s\n", d->name);
}

void detruit(void *data)
{
OBJDATA *d=(OBJDATA*)data;
if(d)
  {
if(d->name)
  {printf("Destruction : %s\n", d->name);
free(d->name);
}
free(d);
}
}
```

L'objet offre trois fonctions d'interface :

1. **init**, pour attribuer toute la mémoire nécessaire (stockage, rangement) et pour initialiser l'objet.
2. **show**, pour afficher l'objet (ici, cela imprime simplement un message).
3. **destroy**, pour libérer de la mémoire.

Pour constituer la bibliothèque partagée `libdobj.so`, procéder comme suit :

```
gcc -fPIC -c obj.c
gcc -shared -Wl,-soname,libdobj.so.1 -o libdobj.so.1.0 obj.o
ln -s libdobj.so.1.0 libdobj.so.1
ln -s libdobj.so.1 libdobj.so
export LD_LIBRARY_PATH=`pwd`:$LD_LIBRARY_PATH
```

Une fois testée, la bibliothèque peut être intallée dans un endroit standard, comme le répertoire `/usr/local/lib/`, après quoi il faut lancer `ldconfig` pour mettre à jour le lien entre `libdobj.so.1` et `libdobj.so.1.0` :

```
ln -s libdobj.so.1 libdobj.so
```

Il faut maintenant un programme qui teste cette nouvelle bibliothèque, qui appelle cette bibliothèque par un `#include <dlfcn.h>` :

```
#include <dlfcn.h>
#include <stdio.h>

void main(void)
{
void *dlobj;
void * (*init_appel)(char *name);
void (*show_appel)(void *data);
void (*destroy_appel)(char *data);

if(dlobj=dlopen("libdobj.so.1", RTLD_LAZY))
  {
void *data;

init_appel=dlsym(dlobj,"init");
show_appel=dlsym(dlobj,"montre");
destroy_appel=dlsym(dlobj,"detruit");

data=(*init_appel)("Test Objets");
(*show_appel)(data);
(*destroy_appel)(data);
  }
}
```

Il faut ensuite compiler ce programme :

```
gcc -o dltest dltest.c -ldl
```

Il ne reste plus qu'à tester le programme :

```
dltest
```

Le résultat s'affiche alors :

```
Cree : Test Objets
Montre : Test Objets
Destruction : Test Objets
```

8.6 RCS, contrôle de versions de code source

8.6.1 Pour débuter

- D'abord, enregistrer le fichier `salut.c` dans RCS :

  ```
  ci salut.c
  ```

  Taper une description du programme. Le fichier `salut.c` est alors enregistré sous forme `salut.c,v`, et mis automatiquement dans le sous-répertoire RCS, s'il a été crée.
- Ensuite, extraire le fichier du système (*check out*) avec vérouillage (option `-l`) pour qu'il puisse être modifié par une seule personne à la fois (sinon, on ne pourrait que lire le fichier, pas le modifier) :

  ```
  co -l salut.c
  ```

- faire les modifications nécessaires, compiler...
- rendre le fichier au système et rentrer une description des modifications :

  ```
  ci -l salut.c
  ```

- pour extraire la version 1.3, entrer :

  ```
  co -l1.3 salut.c
  ```

8.6.2 Mots cléfs dans le fichier source

On peut mettre dans le fichier source des mots cléfs qui donnent des renseignements sur le fichier. Ceux-ci doivent être en commentaires :
- $Author$
- $Date$
- $Header$
- Id
- $Locker$
- Log
- $RCSfile$
- $Revision$
- $Source$

– $State$

Ces informations sont également accessibles par la commande `ident` :

```
ident salut.c
```

8.6.3 Autres commandes utiles

- `rlog` : affiche l'historique des changements d'un fichier.
- `rcsdiff` : compare le fichier avec la version RCS enregistrée. On peut également comparer deux versions 1.1 et 1.2 par la commande :

  ```
  rcsdiff -r1.1 -r1.2 salut.c
  ```

- `merge` : incorpore les changements de deux fichiers dans un troisième.
- `rcsmerge` : fusionne différentes révisions d'un fichier.
- et bien sur `ci`, `co` et `ident`.

8.7 Outils pour la programmation en C

8.7.1 Déboguage avec gdb

Le programme `gdb` permet d'aider le programmeur à trouver les bogues dans un programme C.

Voici un exemple d'utilisation de `gdb` : un programme nommé `bogue` qui plante lorsqu'il lit une entrée au clavier. Voici le code source du programme `bogue` :

```
#include <stdio.h>

static char buf[256];
void lit_entree (char *s);

main ()
{
  char *entree = NULL;

  lit_entree (entree);
  printf ("Vous avez tape : %s\n", entree);
}

void
lit_entree (char *s)
{
```

```
    printf ("Commande : ");
    gets (s);
  }
```

Quand on lance ce programme, il plante si on entre une chaîne au clavier. Par exemple si on tape “test” :

```
Commande : test
Segmentation fault
```

Le débogueur `gdb` va nous aider. Il faut d'abord compiler le programme avec l'option *-g* pour le déboguage, et **sans optimisations** (utiliser éventuellement *-O0*, pour supprimer toute optimisations) :

```
gcc -g -o bogue bogue.c
```

Maintenant, on peut lancer `gdb` :

```
gdb bogue
```

On peut lancer le programme par la commandes `run` :

```
Commande : test

Program received signal SIGSEGV, Segmentation fault.
0x4002fbe8 in _IO_gets (buf=0x0)
(gdb)
```

La commande `where` permet de déterminer où le programme s'est arrêté :

```
#0  0x4002fbe8 in _IO_gets (buf=0x0)
#1  0x10e1 in lit_entree (s=0x0) at bogue.c:17
#2  0x10a3 in main () at bogue.c:11
(gdb)
```

La commande `list` permet d'inspecter les lignes de code suspectes. Par exemple, pour inspecter autour de la ligne 17 :

```
list bogue.c:17
```

```
12      }
13
14      void lit_entree(char *s)
15      {
16      printf("Commande : ");
17      gets(s);
18      }
```

Il semble que l'appel de *lit_entree* pose problème. Pour s'en assurer, on peut regarder où l'appel a été émis (ligne 11) :

```
list bogue.c:11
```

```
6
7       main ()
8       {
9         char *entree = NULL;
10
11        lit_entree (entree);
12        printf ("Vous avez tape : %s\n", entree);
13      }
14
15      void
(gdb)
```

C'est clair, le problème vient de cette fichue ligne 11, puisque le pointeur *entree* n'est pas initialisé. On peut corriger le programme en affectant le pointeur *entree* à *buf* qui lui est initialisé. Pour cela, taper :

```
file bogue
```

Il apparaît alors :

```
A program is being debugged already.  Kill it? (y or n) y

Load new symbol table from "bogue"? (y or n) y
Reading symbols from bogue...done.
(gdb)
```

On peut regarder les premières lignes du programme, là où sont déclarées les variables, par la commande `list` :

```
Source file is more recent than executable.
1       #include <stdio.h>
2
3       static char buf[256];
4       void lit_entree (char *s);
5
6       main ()
7       {
8         char *entree = NULL;
9
10        lit_entree (entree);
(gdb)
```

On met un **point d'arrêt** à la ligne 10, juste avant l'affectation douteuse. La programme s'arrêtera à ce stade pour toutes les opérations qui suivront :

```
break 10
```

On voit alors s'afficher sur l'écran :

```
Breakpoint 1 at 0x80484bd: file bogue.c, line 10.
(gdb)
```

On peut donc lancer le programme par la commande `run`, qui s'exécute jusqu'à la ligne 10 :

```
Starting program: /home/mathieu/prog/C/gdb/bogue

Breakpoint 1, main () at bogue.c:10
10        lit_entree (entree);
```

Et on corrige par la commande `set` :

```
set var entree=buf
```

La commande `cont` permet de continuer l'exécution du programme, puisque maintenant le pointeur *entree* est initialisé :

```
Continuing.
Commande : test
Vous avez tape : test

Program exited with code 026.
(gdb)
```

Le programme s'est terminé normalement. Il ne reste plus qu'à apporter les modifications nécessaires au fichier source, et à recompiler (sans l'option *-g*, puisque tout marche dorénavant).

On peut par ailleurs utiliser bien d'autre commandes sous `gdb`, comme `next` pour exécuter la ligne de code suivante (après un point d'arrêt, par exemple) *dans la même fonction* (il faut bien-sûr que le programme tourne, c'est-à-dire que `run` ait été lancé), ou `step` pour exécuter la ligne de code suivante et éventuellement *les appels de fonctions rencontrés*. On peut aussi exécuter le programme jusqu'à une ligne donnée, par exemple `until 24` exécute le programme jusqu'à la ligne 24. La commande `print` permet de connaître la valeur d'une variable : `print a` donne :

```
$1 = 8
```

et `print b` donne :

```
$2 = 5
```

dans le petit programme `salou` évoqué précédemment.

Pour sortir d'une fonction et retourner dans la fonction courante, utiliser la commande `finish`. La commande `quit` permet de quitter `gdb`.

Pour déboguer un programme en cours de fonctionnement, il faut lancer `gdb` puis le lier au processus actif, par la commande `attach` suivie du numéro de processus (PID, voir section 1.9 page 20). Par exemple :

```
(gdb) attach 254
```

attache le programme de PID 254 (il faut avoir chargé le fichier source correspondant en appelant `gdb`). On peut aussi lier le programme lorsqu'on appelle `gdb` :

```
gdb salou 254
```

Si on apporte des modifications au fichier source et qu'on le recompile, il faut avant détacher le processus avec la commande `detach`, effectuer les modifications, recompiler, et employer la commande `file` pour recharger le nouvel exécutable dans le débogueur. Il ne reste plus qu'à attacher la nouvelle version du programme par la commande `attach`.

Pour examiner les valeurs des variables du programme, la commande `print` est la plus souvent utilisée. On peut savoir la valeur d'une variable, d'une fonction, d'un élément de tableau...On peut aussi se contenter de fournir le numéro *interne* à `gdb`, lors d'un précédent `print` (comme \$2 pour appeler b, dans l'exemple précédent). On peut même affecter une valeur :

```
(gdb) print b=7
$3 = 7
(gdb)
```

Pour avoir une définition plus complète d'une variable, utiliser la commande `ptype` suivie de l'adresse ou de la variable :

```
(gdb) ptype b
type = int
(gdb) ptype min
type = int (int, int)
(gdb)
```

La commande `x` permet d'examiner la mémoire à un niveau plus bas (dans ce cas, c'est à l'adresse de la variable que `x` ira) :

```
(gdb) x argv[0]
0xbffff4f1 <__ypbindlist+2146674173>:   0x6d6f682f
(gdb)
```

On peut spécifier le type d'affichage :

```
(gdb) x/5x argv[0]
0xbffff4f1 <__ypbindlist+2146674173>: 0x6d6f682f 0x616d2f65 0x65696874
0x72702f75
0xbffff501 <__ypbindlist+2146674189>: 0x432f676f
(gdb)
```

affiche 100 octets de données. On peut spécifier d'autre formats d'affichage (taper `help x` pour plus d'informations) :

```
(gdb) x/6c argv[0]
0xbffff4f1 <__ypbindlist+2146674173>: 47 '/' 104 'h' 111 'o' 109 'm' 101
'e'47 '/'
(gdb) x/s argv[0]
0xbffff4f1 <__ypbindlist+2146674173>: "/home/mathieu/prog/C/salou_gdb"
(gdb)
```

La commande `info` permet d'avoir des informations sur le programme en cours. Par exemple, `info program` affiche le statut d'exécution :

```
(gdb) info program
        Using the running image of child process 242.
Program stopped at 0x8048679.
It stopped at a breakpoint that has since been deleted.
(gdb)
```

La commande `info locals` donne le nom et les valeurs de toutes les variables locales de la fonction courante :

```
(gdb) info locals
i = 1
(gdb)
```

De même, `info variables` affiche la liste de toutes les variables connues dans le programme, y compris les variables manipulées dans les bibliothèques système (seules les valeurs des variables locales et globales sont accessibles).

Pour savoir où la variables est stockée, taper `info address` :

```
(gdb) info address i
Symbol "i" is a local variable at frame offset -4.
(gdb)
```

Ici i est stockée à 4 octets du haut de la structure de pile courante (“*frame offset -4*”).

Pour obtenir des informations sur la structure de pile courante, taper `info frame` :

```
(gdb) info frame
Stack level 0, frame at 0xbffff388:
 eip = 0x8048679 in main (salou.c:58); saved eip 0x804845e
 called by frame at 0xbffff39c
 source language c.
 Arglist at 0xbffff388, args: argc=1, argv=0xbffff3a8
 Locals at 0xbffff388, Previous frame's sp is 0x0
 Saved registers:
  ebp at 0xbffff388, eip at 0xbffff38c
(gdb)
```

A propos de *break* et *watch* :

La commande `break` permet d'arrêter le programme à un **endroit particulier**, comme une ligne (`break 20`), une ligne dans un autre fichier (`break cercle.c:8`), une fonction (`break aire_du_cercle`). Les point d'arrêt peuvent être conditionnels :

```
break aire_du_cercle if (r == 0)
```

La commande `condition` permet de modifier la condition d'arrêt. La commande `info break` permet d'obtenir des informations sur tous les points d'arrêt :

```
(gdb) info break
Num Type           Disp Enb Address    What
1   breakpoint     keep y   0x08048654 in main at salou.c:56
        breakpoint already hit 1 time
(gdb)
```

Les commandes `disable` et `enable` permettent de désactiver et d'activer un point d'arrêt. Les commandes `clear` et `delete` permettent de supprimer un point d'arrêt.

La commande `watch` permet de placer des points d'observation. La différence avec `break` est qu'ils peuvent être déclenchés lorsqu'une expression est vrai **quelque soit l'endroit du programme** :

```
watch (i < 2 && argv[i] == ``-h'')
```

L'expression obéit aux mêmes règles que celles des points d'arrêt conditionnels.

8.7.2 Utiliser gdb sous Emacs

Emacs offre un mode `gdb` qui permet d'offrir une interface graphique à `gdb`.

Pour lancer `gdb` sous Emacs, taper `M-x gdb`. Donner le nom du programme à déboguer. Pour charger un fichier *core* taper `core-file`, ou `attach` pour attacher le débogueur à un processus en cours.

Le tableau 8.1 présente plusieurs raccourcis des commandes les plus courantes. Taper `M-p` pour remonter et `M-n` pour avancer dans l'historique.

Raccourcis	**Signification**
`C-x C-a C-s`	Equivalent de la commande `step`.
`C-x C-a C-n`	Equivalent de la commande `next`.
`C-x C-a C-r`	Equivalent de la commande `continue`.
`C-x C-a <`	Equivalent de la commande `up`.
`C-x C-a >`	Equivalent de la commande `down`.

TAB. 8.1 – Principaux raccourcis sous Emacs pour **gdb**.

8.7.3 Examen d'un fichier core

Un fichier *core* est crée lors d'une faute grave, comme une violation d'espace mémoire. Avec `gdb`, on peut savoir la valeur des variables et des données à l'instant précis de l'erreur.

Pour valider la création des fichiers *core*, entrer dans le fichier `.bashrc` la ligne :

```
ulimit -c unlimited
```

Pour utiliser ce fichier *core*, il faut avoir compilé le programme avec l'option de déboguage, puis lancé `gdb` en indiquant le nom du programme suivit du nom du fichier *core* :

```
gdb bogue core
```

La commande `backtrace` affiche la *pile d'appels* au moment où le problème est survenu. La première fonction affichée (#0) est aussi la dernière appelée au moment de l'erreur (celles qui suivent ont été appelées dans l'ordre chronologiquement). On peut donc savoir dans quelle fonction se situe l'erreur. La commande `frame` permet d'accéder à l'appel correspondant. Par exemple :

```
frame 2
```

accède à la fonction correspondant au numéro 2 (#2). Si le problème se situe en amont (dans une fonction précédente), utiliser la commande `up` pour remonter la pile d'appels.

8.8 Outils de développement

Outre `gdb`, il existe plusieur débogueurs présentant des interfaces graphiques, comme `xxgdb, xdbx` ou `ddd`.

- `gprof` : on peut analyser, suivre et mesurer les performances d'un programme grâce à `gprof`. Il faut d'abord compiler le programme avec l'option *-pg*, puis lancer le programme. Un fichier nommé `gmon.out` est alors crée, qui sera utilisé par `gprof`. Il suffit ensuite d'appeler `gprof` :

  ```
  gprof linatd2 | less
  ```

  La sortie étant assez longue, il faut mieux la rediriger. On peut lire le temps passé par chaque fonction, en pourcentage et en secondes, pour la fonction elle même et pour les autre fonctions appelées, le nombre d'appels...La seconde partie donne un "graphique d'appels" décrivant qui appelle quoi, et combien de fois.
- `strace` : le programme `strace` permet d'afficher les appels système exécutés lors du fonctionnement d'une application. On constate qu'un programme commence par charger les bibliothèques partagées (`open`...`close`) puis récupére les caractéristiques du terminal avant de tenter d'y expédier des données (`ioctl`). Enfin, le message affiché à l'écran est écrit (`write`).
- `checker` : pour les violations d'espace mémoire, l'option *-lchecker* peut être utile lors de la compilation. Par exemple, le programme :

  ```
  #include <malloc.h>

  int
  main ()
  {
    char *lamemoire, ch;

    lamemoire = (char *) malloc (10 * sizeof (char));

    ch = lamemoire[1]; /* lecture de memoire non initialisee */
    lamemoire[12] = ' '; /* ecriture apres un bloc non alloue */
    ch = lamemoire[-2]; /* lecture avant le bloc alloue */
  }
  ```

tente quelque opérations douteuses. Avec Checker, lors de l'exécution, des messages d'erreur s'afficheront. Une fois ces erreurs corrigées, on pourra compiler le programme normalement.

8.9 Patcher des fichiers

Soient deux versions d'un même programme : `salut.c.old` et `salut.c`. Pour créer un patch de la première version à la seconde, entrer la commande :

```
diff -c salut.c.old salut.c > salut.patch
```

Pour appliquer le patch, entrer la commande suivante :

```
patch < salut.patch
```

L'ancienne version est alors sauvegardée avec le même nom, mais avec l'extension *.orig*. Si on applique le patch une seconde fois, les modifications sont annulées.

Pour patcher deux répertoires, entrer la commande :

```
diff -cr salut.c.old salut.c > salut.patch
```

Et pour appliquer le patch :

```
patch -p0 < salut.patch
```

8.10 Indentation du code C

La commande `indent` indente un fichier source C :

```
indent salut.c
```

Plusieurs options sont disponibles, consulter la page de manuel. L'une d'elle permet de mettre en valeur le document grâce à `groff` :

```
indent -troff salut.c | groff -mindent
```

Chapitre 9

Administration système

L'administration système se fait sur le compte **root** (super-utilisateur). Ce compte donne accès à tous les droits de lecture, d'écriture et d'exécution des fichiers ou des commandes. C'est donc **dangereux** car une erreur est vite arrivée. Ce compte n'est à utiliser qu'en cas de nécessité absolue (celles décrites dans cette section entre autres) et chaque action doit être **réfléchie**.

Souvent un fichier évoqué n'aura pas le même nom sur le système. Cela dépend des distributions, mais on peut facilement retrouver ce fichier grâce à la commande `locate`. Par exemple pour le fichier de configuration de LILO (chargeur de Linux), il existe divers noms comme `/etc/lilo.conf` ou `/etc/lilo/config` ou encore `/boot/lilo.conf`. Pour trouver un tel fichier, on peut par exemple entrer la commande suivante :

```
locate lilo | grep conf
```

Le résultat montre que ce fichier existe et se trouve à l'emplacement habituel sur le système en question :

```
/etc/lilo.conf
```

Pour déterminer qu'il s'agit bien de ce fichier, il a fallut "éliminer" les autres fichiers qui visiblement font partie du manuel (lignes contenant la chaîne de charactères **man**) ou de la documentation du système (lignes contenant la chaîne de charactères **doc**) :

```
/usr/doc/support-db/sdb/kgw_liloconf.html
/usr/man/allman/man5/lilo.conf.5.gz
/usr/man/man5/lilo.conf.5.gz
/var/catman/cat5/lilo.conf.5.gz
```

Attention ! Ne pas arrêter le système n'importe comment ! ! !

Voir section 9.13 page 240.

9.1 Utilisation de la commande find

La commande `find` permet de chercher des fichiers, et eventuellement d'éxecuter une action dessus. Par exemple :

```
find . -print | less
```

affiche la liste de tous les fichiers du répertoire courant (l'option *-print* est normalement incluse par défaut). On peut rediriger les messages d'erreur vers le "trou noir" (le périphérique `/dev/null`) :

```
find . -print 2> /dev/null | less
```

ou les inclures (on rassemble alors en un seul canal la sortie standard et le sortie d'erreur standard) :

```
find . -print 2>&1 liste | less
```

Les options de la commande `find` sont nombreuses. Le tableau 9.1 en donne un apercu.

Option	Signification
-name	Recherche par **nom** de fichier.
-type	Recherche par **type** de fichier.
-user	Recherche par **propriétaire**.
-group	Recherche par appartenance à un **groupe**.
-size	Recherche par **taille** de fichier.
-atime	Recherche par date de **dernier accès**.
-mtime	Recherche par date de **dernière modification**.
-ctime	Recherche par date de **création**.
-perm	Recherche par **autorisations d'accès**.
-links	Recherche par **nombre de liens** au fichier.

TAB. 9.1 – Principales options de la commande **find**.

Pour les options *-size*, *-atime*, *-mtime*, *-ctime* et *-links*, il faut spécifier une valeur, précédée par le signe "+" pour "supérieur à", "-" pour "inférieur à", ou rien pour "égal à". Par exemple :

```
find . -mtime -3 -print
```

affiche les fichiers dont les dernières modifications remontent à moins de **3** jours (donc tous les fichiers modifiés entre aujourd'hui et il y a trois jours seront affichés). De même, *+5* afficherait les fichiers dont les dernières modifications remontent à plus de **5** jours.

Voici d'autres exemples d'utilisation de la commande `find` :

– Pour afficher tous les fichiers se terminant par "**.c**" :

```
find . -name ".c" -print
```

– Pour afficher tous les répertoires dont le nom se termine par "**s**" :

```
find . -type d -name "*s" -print
```

Pour afficher tous les fichiers, on aurait utilisé le code *f*.

– Pour afficher tous les fichiers ayant une taille de 10 blocs[1] :

```
find . -size 10 -print
```

ce qui est équivalent à demander la liste des fichiers ayant une taille de 5120 caractères :

```
find . -size 5210c -print
```

On aurait pu aussi demander la liste des fichiers ayant une taille **supérieure** (*+200k*) ou **inférieure** (*-200k*) à 200 Ko.

– Pour afficher tous les fichiers ayant une certaine permission exprimée en octale (voir section 1.9.6 page 24) :

```
find . -perm 755 -print
```

ou ayant **au minimum** les droits d'écriture pour le groupe :

```
find . -perm -020 -print
```

– Pour éxecuter la commande `ls -l` pour chaque fichier trouvé :

```
find . -type f -exec ls -l {} \;
```

et pour demander confirmation avant chaque action :

```
find . -type f -ok -exec rm {} \;
```

Toute autre commande peut être éxecutée avec l'option *-exec*.

[1]Un bloc est formé de 512 caractères.

– Pour associer plusieurs critères avec *-a* pour **et**, *-o* pour **ou**, *!* pour **négation**. Par exemple :

```
find . ! -user root -print
```

affiche tous les fichiers n'appartenant à **root**, et

```
find . \( -name '*.tex' -o -name '*.dvi' \) -print
```

affiche tous les fichiers se terminant par "**.tex**" ou "**.dvi**".

9.2 Utilisation de la commande locate

Lorsque le package GNU "find" est installé, en plus du programme de recherche se trouve un programme `locate` qui permet de retrouver rapidement l'emplacement d'un fichier. Ce programme se trouve souvent dans le répertoire `/usr/lib/locate`. Avant d'utiliser `locate`, il faut lancer la commande `/usr/lib/locate/updatedb`. Cela invoquera une commande `find /` sur les disques montés (penser à démonter les partitions MS-DOS et le CD-ROM...) et placera les noms de tous les fichiers dans le répertoire `/usr/lib/locate/find.codes`. La commande `locate` permettra ensuite de localiser rapidement l'un d'eux. On peut également insérer dans la crontab la ligne :

```
updatedb --prunepaths='/tmp /usr/tmp /var/tmp /proc /users
/root /dos /mnt /var /adm /var/spool /var/catman /home'
```

Cette commande réalise la mise à jour de l'arborescence sans les fichiers temporaires, l'arborescence utilisateur, une partition MS- DOS...

9.3 Archivage des données

9.3.1 gzip

`gzip` permet de compresser des fichiers, `gunzip` permet de les décompresser :

```
gzip salut.c
gunzip salut.c.gz
```

Par défaut, la vitesse et la qualité de la compression sont à *-6* ; la meilleure compression (*–best*) est à *-9* (c'est la plus lente) et la plus rapide (*–fast*) est à *-1* (c'est la moins fiable) :

```
gzip -9 salut.c
```

Bien entendu, `gunzip` saura décompacter le fichier sans aucune précision de vitesse.

L'option *-l* permet d'obtenir des informations sur le fichier compressé :

```
gzip -l salut.c.gz
```

Si aucun fichier n'est précisé, `gzip` tente de compacter les données en provenance de l'entrée standard :

```
ls -laR $HOME | gzip > list.gz
```

crée une liste de tous les fichiers du répertoire personnel, puis la sauvegarde dans le fichier compressé `liste.gz`.

L'option *-c* permet d'écrire le résultat sur la sortie standard :

```
gunzip -c liste.gz | less
```

`gunzip -c` est l'équivalente de la commande `zcat` sur la plupart des systèmes.

Le tableau 9.2 résume les principales options de la commande `gzip`.

Option	**Signification**
-1...-9	Qualité de la vitesse de compression ; **-6** par défaut.
-l	Permet d'obtenir des informations sur le fichier compressé.
-c	Permet d'écrire les résultat sur la sortie standard. On peut de cette facon utiliser un pipe.

TAB. 9.2 – Principales options de la commande **gzip**.

9.3.2 tar

`tar` permet de rassembler plusieurs fichiers en un seul :

```
tar cvf prog.tar prog/
```

On peut mettre plusieurs *v* pour avoir plus d'informations :

```
tar cvvf prog.tar prog/
```

Pour avoir des **informations** sur l'archive :

```
tar tvf prog.tar
```

Pour **extraire** les fichiers, se placer à l'endroit désiré et lancer :

```
tar xvf prog.tar
```

On peut n'**extraire qu'un seul fichier** de l'archive :

```
tar xvf prog.tar prog/C/salut.c
```

Pour **combiner** `tar` et `gzip`, on peut faire de deux facons :

```
tar cvf- <fichiers> | gzip -9c > <archive.tar.gz>
gunzip -9c <archive.tar.gz> | tar xvf-
```

Ou alors utiliser l'option `z` :

```
tar czvf <fichiers>
```

```
tar xzvf <archive.tar.gz>
```

Pour **consulter** une telle archive, il faut alors entrer :

```
tar tzvf <archive.tar.gz>
```

L'option *-T* permet de lire dans un fichier les données à sauvegarder :

```
tar -cv -T /tmp/liste.jour -f /dev/fd0
```

Comme alias pour résumer ces commandes, on peut par exemple définir (dans le fichier *.bashrc*) :

```
tarc() {tar czvf $1.tar.gz $1}
tarx() {tar xzvf $1}
tart() {tar tzvf $1}
```

Le tableau 9.3 résume les principales options de la commande `tar`.

Option	**Signification**
c	Crée une nouvelle archive.
x	Extrait des fichiers d'une archive.
t	Affiche le contenu d'une archive.
r	Ajoute des fichiers à une archive.
u	Met à jour les fichiers de l'archive.
d	Compare les fichiers contenus dans l'archive à ceux présents sur le disque dur.
v	Mode verbeux (donne plus d'informations).
k	N'écrase aucun fichier.
f *fichier*	Spécifie le nom du fichier d'archive à lire ou écrire.
z	Pour compresser / décompresser avec `gzip` avant d'archiver.
M	Pour enregistrer sur plusieurs disquettes.

TAB. 9.3 – Principales options de la commande **tar**.

9.3.3 Utilisation de tar et find pour les sauvegardes

On commence par rechercher les fichiers modifiés dans les dernières 24 heures (cf. `find` section 9.1 page 214) :

```
find / -mtime -1 \! -type d -print > /tmp/liste.jour
```

On archive :

```
tar -cv -T /tmp/liste.jour -f /dev/fd0
```

Pour des sauvegardes sur bandes, on pourra indiquer comme fichier de périphérique **/dev/rft0** (premier lecteur de bandes qui se connecte à la place d'une unitée de disquettes), **/dev/st0** (premier lecteur de bandes SCSI), **/dev/nrft0** (pareil que `/dev/rft0` sans rembobinage à la fin de la sauvegarde), **/dev/nrst0** (pareil que `/dev/st0` sans rembobinage à la fin de la sauvegarde). Pour rembobiner la bande, taper la commande :

```
mt -f /dev/nrst0 rewind
```

et pour retendre la bande (aller-retour complet, ce qui assure un défilement régulier) :

```
mt -f /dev/nrst0 reten
```

Pour aller au fichier suivant, taper :

```
mt -f /dev/nrst0 fsf 1
```

et pour aller deux fichiers plus loin (à partir de la position courante) :

```
mt -f /dev/nrst0 fsf 2
```

Il est nécessaire de bien positionner la bande avant d'utiliser `tar` (et après l'avoir utilisé, car la bande ne se trouve pas exactement en fin de fichier).

9.3.4 Utilisation de dump et restore pour les sauvegardes

L'utilisation de `dump` et `restore` est relativement simple. Pour effectuer la sauvegarde d'une partition **/dev/hda4** sur **/dev/rmt0**, par exemple, il suffit de faire :

```
dump 0sfu 3600 /dev/rmt0 /dev/hda4
```

ou, pour sauvegarder un disque sur un périphérique distant (par exemple situé ici sur la machine **zoubida**) :

```
dump 0sfu zoubida:/dev/rmt0 /dev/hda4
```

Option	Signification
0 à 9	Niveau de sauvegarde. **0** correspond à une sauvegarde complète, alors que les autres niveaux **n** correspondent à la sauvegarde des fichiers qui ont été modifiés depuis la n^{ieme} sauvegarde.
s	Taille de la bande (en pieds).
f	Fichier. Peut être composé de `machine:fichier`.
u	Ecriture de la date et du niveau de sauvegarde dans le fichier `/etc/dumpdates`.

TAB. 9.4 – Principales options de la commande **dump**.

Les options de `dump` peuvent sembler complexes. Le tableau 9.4 en présente une courte description.

Il existe deux modes pour effectuer une restauration : en ligne de commande ou en mode dit “interactif”. Le deuxième mode est plus simple pour des restaurations partielles. Le premier est surtout utilisé pour des restaurations complètes. Pour restaurer la bande en mode interactif, il suffit de faire :

```
restore -if /dev/rmt0
```

ou, pour restaurer à partir de la machine **zoubida** :

```
restore -if zoubida:/dev/rmt0
```

Dans ce cas, un mini-interpréteur de commandes est lancé. Utiliser la commande `help` pour plus de détails. Pour restaurer une bande complétement, lancer :

```
restore rf /dev/rmt0
```

Pour l'utilisation de `dump` et `restore` à travers un réseau (sauvegarde sur des périphériques distants), il faut utiliser des fichiers `.rhosts`. Dans l'exemple de sauvegarde ci-dessus, la machine *zoubida* doit contenir dans le fichier `.rhosts` le nom contenu dans la variable d'environnement **$HOSTNAME** (nom d'hôte du système sur lequel on opère).

9.3.5 cpio

La commande `cpio` permet également de faire des sauvegardes. Voici quelque exemples d'utilisation :

La commande `find` cherche tous les fichiers non utilisés depuis 7 jours que `cpio` sauvegarde sur disquette (**-o /dev/fd0**) avec en-tête dans chaque fichier ASCII (**-c**), par blocs de 5 Ko (**-B**) (cf. `find` section 9.1 page 214) :

```
find / -mtime -7 -print | cpio -ocvB > /dev/fd0
```

Les données sauvegardées sur disquette sont restaurées. Si le répertoire lu n'existe pas, il est crée (**-d**) :

```
cpio -ivcBd < /dev/fd0
```

Tous les fichiers dont le nom commence par une lettre comprise entre *a* et *m* seront restaurés :

```
cpio -ivcB "[a-m]*" < /dev/fd0
```

Une table des matières peut être crée (**-t**). Dans l'exemple qui suit, seule la table des matières sera restaurée :

```
cpio -ivctB < /dev/fd0
```

Le tableau 9.5 résume les principales options de la commande `cpio`.

Option	**Signification**
-o	Fichier à sauvegarder.
-i	Fichier destination.
-p	Indique le répertoire dans lequel les fichiers seront placés.
-B	Copie avec des blocs plus grands (5 Ko).
-c	Utilise pour chaque fichier un en-tête de gestion composé de caractères ASCII.
-t	Affiche la table des matières (avec l'option **-i**).
-d	Lors de la lecture, de nouveaux répertoires sont crées si nécessaire.

TAB. 9.5 – Principales options de la commande **cpio**.

9.3.6 dd

La commande `dd` permet également de faire des sauvegardes. Par exemple, pour la copie de quelque blocs (**count=**) d'un fichier (**if=**) vers un autre fichier (**of=**) :

```
dd if=/dev/fd0 of=/tmp/Disquette count=2
```

La commande `od` permet de lire le fichier copié (*/tmp/Disquette* dans l'exemple précédent).

L'option **bs=** permet de définir la taille des blocs en octets avec laquelle les données seront lues, converties et réécrites (512 octets par défaut) :

```
dd if=/usr/src/linux-2.0.33.pre.SuSE.3/arch/i386/boot/zImage
of=/dev/fd0 bs=8192
```

Le tableau 9.6 résume les principales options de la commande `dd`.

Option	Signification
if=	Fichier à copier.
of=	Fichier destination.
count=	Nombre de blocs à lire ou à sauvegarder.
bs=	Taille des blocs en octets avec laquelle les données seront lues, converties et réécrites (512 octets par défaut).
skip=	Nombre de blocs à sauter avant la lecture.
seek=	Nombre de blocs à sauter avant l'écriture.
conv=	Effectue différents types de conversion : **ebcdic** en jeu de caractères EBCDIC, **ascii** en jeu de caractères ASCII, **lcase** en minuscules, **ucase** en majuscules.

TAB. 9.6 – Principales options de la commande **dd**.

9.4 Ordonnancement de travaux avec crontab

La commande `crontab` permet de lancer des commandes à intervalles réguliers ou à certaines dates. Pour l'utiliser, il faut lancer la commande `crontab` avec l'option *-e*. L'éditeur par défaut est alors appellé[2] (**vi**). Il ne reste plus qu'à entrer les différents champs, dans un ordre particulier :

1. **minute** (0 à 59).
2. **heure** (0 à 23).
3. **jour du mois** (1 à 31).
4. **mois** (1 à 12), ou un nom com *jan, feb*...
5. **jour de la semaine** (0 à 6 : 0 = *Dimanche*, 1 = *Lundi*...ou *mon, tue* ...).
6. **commande**, telle qu'elle serait saisie sous shell.

[2]Pour changer d'éditeur par défaut, changer la valeur de la variable d'environnement **$VISUAL** : *export VISUAL=emacsclient* ou *setenv VISUAL=emacsclient* pour affecter Emacs, par exemple. Il se peut que la version de **crontab** demande que la variable **$EDITOR** doive être modifiée.

Quitter ensuite l'éditeur (`<Echap>:wq<Enter>`, pour **vi**), et vérifier éventuellement la saisie avec la commande `crontab -l`.

Voici quelque exemples sur la commande `find`, qui nettoie le répertoire `/tmp` des vieux fichiers :

- pour effectuer la commande le premier jour de chaque mois, à une heure du matin (il y a peu d'utilisateurs à cette heure là...) :

```
0 1 1 * * find /tmp -atime 3 -exec rm -f {} \;
```

- pour effectuer la commande tous les lundis :

```
0 1 * * mon find /tmp -atime 3 -exec rm -f {} \;
```

- pour effectuer la commande le premier et le quinze de chaque mois :

```
0 1 1,15 * * find /tmp -atime 3 -exec rm -f {} \;
```

- pour effectuer la commande tous les jours entre le premier et le quinze de chaque mois :

```
0 1 1-15 * * find /tmp -atime 3 -exec rm -f {} \;
```

- pour effectuer la commande tous les cinq jours (le premier, le 6, le 11...) :

```
0 1 */5 * * find /tmp -atime 3 -exec rm -f {} \;
```

L'exemple qui suit sert à vérifier tous les deux jours qu'aucun courrier n'est en attente dans la queue, et envoie un message à l'administratrur du courrier :

```
0 6 */2 * * mailq -v | mail -s "Messages bloques" postmaster
```

Pour ne pas recevoir de message à chaque action de `crontab`, et pour que tous ces messages soient redirigés vers un fichier (y compris la sortie d'erreur), entrer la commande suivante :

```
0 1 * * * find /tmp -atime 3 -exec rm -f {} \; >> $HOME/log 2&1
```

Les variable d'environnement **$USER**, **$HOME**, et **$SHELL** sont reconnues par `cron` (le programme qui lance le démon `crontab`). Bien entendu, un script peut être lancé à la place d'une commande (pour une commande ou une suite de commande difficile à faire tenir en une seule ligne).

La *crontab* est très utile si on utilise **UUCP** (pour envoyer du courrier, récupérer les news...). N'importe quel utilisateur peut se créer une crontab grâce la commande `crontab`.

9.5 Gestion des comptes utilisateurs

Les fichiers de gestion d'utilisateurs sont `/etc/passwd` et `/etc/group`. La commande qui gère les utilisateurs est `adduser`.

Chaque compte est référencé dans le fichier `/etc/passwd`, qui contient une ligne par utilisateur avec divers attributs. Voici la syntaxe du fichier `/etc/passwd` :

```
Nom:Mot de passe:No d'utilisateur:No de groupe:Champ special:
Repertoire personnel:Programme de demarrage
```

Le mot de passe est bien entendu crypté (0 pour root), et si le premier charactère de ce champ est une astérisque (`*`), il ne correspondra alors à aucun mot de passe possible. Ce champ peut par ailleurs être vide (mais c'est déconseillé...). Pour changer de mot de passe, taper la commande `passwd`. Le numéro d'utilisateur **UID** est propre à l'utilisateur, c'est par lui que le système distingue les différents utilisateurs. De même le numéro de groupe **GID** est propre à un groupe d'utilisateurs. Le **champ spécial** contient des informations personnelles, comme le véritable nom de l'utilisateur ou son adresse. Des programmes comme `finger` ou `mail` peuvent utiliser ces informations.

A propos du mot de passe, le fichier `/etc/passwd` doit être autorisé en lecture par tous car c'est sur ce fichier que se basent certains programmes. Ce mot de passe est souvent matérialisé par le charactère '**x**' dans le fichier `/etc/passwd`, ce qui signifie qu'en fait il est stoqué pour des raisons évidentes de sécurité dans un autre fichier aux droits d'accés plus resctrictifs. En effet, même crypté, il n'est pas impossible de retrouver le mot de passe... Ce fichier, qui contient lui les mots de passes cyptés, s'appelle `/etc/shadow`. C'est la commande `/usr/sbin/pwconv` qui s'occupe de transférer les mots de passes cryptés, commande à utiliser donc après chaque modification de `/etc/passwd`.

Pour créer un compte "à la main", il suffit de rajouter dans les fichiers `/etc/passwd` et, si besoin est, dans le fichier `/etc/group`, une ligne comme expliqué ci-dessus, et de créer le répertoire personnel :

```
mkdir /home/toto
cp /etc/skel/* /home/toto
chown toto /home/toto
chgrp le_groupe_de_toto /home/toto
```

Il ne reste plus qu'à attribuer un mot de passe en tapant `passwd toto`. Voici la syntaxe du fichier `/etc/group` :

```
Nom de groupe:Champ special:No de groupe:Membre1, Membre2...
```

Tous les utilisateurs d'un même groupe peuvent jouir des mêmes droits d'accès sur les fichiers de leurs groupes (voir section 1.9.6 page 24 concernant les droits d'accès aux fichiers). Les commandes utiles sont `chown` (change de propriétaire, option **-R** pour tous les sous répertoires), `chgrp`[3] ou `newgrp` (change de groupe), `chmod` (change de droits d'accés) et bien entendu `mkdir`. Noter que si on n'opère pas en tant que root, la commande `chown` ne permet plus de changer de groupe, il vaut donc mieux l'exécuter en dernier...

Pour supprimer un compte, il existe une commande :

```
userdel -r piou
```

L'option **-r** force à la suppression du répertoire personnel. Tous les fichiers restants appartenant à l'utilisateur **piou** peuvent être trouvés grâce à la commande `find` :

```
find / -user piou -ls
```

ou, si la ligne concernant l'utilisateur dans le fichier `/etc/group` a été supprimée, avec l'option **-uid** :

```
find / -uid 501 -ls
```

9.6 Pilotes de périphériques chargeables (modules)

Un module est un fichier objet (*.o*) qui peut être chargé en mémoire et enlevé sans avoir à relancer le système. Ce sont des pilotes chargeables, ajoutés ou supprimés de la mémoire pendant le fonctionnement du système par une série de commandes. Par exemple pour le pilote de lecteurs de bandes à interface disquettes, *ftape*, on peut l'installer avec la commande `insmod` (les modules sont souvent installés dans le répertoire `/boot`) :

```
insmod /boot/ftape.o
```

La commande `lsmod` affiche la liste des modules chargés en mémoire[4] et la commande `rmmod` supprime le module. Cette facon de programmer rend le déboguage plus facile. La commande `kerneld` pourra charger automatiquement les modules nécessaires au démarrage.

[3]La commande **chown toto.users /home/toto** affectera le propriétaire *toto* du groupe *users* au répertoire */home/toto*.

[4]Une page occupe 4 Ko sous Linux.

9.7 Les fichiers de périphériques

Les fichiers de périphériques permettent aux programmes d'accéder au matériel qui équipe la machine. La commande `ls -l` appliquée à l'un de ces fichiers donne des informations spéciales, comme :

- **le type** (première lettre), comme *b* pour *bloc* comme pour les disques durs, par exemple, et dont les données sont lues et écrites sous forme des blocs entiers, ou *c* pour *caractère* pour lesquels les entrées/sorties peuvent s'effectuer octet par octet.
- **les numéros *majeurs* et *mineurs*** associés au type de périphérique. Le noyau ne se base que sur ces numéros pour reconnaître le périphérique.

Pour la mise en place d'un nouveau pilote, il peut être nécessaire de créer un fichier spécial, à l'aide de la commande `mknod`. Par exemple, pour un fichier de permissions *666* (option *-m*), de type *b* et de numéros majeurs et mineurs *42* et *0*, entrer :

```
mknod -m 666 /dev/bidon b 42 0
```

Les fichiers spéciaux ne doivent en général pas être accessibles aux utilisateurs, pour des raisons de sécurité (certains, comme pour les ports séries ou les consoles virtuelles, le doivent cependant).

Pour savoir comment intégrer un nouveau module, voir la section 9.14 page 240.

9.8 Monter et démonter un système de fichiers

La commande `mount` permet de monter un système de fichiers (partition, lecteur de disquette ou de cédérom, second disque dur...). On peut alors accéder aux fichiers contenus dans ce système de fichiers. La commande `mount -a` permet de monter tous les systèmes de fichiers déclarés dans le fichier `/etc/fstab`. Voici un exemple de fichier `/etc/fstab` :

```
/dev/hda3       /                         ext2         defaults    1   1
/dev/hda2       swap                      swap         defaults    0   0
/dev/hda6       /usr                      ext2         defaults    1   2
/dev/hda5       /home                     ext2         defaults    1   2
/dev/hda1       /Dos/C                    vfat         defaults    0   0

/dev/hdb        /cdrom                    iso9660      ro,noauto,user 0    0
```

```
none            /proc                     proc          defaults   0   0
```

L'ordre des champs dans ce fichier est le suivant :

périphérique point de montage type options

L'option *user* permet à tous les utilisateurs de monter un périphérique (un lecteur de CD-ROM, par exemple).

La syntaxe de la commande `mount` est la suivante :

`mount -t` *type_périphérique point_de_montage*

Le tableau 9.7 présente les principaux types de systèmes de fichiers utilisés.

Type	Utilisation
ext2	Le plus courant sous Linux.
msdos, vfat	Pour les fichiers MS-DOS.
iso9660	Déstiné à la plupart des CD-ROM.
proc	Systèmes de fichiers virtuels, qui donne des informations sur le noyau (utilisé par `ps`, par exemple). Voir section 9.14.1 page 241.

TAB. 9.7 – Principaux types de **systèmes de fichiers** utilisés sous Linux.

Le périphérique est situé en général dans le répertoire `/dev`.

Le point de montage est le répertoire où le systèmes de fichiers doit être intégré.

Voici quelque exemples de montages ou de démontage de systèmes de fichiers :

Montage d'une partition Dos :

```
mount -t vfat /dev/hda1 /Dos/C/
```

ou

```
mount /dev/hda1 /Dos/C/
```

Pour convertir automatiquement les fins de lignes des fichiers ASCII MS-DOS au format UNIX[5], utiliser l'option *conv* :

[5]Les fichiers MS-DOS contiennent un retour chariot suivit d'un saut de ligne, à la différence des fichiers UNIX qui contiennent simplement un saut de ligne à la fin de chaque ligne. Cette différence est à l'origine de “l'effet d'escalier” lorsqu'on imprime des fichiers MS-DOS, voir section 9.15.5 page 262.

```
mount -o conv=auto -t vfat /dev/hda1 /Dos/C/
```

Démontage d'une partition Dos (le systèmes de fichiers ne doit pas être en cours d'utilisation) :

```
umount /Dos/C/
```

Montage du lecteur de CD-ROM :

```
mount -t iso9660 /dev/hdb /cdrom
```

ou

```
mount /dev/cdrom /cdrom
```

On peut préciser l'option *-o ro* ou *-r* pour monter un périphérique en lecture seule, tel qu'un CD-ROM ou une disquette protégée en écriture par exemple.

Montage du lecteur de disquettes :

```
mount /dev/fd0 /floppy
```

La commande `mount` sans arguments affiche les systèmes de fichiers actuellement montés.

Lorsqu'on écrit sur un périphérique, les données sont envoyées quand ca arrange le système. Pour forcer l'écriture tout de suite, on peut invoquer la commande `sync`.

Une autre commande utile qui permet de gérer les systèmes de fichiers est la commande `df` qui affiche l'espace occupé et l'espace libre sur chaque systèmes de fichiers monté. L'option `-h` permet d'afficher le résultat en format lisible par un humain (les tailles sont affichées en Mo). Il existe aussi une commande qui donne l'espace disque occupé en Ko pour chaque fichier ou répertoire demandé (pour le répertoire courant si rien n'est spécifié). Il s'agit de la commande `du`.

9.9 Installation de nouvelles partitions

Pour installer une nouvelle partition, il faut suivre les étapes suivantes :

1. vérifier qu'il existe un fichier de périphérique et qu'il est bien accessible par le noyau du système. Le tableau 9.8 présente une brève liste de correspondance entre les fichiers et les périphériques matériels.
2. une fois que le noyau du système d'exploitation a accès physiquement au disque dur, partitionner les nouveaux disques avec la commande `fdisk`. Le tableau 9.9 présente les principales commandes de `fdisk`.

3. créer un nouveau système de fichiers sur les nouvelles partitions à l'aide de la commande `mkfs` ou de l'une de ses variantes. Suivant le type de système de fichiers, on utilisera par exemple `mke2fs` pour un système de fichiers de type *Extended 2*, `mkfs.minix` pour un système de fichiers de type *Minix*...Les système de fichiers de type *Extended 2* sont les mieux gérés par Linux, c'est donc celui conseillé. Par exemple pour une disquette :

```
mke2fs /dev/fd0
```

4. monter manuellement ou automatiquement (au démarrage) ce système de fichiers à l'aide de la commande `mount`.

Fichier	**Périphériques matériel**
/dev/hda	Totalité du premier disque dur IDE.
/dev/hda1	Première partition du premier disque dur IDE.
/dev/hda2	Seconde partition du premier disque dur IDE.
/dev/hda5	Premier lecteur logique d'une partition étendue MS-DOS.
/dev/hda6	Second lecteur logique d'une partition étendue MS-DOS.
/dev/hdb	Totalité du second disque dur IDE ou lecteur de CD-ROM ATAPI.
/dev/hdb1	Première partition du second disque dur IDE.
/dev/hdb2	Seconde partition du second disque dur IDE.
/dev/sda	Totalité du premier disque dur SCSI.
/dev/sda1	Première partition du premier disque dur SCSI.
/dev/fd0	Premier lecteur de disquette.
/dev/fd1	Second lecteur de disquette.

TAB. 9.8 – Correspondance entre les **fichiers** et les **périphériques**.

Commande	**Signification**
m	Affiche la liste des commandes.
p	Affiche la liste des partitions existantes.
n	Ajoute une nouvelle partition.
t	Change le type d'une partition.
w	Ecrit la table des partitions.
q	Quitte `fdisk`.

TAB. 9.9 – Principales commandes de **fdisk**.

9.10 Création d'une zone de swap

Pour créer une zone de swap, il faut :

1. créer une partition (à l'aide de `fdisk`) ou un fichier (à l'aide de `touch`) destiné(e) à abriter la zone de swap.
2. générer un fichier de swap. Par exemple, pour un fichier de 8 Mo (8192 blocs de 1024 octets) :

   ```
   dd if=/dev/zero of=/swap bs=1024 count=8192
   ```

   pour un fichier, et

   ```
   dd if=/dev/zero of=/dev/hda2 bs=1024 count=8192
   ```

   pour une partition.
3. appeler la commande `sync`.
4. formater cette zone à l'aide de la commande `mkswap` (la taille est exprimée en blocs de 1024 octets). L'option *-c* sert à rechercher les blocs défectueux :

   ```
   mkswap -c /swap 8192
   ```

   pour un fichier, et

   ```
   mkswap -c /dev/hda2 8192
   ```

   pour une partition.
5. appeler à nouveau la commande `sync`.
6. mettre en service la zone de swap à l'aide de la commande `swapon` et vérifier à l'aide de la commande `free` le nouvel espace total disponible pour le swap :

   ```
   swapon /swap
   ```

   pour un fichier, et

   ```
   swapon /dev/hda3
   ```

   pour une partition.

Cette zone de swap peut être mise en place au démarrage du système par les script de démarrage (voir section 9.12 page 237), généralement le fichier `/etc/rc.d/rc.sysinit`. La commande `swapon -a` s'effectue alors pour toutes les entrées du fichier `/etc/fstab` contenant l'option *sw* (comme "swap"). Une telle entrée pourait avoir la forme suivante :

```
/swap     none     swap     sw
```

pour un fichier, et

```
/dev/hda2     none     swap     sw
```

pour une partition.

Pour mettre hors service une zone de swap, il suffit d'entrer la commande `swapoff` suivie du nom du périphérique. Ainsi, la commande :

```
swapoff /swap
```

met hors service une zone de swap contenue dans un fichier, et

```
swapoff /dev/hda2
```

met hors service une zone de swap contenue dans une partition. C'est seulement après qu'on peut effacer fichier ou partition.

9.11 Charger Linux

9.11.1 Comment se démarre Linux ?

Linux se démarre à partir d'une disquette ou d'un disque dur. Dans tous les cas, une image du noyau se décompresse en mémoire et tente de monter un système de fichier : la racine (/). Le noyau est chargé en mémoire jusqu'au prochain démarrage de la machine. L'image du noyau peut être installé sur une disquette ou sur un disque dur au moyen de la commande `rdev`. Le noyau est souvent placé dans le fichier `vmlinuz` situé à la racine ou dans un répertoire spécial. On peut trouver l'emplacement de ce fichier grâce à la commande `locate` :

```
locate vmlinuz
```

9.11.2 Charger Linux à partir d'une disquette

Si le fichier `/vmlinuz` abrite une image du noyau[6], il faut d'abord assigner le périphérique à monter sur la racine grâce à la commande `rdev`. Pour connaitre le nom du périphérique racine, taper :

```
rdev /vmlinuz
```

Le résultat s'affiche alors :

[6] Ce peut être aussi le fichier **zImage**, situé dans un des sous-répertoires du répertoire **/usr/src**.

```
Root device /dev/hda3
```

Si on se contente de taper `rdev` sans aucun argument, on obtient alors :

```
/dev/hda3 /
```

Si le système de fichiers est `/dev/hda2` par exemple, taper :

```
rdev /vmlinuz /dev/hda2
```

ou, si le noyau est sur la disquette :

```
rdev /dev/fd0 /dev/hda2
rdev -R /dev/fd0 1
```

Il faut maintenant copier le noyau sur une disquette formatée sous MS-DOS (`format a:`) ou sous Linux (avec `fdformat`, voir section 10.1 page 271). Cette copie se fait grâce à la commande `dd` (voir section 9.3.6 page 221)[7] :

```
dd if=/vmlinuz of=/dev/fd0 bs=8192
```

Il ne reste plus qu'à relancer le système grâce à la commande `shutdown` (voir section 9.13 page 240). La disquette d'amorcage devra bien entendu être présente dans le lecteur de disquette au moment du prochain démarrage de Linux...

9.11.3 Comment démarrer Linux avec LILO ?

LILO est un gestionnaire d'amorcage qui est capable de démarrer n'importe quel système d'exploitation installé sur le disque dur (donc Linux). S'il est installé sur le secteur d'amorcage du disque dur (MBR), c'est le premier code qui sera exécuté.

Parmis les systèmes d'exploitation autres que Linux, il faut faire attention pour deux d'entre eux : MS-Windows et OS/2.

MS-Windows 95 écrase le contenu du MBR du disque primaire, il faut donc installer LILO *après* avoir installé MS-Windows 95, et pour MS-Windows-NT il faut installer LILO en tant que chargeur secondaire (ou sur une disquette). Pour OS/2 même remarque que pour MS-Windows-NT, mais en plus il faudra faire les partitions Linux avec OS/2, le gestionnaire d'amorcage de OS/2 êtant bogué.

Le fichier de configuration de LILO est souvent appelé `/etc/lilo.conf`. Voici un exemple de fichier `/etc/lilo.conf` :

[7] Cette copie peut aussi se faire avec la commande **cat**, comme **cat /zImage > /dev/fd0**, par exemple, mais bon...Quand on est administrateur système, ca fait plus classe avec **dd** y paraît...

```
#------------------------------#
# Exemple de fichier lilo.conf #
#------------------------------#
#
# LILO est un gestionnaire d'amorcage qui est capable de demarrer
# n'importe quel systeme d'exploitation installe sur le disque dur
#
# lancer la commande /sbin/lilo pour installer LILO
#
#-------------------------------#
# Quelque parametres essentiels #
#-------------------------------#
#
# boot= : indique le nom du peripherique sur lequel LILO doit s'installer
# ici : /dev/hda (disque dur IDE). LILO sera installe dans cette partition
# en tant que chargeur primaire.
# pour installer LILO sur disquette par exemple, on pourrait mettre :
# boot=/dev/fd0
#
# compact permet d'effectuer quelque optimisations.
#
# map= : designe le fichier que LILO genere lorsqu'il est installe.
# Ce fichier sera absent jusqu'a la premiere installation de LILO.
#
# install= : designe le fichier contenant le code d'amorcage a
# recopier sur le secteur du disque dur.
#
# on pourait aussi ajouter une ligne comme
# delay=50
# qui indiquerait le temps d'attente au demarrage avant que LILO charge par
# defaut le premier systeme specifie, mais bon...
#
boot=/dev/hda
compact
map=/boot/map
install=/boot/boot.b
#
prompt
message="/etc/boot.msg"
#
#------------------------------------------------#
```

```
# Configuration de chaque systeme d'exploitation #
#------------------------------------------------#
#
#--------------------------------------------------------#
# Section pour Linux avec partition racine sur /dev/hda3 #
#--------------------------------------------------------#
#
# image= : nom de l'image du noyau (son emplacement)
#
# label= : nom du systeme pour le menu d'amorcage
#
# root= : emplacement de la partition racine. Pour une disquette :
# root=/dev/fd0
# Si rien n'est specifie, ce seront celles definies par la commande rdev.
# Sinon, ce sont les valeurs de ce fichier qui seront prises en compte.
# La commande rdev n'a alors plus aucune utilite.
#
# vga= : specifie le mode d'affichage textuel utilise par la console.
# Options possibles : normal (80x25), extended (132x44 ou 132x60) ou ask
# (pour avoir le choix au demarrage). Les chiffres de 1 a 3 correspondent
# au numero du mode lorsque l'option ask est requise
#
image=/vmlinuz
label=linux
root=/dev/hda3
vga=ask
read-only
#
image=/vmlinuz
append="load_ramdisk=1 prompt_ramdisk=0 ramdisk_start=550"
label=linux_floppy
root=/dev/hda3
read-only
#
image=/vmlinuz
append="load_ramdisk=1 prompt_ramdisk=0 ramdisk_start=550 rescue"
label=rescue
root=/dev/fd0
read-only
#
#-----------------------------------#
```

```
# Configuration de MS-DOS (ou OS/2) #
#----------------------------------#
#
# Si MS-DOS est situe sur un autre disque dur, ajouter la ligne :
# loader=/boot/any_d.b
#
# Si OS/2 est situe sur un autre disque dur, ajouter la ligne :
# loader=/boot/os2_d.b
#
# other= : emplacement de la partition MS-DOS
#
# table= : emplacement de la table de partition dont depend la partition
# MS-DOS
#
# label= : nom du systeme pour le menu d'amorcage
#
other=/dev/hda1
table=/dev/hda
label=msdos
```

Il ne reste plus qu'à lancer la commande `/sbin/lilo` pour installer LILO avec éventuellement l'option `-v` pour avoir des détails au cas ou quelque chose clocherait, ou l'option `-C` pour spécifier un autre fichier que `/etc/lilo.conf`.

Il faut vaut mieux supprimer les fichiers `/boot/boot.0300` pour les disques IDE et `/boot/boot.0800` pour les disques SCSI avant d'installer LILO...

Au redémarrage du système avec la commande `shutdown` (voir section 9.13 page 240), le premier système d'exploitation indiqué dans le fichier de configuration sera lancé automatiquement. Pour en lancer un autre, maintenir enfoncé la touche `Shift` ou `Ctrl` lors du démarrage de la machine et à l'invite taper le nom du système désiré (un de ceux définits par la ligne *label* du fichier `/etc/lilo.conf`) :

```
boot:
```

Appuyer sur la touche `Tab` pour afficher la liste des options d'amorcage (les noms définits par la ligne `label` du fichier `/etc/lilo.conf`).

Pour **installer LILO en tant que chargeur secondaire** (sur la partition racine du système, `/dev/hda3` par exemple), il suffit de modifier dans le fichier `/etc/lilo.conf` la ligne `boot=` comme suit :

```
boot=/dev/hda3
```

Cette partition doit être déclaré comme bootable avec la commande `fdisk` (commande **a**) et doit être une partition primaire. C'est de cette facon que l'on peut faire cohabiter Linux et MS-Windows-NT ou OS/2 (il faut alors les déclarer comme autres systèmes d'exploitation dans le fichier `/etc/lilo.conf`).

On peut spécifier des paramètres à l'amorcage. Le tableau 9.10 en présente une liste sommaire.

Option	**Signification**
hd=683,16,38	Indique la géométrie du disque dur (respectivement les nombres de **cylindres**, **têtes** et **secteurs**).
single	Lance le système en mode mono-utilisateur (la configuration est sommaire et un shell **root** est lancé).
root=/dev/hda3	Tente de monter la partition `/dev/hda3` (dans cet exemple) en tant que système de fichier (au lieu de celle spécifiée dans le fichier `/etc/lilo.conf`).
ro	Monte la racine en lecture seule (employé pour lancer la commande `fsck` (voir section 9.18 page 265).
vga=	Spécifie le mode d'affichage (valeurs possibles : **normal** -80x25-, **extended** -132x44 ou 132x60- ou **ask** -pour avoir le choix au démarrage).

TAB. 9.10 – Principaux paramètres à l'amorcage avec **LILO**.

On peut spécifier ces options dans le fichier `/etc/lilo.conf` comme suit (pour indiquer la taille du disque dur dans l'exemple qui suit) :

```
append=''hd=683,16,38''
```

ou même encore en spécifier plusieurs sur une même ligne (pour indiquer la taille des deux disques durs dans l'exemple qui suit) :

```
append=''hd=683,16,38 hd=64,32,202''
```

En cas de problème avec LILO, il faut d'abord essayer de supprimer l'option *compact* qui peut se trouver dans son fichier de configuration. Ne surtout pas oublier de relancer LILO après chaque modification de son fichier de configuration !

Pour **supprimer LILO**, on peut remplacer le code de LILO par un code d'amorcage ordinaire avec la commande `fdisk` de MS-DOS (sous DOS, donc) :

```
FDISK /MBR
```

Les copies de sauvegarde du secteur d'amorcage original (avant installation de LILO) sont sauvegardées dans les fichiers `/boot/boot.0300` pour les disques IDE et `/boot/boot.0800` pour les disques SCSI. Pour remettre en place l'ancien secteur de cette sauvegarde, il faut lancer :

```
dd if=/boot/boot.0300+ of=/dev/hda bs=446 count=1
```

Le code d'amorcage est contenu sur les 446 premiers octets, ce sont seulement ceux-là qui sont copiés (le reste contient la table des partitions, jusqu'à 512).

Attention ! Bien vérifier que ces fichiers comprennent les secteurs d'amorcage désirés (il existe parfois par défaut des versions inutilisables de ces fichiers, il faut les supprimer avant d'installer LILO...).

9.11.4 Mettre une console externe comme console

Pour mettre comme console un minitel ou un vieux vt100 qui ne passe pas sur la carte vidéo mais sur le port série, il suffit d'ajouter dans le fichier `lilo.conf` la ligne suivante :

```
append = "scon=0x03f8  serial=0,9600n8"
```

9.11.5 Utiliser deux cartes réseaux

Certaines machines possèdent deux (ou plus) cartes réseaux. Pour qu'elles soient toutes les deux reconnues, il est nécessaire d'ajouter dans la configuration de LILO la ligne suivante :

```
append="ether=0,0,eth1"
```

9.12 Initialisation du système Linux

La commande `/sbin/init` lit le fichier `/etc/inittab` et configure le système en fonction de ce fichier. Chaque ligne est traitée et contient une instruction à lancer. Voici la syntaxe d'une ligne de ce fichier :

```
Abreviation:Niveau d'execution:Mode:Commande
```

Chaque ligne contient quatre champs séparés par " `:` ". Voici les détails de chaque champs :

- **Abreviation** : chaque entrée est différenciée par une abréviation.

- **Niveau d'exécution** : indication du niveau pour lequel l'action correspondante doit être entreprise. Ces modes sont représentés par des chiffres allant de **0** à **6** et les lettres "**S**" ou "**s**". Le mode **1** correspond au mode **mono-utilisateur**, les modes **2** et **5** correspondent aux modes **multi-utilisateur**. Les modes **0** et **6** correspondent aux activitées qui sont à exécuter avant la mise hors tension de la machine. Les niveaux **3** et **4** sont généralement libres.
- **Mode** : des mots cléfs indiquent avec quelle fréquence et de quelle manière la commande du dernier champ doit être exécutée. Le tableau 9.11 présente les modes possibles.

Mode	Signification
respawn	`init` devra relancer la commande correspondante chaque fois que le processus correspondant se terminera (par exemple si un utilisateur se déconnecte, il faut lancer la commande `getty` ou l'une de ses variantes).
off	La commande du dernier champ ne doit pas être exécutée.
wait	Attends la fin de l'exécution de la commande correspondante.
sysinit	La commande correspondante n'est lancée qu'au démarrage du système.
powerfail	La commande correspondante ne sera traitée que si le processus `init` a recu un signal SIGPWR (correspondant à une rupture d'alimentation électrique).
initdefault	Ce mode définit le niveau d'exécution au démarrage du système

TAB. 9.11 – **Modes** possibles pour le fichier **/etc/inittab**.

- **Commande** : La commande sera traitée par le shell.

Le script shell `/etc/rc.d/rc` exécute une série de script situés dans les répertoires `/etc/rc.d/rcN.d`, où **N** repésente un niveau d'exécution. Ces répertoires contiennent des scripts de la forme `Snnxxxx` ou `Knnxxxx`, où '**S**' signifie que la commande '**xxxx**' sera activée (*Start*) et '**N**' signifie que la commande '**xxxx**' sera désactivée (*Kill*). Une fois atteint le niveau d'exécution N, les scripts sont lancés dans l'ordre suivant les *nn* croissants, avec *nn* compris entre 00 et 99. Au démarrage et à l'arrêt du système, ces scripts seront exécutés dans l'ordre. Le script `S99local` est un lien symbolique vers le script `/etc/rc.d/rc.local` (ou `/etc/rc.local`). C'est donc ce script qui sera lancé en dernier. Ce script pourra contenir certaines commandes système définies par l'administrateur système qui seront lancées au démarrage. En voici un exemple :

```
#! /bin/sh

# Un peu de menage...
echo "Nettoyage de /tmp..."
find /tmp -atime 3 -exec rm -f {} \;

# On peut aussi tout effacer, mais c'est dangereux...

#/bin/rm -fr /tmp
#mkdir /tmp
#chmod 1777 /tmp

# Pour un clavier azertyuiop

echo "Chargement du clavier francais..."
/usr/bin/loadkeys /usr/lib/kbd/keytables/fr-latin1.map

# En cas de probleme avec le clavier azertyuiop...

keycode   3 = eacute          two              asciitilde
keycode   8 = egrave          seven            grave
keycode  10 = ccedilla        nine             asciicircum
keycode  14 = Delete          BackSpace

# Pour avoir la touche <Num Lock> configuree au demarrage

for tty in /dev/tty[1-9]*;
do
     setleds -D +num < $tty > /dev/null
done

# Pour les changements d'heure d'ete/hiver

clock -s
```

9.13 Arrêter le système Linux

La commande `shutdown` permet d'arrêter proprement le système. Le tableau 9.12 présente les principales options de cette commande.

Option	Signification
h:mn	Heure à laquelle il faut arrêter le système.
+n	Arrêter le système dans n minutes.
now	Arrêter le système maintenant.
"*Message...*"	Affiche le message sur tous les terminaux actifs.
-f	Arrêt plus rapide.
-r	Redémarrer l'ordinateur une fois la procédure d'arrêt terminée.
-h	Le système doit être arrêté sans redémarrage.
-q	Les messages ne doivent pas être affichés à l'écran.
-k	Affiche les messages sans redémarrer le système.
-c	Annule la procédure d'arrêt en cours.

TAB. 9.12 – Principales options de **shutdown**.

La séquence de touches ***Ctrl-Alt-Del*** effectue un arrêt du système au moyen de la commande `shutdown` comme définit dans le fichier `/etc/inittab` :

```
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
```

Les commandes `reboot`, `halt` et `fasthalt` sont des racourcis vers `shutdown` avec les options appropriées (respectivement `-r now`, `-h now` et `-f now`).

9.14 Recompiler le noyau

Le noyau est le premier programme lancé au démarrage. Il sert de relais entre le matériel et les programmes, mais il charge en mémoire au démarrage des modules relatifs à un matériel qui n'est pas présent sur le système (pour être sûr que Linux marche bien après l'installation). Il faut donc supprimer ces modules, et charger ceux dont on a besoin et qui ne seraient pas chargés au démarrage (comme pour la carte son, par exemple). Il faut donc recompiler le noyau.

9.14.1 Trouver des informations sur le noyau

Des informations nécessaires peuvent êtres contenues dans les fichiers du répertoire `/proc`, comme sur les pilotes :

```
cat /proc/devices
```

ou sur les types de système de fichier dans le noyau :

```
cat /proc/filesystems
```

Jeter aussi un coup d'oeil aux fichiers `/proc/cpuinfo`, `/proc/pci`, `/proc/interrupts`, `/proc/dma`, `/proc/ioports`. Ces informations pourront être utiles si le nouveau périphérique n'est pas reconnu après recompilation du noyau.

De plus, les messages affichés par le noyau lors de l'initialisation du système peut être utiles :

```
dmesg
```

La compilation doit s'effectuer dans le répertoire `/usr/src/linux-x.y.z` (ou `/usr/src/linux`), où *x, y* et *z* représentent les numéros de version du noyau tels qu'affichés par la commande `uname -a`.

Si on remplace le noyau par une version plus récente (pour un programme qui en aurait besoin), il faut décompresser les sources (`tar`...) et faire un peu de ménage (`make mrproper`).

9.14.2 Patcher le noyau

Pour appliquer un patch au noyau avant de le recomplier, taper :

```
cd /usr/scr
gunzip -c fichier.patch | patch -p0
```

Il est préférable d'appliquer les patchs un à un (*patch1, patch2*...), car après *patch1* vient *patch10* et non *patch2* si on introduit une commande du type *patch**. Pour vérifier que tout s'est bien passé, entrer :

```
find /usr/src/linux -follow -name ‘‘*.rej’’ -print
find /usr/src/linux -follow -name ‘‘*#’’ -print
```

pour trouver ceux qui ont été rejetés.

9.14.3 Passer à la compilation

On peut ensuite commencer la configuration :

```
make menuconfig
```

On peut choisir à la place de `menuconfig` simplement `config` qui présente un programme textuel, moins pratique, ou `xconfig` qui contient une aide plus importante.

Il faut ensuite choisir les modules, puis quitter après avoir sauvegardé, et recompiler :

```
make dep; make clean
```

Cette opération peut prendre du temps. On peut maintenant construire l'image :

```
make zImage
```

qui sera dans le répertoire `/usr/src/linux/arch/i386/boot/`.

Pour ré-installer LILO, taper `make zlilo` au lieu de `make zImage`. Pour charger le nouveau noyau à partir d'une disquette, procéder comme suit (garder tout de même l'ancienne disquette de boot, au cas où...).

Les opérations qui suivent consistent à formater une disquette (`fdformat`), à s'assurer que le disque dur sera bien monté en lecture seule au démarrage pour vérifier le système de fichiers avant de lancer Linux (`rdev`) et à copier le nouveau noyau sur la disquette (`cat`) :

```
fdformat /dev/fd0H1440
rdev -R zImage 1
cat zImage > /dev/fd0
```

Il ne reste plus qu'à relancer le système :

```
sync; sync
reboot
```

et si ca ne marche pas, essayer de relancer avec l'ancien noyau ou vérifier que c'est bien le nouveau noyau qui est lancé à l'aide de la commande `uname -a` (regarder en particulier la date de construction de ce noyau).

9.14.4 Tester le périphérique son

Si on a installé une carte son, on peut obtenir des informations sur l'état du driver son dans le fichier `/dev/sndstat`. Pour essayer cette carte son, essayer :

```
cat piano-beep.au > /dev/audio
```

dans le répertoire contenant ce fichier (essayer avec `/dev/dsp` sinon). S'il est lu, la carte son fonctionne correctement. On peut aussi tester le micro (s'il est installé) en enregistrant 10 secondes :

```
dd bs=8k count=10 < /dev/audio > test.au
```

En relisant le fichier *test.au*, on peut voir si le micro fonctionne.

Attention ! Si plusieurs périphériques son sont chargés par le noyau, ca risque de ne pas marcher. Il faut donc ne **charger que celui qui est utilisé par la carte son** lors de la recontruction du noyau.

9.14.5 Installer de nouvaux modules

Pour installer de nouveaux modules, si lors de la recompilation du noyau on en a sélectionnés, taper :

```
make modules
```

puis

```
make module_install
```

pour installer les nouvaux modules. On peut vérifier qu'ils sont bien installés par l'une des commandes suivantes :

```
ls -l /usr/src/linux/modules/
```

qui affiche la liste des liens symboliques des nouvaux modules, ou :

```
modprobe -l
```

qui affiche la liste de tous les modules disponibles, ou :

```
less /lib/modules/2.0.33/modules.dep
```

qui affiche la liste des modules et leurs dépendances. Il ne reste plus qu'à charger chaque module. Par exemple :

```
insmod umsdos
```

charge le module *msdos*. On peut afficher la liste des modules chargés :

```
lsmod
```

ou (équivalent) :

```
cat /proc/modules
```

Enfin, la commande `ksyms` permet d'afficher des informations sur chaque module :

```
ksyms -m
```

affiche des informations sur les symboles et l'affectation de la mémoire. La commande `rmmod` enlève les modules, qu'il faudra peut-être tuer avant (`kill`).

9.15 Installer une nouvelle imprimante

Maintenant que la machine fonctionne, voyons comment imprimer. Les imprimantes sont gerées par le programme `lpc` et par le démon `lpd`. Le programme `lpr` place une copie des fichiers dans un répertoire d'attente (de *spool*), dans lequelle les fichiers se voient attribuer un numéro de Job, suivant leur ordre d'arrivée, qui déterminera l'ordre dans lequel ils seront imprimés. En tout, il y place au moins deux fichier :

- le premier commencant par **cf**, suivit du numéro de Job, et contenant des informations de contrôle relatives à la tâche d'impression, comme le nom de l'utilisateur qui a démarré la tâche d'impression.
- le second commencant par **df**, suivit du numéro de Job, et contenant les données à imprimer.

Le démon `lpd` scrute réguliérement ce répertoire de spool et, lorsqu'il trouve des fichiers, crée une copie de lui même qu'il place dans le répertoire de spool de l'imprimante s'il n'en existe pas déjà une (afin de pouvoir continuer à scruter les autres répertoire de spool, au cas où une autre tâche serait lancée sur une autre imprimante), consulte le fichier de configuration des imprimantes `/etc/printcap` et envoit le résultat vers l'imprimante.

Il faut d'abord s'assurer que le démon d'impression est bien lancé (normalement il l'est au démarage) :

```
ps -ax | grep lpd
```

Le résultat devrait ressembler a ca (si ce n'est pas le cas, il faut le lancer : `/usr/sbin/lpd`) :

```
  102  ?  S      0:00 /usr/sbin/lpd
```

Il faut également s'assurer que le câble est bien connecté au port (série ou parallèlle) :

```
dmesg | grep lp
```

Le résultat devrait ressembler a ca (pour un port parallèle) :

```
lp1 at 0x0378, (polling)
```

9.15.1 Un exemple simple de configuration

Le fichier de configuration d'une imprimante est `/etc/printcap`. Voici un exemple simple de configuration d'imprimante :

```
#---------------------------------------#
# simple, un modele simple pour printcap #
#---------------------------------------#

# cf. /var/lib/apsfilter

# sd nom du repertoire de la file d'attente
#
# lp nom du port auquel l'imprimante est connecte
# /dev/lp1 pour LPT1
# /dev/ttyS0 pour COM1
# /dev/ttyS1 pour COM2
# /dev/null pour des tests sans gacher de papier...
#
# if charge un filtre
#
# lf specifie le fichier journal d'erreurs
# dans cet exemple, il faudra creer le fichier /usr/spool/lp1/simple.log
# et lui affecter le groupe lp et les droits 664 :
# chgrp lp /usr/spool/lp1/simple.log
# chmod 664 /usr/spool/lp1/simple.log
#
# mx#0 desactive des limites de taille de fichiers a imprimer
#
# sh desactive la page d'en tete
#
# sf desactive le saut de page
#
```

```
simple|simple-printer|Un simple printer qui ecrit dans un fichier:\
:sd=/usr/spool/lp1:\
:lp=/dev/null:\
:if=/root/simple-if.tcl:\
:lf=/usr/spool/lp1/simple-log:\
:mx#0:\
:sh:\
:sf:
```

Pour économiser du papier, cette imprimante imprimera sur le périphérique `/dev/null` (corbeille, ou trou noir). Une fois installée et les essais terminés, on n'aura qu'à remplacer la ligne `:lp=/dev/null:\` par `:lp=/dev/lp1:\`.

La déclaration pour chaque imprimante doit tenir sur une seule ligne, le charactère `\` permet de faire tenir les différents champs sur plusieurs lignes (il ne doivent cependant pas être suivis par un espace ou une tabulation).

Le filtre utilisé est un programme Tcl qui écrit la date courante suivi par l'entrée standard dans le fichier **/tmp/simple.out** :

```
#!/usr/bin/tclsh

set outfile [open /tmp/simple.out w]

puts $outfile "-------------------------------------------------"
flush $outfile
exec date >@ $outfile
puts $outfile "-------------------------------------------------"

while { [gets stdin line] != -1} {
    puts $outfile $line
}
close $outfile
```

Le journal des messages d'erreur peut être crée comme suit (on fabrique un fichier vide, puis on lui attribue les droits qu'il faut) :

```
cd /usr/spool/lpd
touch simple-log
chgrp lp simple-log
chmod ug=rw,o=r simple-log
```

Pour utiliser une telle imprimante, il faut éditer le fichier d'initialisation `/etc/rc.d/rc.local`, et rajouter les lignes suivantes :

```
echo "Configuration de l'imprimante simple..."
/usr/sbin/lpc up simple
```

et taper en ligne de commande :

`lpr -Psimple` *fichier*

On peut par ailleurs ajouter un répertoire spécial pour cette imprimante dans la file d'attente (là où sont mis les fichiers en attendant qu'ils puissent être imprimés), comme `/usr/spool/lpd/simple`, a condition de le préciser en remplacant dans le fichier `/etc/printcap` la ligne indiquant le répertoire de la file d'attente :

```
sd=/var/spool/lpd/simple:\
```

Voici un exemple d'utilisation. Il faut d'abord relancer `lpd` :

```
lpc restart all
```

La présence du fichier `/tmp/simple.out` est la preuve que le système est bien configuré (cette impression peut se faire sans qu'une imprimante ne soit réelement connectée, elle permet juste de vérifier que le fichier `/etc/printcap` ainsi que le démon `lpd` et les commandes telles que `lpr`, `lpq`, `lprm` et `lpc` fonctionnent) :

```
ls -l | lpr -Psimple
```

Le fichier `simple.out` doit alors ressembler à ca (sa présence prouve que tout est correctement configuré) :

```
------------------------------------------------
Sat Mar 20 14:42:33 CET 1999
------------------------------------------------
total 37
-rw-r--r--   1 mathieu  users          8 Jan 31 18:16 fictest
-rwxr--r--   1 mathieu  users        313 Jan 31 18:53 lc.tcl
-rwxr--r--   1 mathieu  users        215 Sep 30 22:16 lc.tcl~
-rwxr--r--   1 mathieu  users        378 Jan 31 18:25 questions.tcl
-rwxr--r--   1 mathieu  users         74 Oct  3 23:40 questions.tcl~
-rwxr-xr-x   1 root     root         647 Dec  2 16:02 simple-if.tcl
-rwxr-xr-x   1 root     root         497 Dec  2 15:53 simple-if.tcl~
-rwxr--r--   1 mathieu  users       6990 Jan 31 19:04 simple.tcl
-rwxr--r--   1 mathieu  users       7064 Jan 31 18:42 simple.tcl~
-rwxr--r--   1 mathieu  users       3153 Feb  2 07:42 tkdraw
-rwxr--r--   1 mathieu  users       3153 Feb  2 07:33 tkdraw~
-rwxr--r--   1 mathieu  users        937 Jan 31 20:53 tkedit
-rwxr--r--   1 mathieu  users        399 Oct  6 19:42 tkedit~
```

```
-rwxr--r--   1 mathieu  users        2796 Feb  1 19:52 tkmin
-rwxr--r--   1 mathieu  users         777 Oct  6 20:16 tkmin~
-rwxr--r--   1 mathieu  users         442 Jan 31 19:22 tksalut
-rwxr--r--   1 mathieu  users         106 Oct  5 23:30 tksalut~
```

Sans spécification de nom d'imprimante, celle utilisée par défaut sera celle contenue dans la variable d'environnement **$PRINTER** (**lp** en général). Pour définir l'imprimante lp, il suffit de la déclarer comme suit dans le fichier `/etc/printcap` :

```
lp|Nom de l'imprimante:\
```

ou

```
Nom de l'imprimante|lp:\
```

On peut remplacer `Nom de l'imprimante` par le vrai nom de l'imprimante, ou par tout autre nom.

Si ca ne marche pas avec une vraie imprimante, il existe une commande, `lptest`, qui produit une sortie ASCII. On peut rediriger cette sortie vers le périphérique de l'imprimante pour voir si il s'agit d'un problème de configuration d'imprimante :

```
lptest > /dev/lp1
```

ou, pour avoir 6 lignes de 35 caractères seulement :

```
lptest 35 6 > /dev/lp1
```

Le résultat ressemble à ceci :

```
!"#$%&'()*+,-./0123456789:;<=>?@ABC
"#$%&'()*+,-./0123456789:;<=>?@ABCD
#$%&'()*+,-./0123456789:;<=>?@ABCDE
$%&'()*+,-./0123456789:;<=>?@ABCDEF
%&'()*+,-./0123456789:;<=>?@ABCDEFG
&'()*+,-./0123456789:;<=>?@ABCDEFGH
```

On peut aussi tester simplement l'une des commandes suivantes :

```
lptest | lpr
lpr test.txt
lptest > /dev/lp1
cat test.txt > /dev/lp1
```

Les fichiers suivants recueillent les messages d'erreur de l'imprimante, il seront donc à consulter si l'impression ne marche pas (l'exemple est prit pour l'imprimante *simple*) :

```
/var/spool/lpd/simple/status
/var/log/lpd-errs
/sbin/pac
```

9.15.2 Filtres d'impression

Ghostscript permet d'afficher et d'imprimer des fichiers PostScript entre autre. Pour l'appeller, il faut taper `gs`, puis entrer *help* ou une des commandes possibles dans le mode interactif :

```
Aladdin Ghostscript 4.03 (1996-9-23)
Copyright (C) 1996 Aladdin Enterprises, Menlo Park, CA.
All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>
```

Pour avoir la liste des périphériques que Ghostscript reconnaît, taper :

```
devicenames ==
```

La liste s'affiche alors :

```
[/tiffg4 /pnmraw /oce9050 /la50 /ap3250 /ljet3d /pjxl /miff24
/png16m /mgrgray4 /tiffg3 /pgnmraw /m8510 /lips3 /bjc600 /ljet2p
/pj /t4693d8 /png16 /mgrmono /faxg4 /pgmraw /imagen /bjc800 /bj200
/hpdj /dnj650c /cdj550 /pngmono /bitrgb /faxg3 /pbmraw /cp50 /t4693d2
/lp2563 /iwlo /djet500 /cgm24 /pcx256 /psmono /dfaxhigh /xes /lj250
/cdjmono /pdfwrite /lj4dith /cdj850 /cgmmono /pcxgray /tifflzw /pkm
/r4081 /la75plus /cdeskjet /st800 /eps9high /appledmp /bmp256 /mgr8
/tiff12nc /ppm /oki182 /la70 /epson /ljet4 /x11cmyk /x11 /bmpmono
/mgrgray8 /tiffg32d /pnm /necp6 /declj250 /tek4696 /ljet3 /pjetxl
/png256 /mgrgray2 /tiffcrle /pgnm /jetp3852 /nullpage /lbp8 /laserjet
/paintjet /t4693d4 /pnggray /bitcmyk /faxg32d /pgm /ibmpro /iwlq /bj10e
/djet500c /cif /pcx24b /bit /dfaxlow /pbm /ccr /cdj500 /ljetplus /iwhi
/deskjet /cgm8 /pcx16 /tiffpack /pkmraw /sj48 /ln03 /cdjcolor /epsonc
/x11mono /bmp16m /pcxmono /tiff24nc /ppmraw /okiibm /la75 /stcolor
/eps9mid /pjxl300 /x11alpha /bmp16 /mgr4]
```

et pour quitter :

```
GS>quit
```

Pour imprimer en utilisant le péripherique *epson*, par exemple, appeller Ghostscript comme suit :

```
gs -sDEVICE=epson
```

ou initialiser la variable d'environnement **$GS_DEVICE** :

```
export GS_DEVICE=epson
```

Plusieurs filtres peuvent être utilisés pour imprimer (**nenscript, APS-filters. . .**). Voici un exemple de filtrage (pour HP-500, 510, 520,. . .).

```
#!/bin/sh
# Script d'impression pour imprimante HP-500 Noir et Blanc
#
# Eric.Dumas@freenix.org
#
# Version 2.0
#
# 5/01/95   (ED) : Ajout du format dvi ;
# 27/10/95  (ED) : Conversion des fichiers textes en fichier PostScript ;
# 01/11/95  (ED) : Un peu de mnage ;
# 12/08/96  (ED) : modification pour gs
# 04/12/96  (ED) : quelques corrections et ajouts.

TmpDir=/tmp
TmpFile=$TmpDir/deskjet.$$

# Utilisateur  prevenir en cas d'erreur
NOTIFY=lp-owner

# Programmes
CAT=/bin/cat
DVIPS=/usr/TeX/bin/dvips
PGS=/usr/bin/gs
AIIPS=/usr/local/bin/a2ps

# Chemins d'accs pour GS
GS_LIB=/usr/lib/ghostscript:/usr/lib/ghostscript/psfonts:\
        /usr/lib/ghostscript/Type1:/usr/lib/ghostscript/fonts

# C'est parti
$CAT - > $TmpFile

echo -ne '\033E'

set -- `file $TmpFile`
```

```
shift

FileType=$*

# Transformation du PostScript en format Deskjet500
# rsolution  300x300 - format a4
GS="$PGS -I$GS_LIB -q -sDEVICE=djet500 -r300x300 -sPAPERSIZE=a4 -dNOPAUSE\
-sOutputFile=- - /usr/lib/ghostscript/quit.ps  || echo -ne '\033&10H'"

case $FileType in

       *DVI*) # Fichier DVI
         $DVIPS -t a4 $TmpFile \
           -f | $GS
        ;;

       *PostScript*)    # Impression de fichiers Postscript
         $CAT $TmpFile | $GS
        ;;

        *text*|*script*)
         $CAT $TmpFile | $AIIPS -nP -r -8 | $GS
        ;;

        *data*)
          echo -ne '\033&;k0G'  # C'est un et commercial sans ; !
          /bin/cat $TmpFile
         ;;

        *)
          echo "Deskjet: Unknow filetype $FileType" >> /dev/console
          echo "Deskjet: $TmpFile Unknow filetype $FileType" | mail $NOTIFY
        ;;

       *DVI*) # Fichier DVI
         $DVIPS -t a4 $TmpFile \
           -f | $GS
        ;;

       *PostScript*)    # Impression de fichiers Postscript
         $CAT $TmpFile | $GS
```

```
        ;;

        *text*|*script*)
         $CAT $TmpFile | $AIIPS -nP -r -8 | $GS
        ;;

        *data*)
          echo -ne '\033&;k0G'  # C'est un et commercial sans ; !
          /bin/cat $TmpFile
         ;;

        *)
          echo "Deskjet: Unknow filetype $FileType" >> /dev/console
          echo "Deskjet: $TmpFile Unknow filetype $FileType" | mail $NOTIFY
        ;;
esac

/bin/rm -f $TmpFile
echo -ne '\033E'
```

Le filtre joint au paquet de gestion d'impression se nomme `/usr/sbin/lpf`. Il est peu pratique, mais d'autres filtres sont sûrement disponibles sur le système. Pour en avoir la liste, taper :

```
man -k filter
```

Pour spécifier un filtre, il faut modifier la valeur de la variable *if* dans le fichier de configuration `/etc/printcap`, puis relancer `lpd` :

```
lpc restart all
```

Il existe deux filtres fréquemment utilisés car très puissants :

– le filtre **nenscript** assure entre autres la conversion de texte simple vers le PostScript. Il est situé en général dans le répertoire `/usr/bin`. Il imprime sur l'imprimante spécifiée par, dans l'ordre de préférence :
  – la variable d'environnement **$NENSCRIPT**.
  – la variable d'environnement **$PRINTER**, si la précédente variable d'environnement n'est pas initialisée.
  – l'imprimante choisie par défaut par `lpr` si aucune des deux précédentes variables d'environnement n'est initialisée.

  L'option *-Z* empêche la conversion si le fichier ne commence pas par "`%!`" (la plupart des fichiers PostScript commencent par "`%!`"). Bien que ce filtre ne soit pas fiable à 100%, `nenscript` devrait convenir dans la plupart des cas. Les options à passer comme *-Z* par exemple pouront être spécifiées par la variable d'environnement **$NENSCRIPT**. Voici un exemple d'utilisation :

```
nenscript -B -L66 -N -Pepson memo.txt
```

L'option *-B* supprime l'en-tête, *-N* numérote les lignes du fichier, *-L66* fixe le nombre de lignes par page à 66, et *-Pepson* imprime sur l'imprimante "epson". Si aucune imprimante n'est spécifiée, l'impression se fera sur l'imprimante standard : il n'y a pas besoin de mettre en place de tube, sauf si l'option *-p* est spécifiée.

La variable d'environnement **$NENSCRIPT** peut contenir les options standard :

```
export NENSCRIPT=''-B -L66 -Pepson''
```

Pour convertir un fichier texte en PostScript, il faut fournir l'option *-p*, qui représente la sortie standard :

```
nenscript -2 -fCourier6 -TA4 -pmemo.ps memo.txt
```

L'option *-2* imprimera sur deux colonnes, *-f* précise la police (ici Courier 6 pt, 10 pt par défaut), *-TA4* représente le format de la page. Le document ASCII `memo.txt` sera donc convertit en fichier PostScript `memo.ps`.

Pour imprimer un document sans enrichissements, utiliser la commande `col` :

```
man nenscript | col -b | nenscript
```

La page de manuel se voit supprimer ses caractères de mise en valeur, puis le filtre `nenscript` imprime le tout.

– un *filtre magique* comme **APSfilter**. Pour l'installer, décompresser l'archive dans le répertoire `/usr/lib/apsfilter/` :

```
gunzip apsfilter*gz
tar -xvf apsfilter*gz -C /usr/lib
```

Il faut ensuite spécifier ou corriger dans les fichiers `Global.sh` ou `apsfilter` les variables et les chemins suivants, s'il ne sont pas correctes :

```
LP_OWNER=root
LP_GROUP=lp
SPOOL=/var/spool/lpd
MAGIC=etc/magic
```

Bien lire la documentation pour plus d'information. Ne pas oublier de relancer le démon d'impression :

```
lpc restart all
```

Dans tous les cas, un filtre général comme présenté plus haut pourra invoquer ou non suivant les cas différents filtres comme `gs` ou `nenscript`.

9.15.3 Contenu du dispositif d'impression

Le répertoire `/var/spool` est souvent considéré comme répertoire par défaut de spool. Certains filtres et utilitaires considéreront que c'est le répertoire `/usr/spool`, il faut donc faire un lien vers ce dernier. Le répertoire par défaut de spool d'impression est `/var/spool/lpd`. Chaque imprimante pourra avoir son propre répertoire de spool, référencé dans le fichier de configuration `/etc/printcap`[8], comme le répertoire `/var/spool/lpd/simple` pour l'imprimante **simple**, par exemple.

Le tableau 9.13 résume l'ensemble des fichiers, leurs propriétés et une courte description est donnée.

Pour mettre en place le répertoire `/var/spool/lpd`, ou tout autre répertoire d'imprimante, il faut le créer au moyen de la commande `mkdir`, puis affecter les droits usuels avec `chmod`[9] :

```
mkdir /var/spool/lpd
chmod 755 /var/spool/lpd
chmod +s /var/spool/lpd
chgrp lp/var/spool/lpd
```

Reste à créer un fichier *.seq* dans chaque répertoire d'impression :

```
touch .seq
chown root.lp .seq
```

[8]Pour faciliter son identification, le nom du répertoire de spool doit être le premier nom référencé dans l'entrée du fichier */etc/printcap*.

[9]On initialise aussi le *sticky bit* afin de permettre à **lp** de prendre les droits de **root**.

Fichier	**Propriétés**	**Description**
/dev/ttyS1	`crwsr---- root lp`	Périphérique d'impression série typique.
/dev/lp1	`crws------ root lp`	Périphérique d'impression parallèle typique.
/usr/bin/lpr	`-rwsrwsr-x root lp`	Recoit les fichiers à imprimer et les stockes dans le répertoire de spool.
/usr/bin/lpq	`-rwsrwsr-x root lp`	Affiche les noms des fichiers en attente ainsi que l'état de la file d'attente.
/usr/bin/lprm	`-rwsrwsr-- root lp`	Supprime les tâches d'impression du spool.
/usr/bin/tunelp	`-rwsr-sr-- root lp`	Teste les services d'impression à des fins d'optimisation.
/usr/bin/lptest	`-rwxr-xr-x root root`	Transmet à l'imprimante un fichier test ASCII.
/usr/sbin/lpd	`-rwsr-s--- root lp`	Démon qui organise l'impression en fonction du fichier *printcap* et des informations fournies par **lpr**.
/usr/sbin/lpc	`-rwsrwsr-x root lp`	Gestion du spool d'impression.
/usr/sbin/lpf	`-rwxr-xr-x root lp`	Filtre d'impression rudimentaire pour fichier texte.
/usr/sbin/pac	`-rwxr--r-- root root`	Utilitaire rendant compte de l'activité de l'imprimante et de l'état des requêtes de l'utilisateur spécifié.
/var/spool	`drwxr-sr-x root daemon`	Répertoire consacré aux fichiers temporaires.
/var/spool/lpd	`drws--s--x root lp`	Chemin standard du spool d'impression.
*/var/spool/lpd/**	`drwxr-sr-x root lp`	Sous-répertoires de spool des imprimantes définies.
/var/spool/lpd//filter*	`-rwxr-xr-x root lp`	Filtre générés pour chaque spool d'impression.
/var/spool/lpd/lpd.lock	`-rw-rw---- root lp`	Fichier de verrouillage de la file gérée par **lpd**.
/var/spool/lpd//.seq*	`-rw-rw---- lp lp`	Fichier généré par **lpd** pour la gestion des fichiers en attente.

Fichier	**Propriétés**	**Description**
/var/spool/lpd//lock*	`-rw------- root lp`	Vérrou crée par **lpd** pour interdire l'expédition du fichier suivant tant que l'imprimante n'est pas prête.
/var/spool/lpd//status*	`-rw------- lp lp`	Fichier contenant les dernières informations sur l'état de la file.
/var/log/lp-acct	`-rw------- root root`	Fichier de compatibilité, utilisé par l'utilitaire **pac**.
/var/log/lpd-errs	`-rw-rw-r-- root lp`	Fichier standard de compte rendu des erreurs **lpd**.

TAB. 9.13 – Propriétés et description des principaux fichiers du dispositif d'impression.

9.15.4 Gestion des services d'impression avec lpc

La commande `lpq` donne la liste des fichiers à imprimer, avec leur état, leur propriétaire, leur numéro de Job. . . :

```
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
active mathieu    27   /tmp/sort.tmp         313482 bytes
1st    mathieu    28   /tmp/Disquette        1024 bytes
2nd    mathieu    29   /tmp/x.out            4647 bytes
3rd    mathieu    30   /tmp/y.out            1652 bytes
4th    mathieu    31   /tmp/SuperProbe.txt   933 bytes
```

La commande `lprm` permet d'effacer de la queue un fichier. Pour effacer le fichier */tmp/x.out* dans l'exemple précédent, auquel est associé le numéro de Job 29, taper :

```
lprm 29
```

On peut vérifier que la requête a bien été effacée avec la commande `lpq` :

```
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
active mathieu    27   /tmp/sort.tmp         313482 bytes
1st    mathieu    28   /tmp/Disquette        1024 bytes
2nd    mathieu    30   /tmp/y.out            1652 bytes
3rd    mathieu    31   /tmp/SuperProbe.txt   933 bytes
```

et pour effacer tous les fichiers de la file d'attente, entrer :

```
lprm -
```

On peut également effacer tous les fichiers de la file d'attente d'une imprimante, spécifiée après l'option *-P* :

```
lprm -Pada
```

ou d'un utilisateur :

```
lprm root
```

L'utilitaire `lpc` permet de gérer les files d'impression. Pour obtenir des informations sur l'état de l'ensemble des imprimantes et des utilisateurs, taper `lpc status` :

```
lp:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
simple:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
```

ou, en cas d'impression en cours :

```
lp:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        lp is ready and printing
simple:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
```

et pour avoir des informations sur une imprimante particulière, taper `lpc status simple`, par exemple :

```
simple:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
```

On peut aussi lancer `lpc` en mode commande en invoquant seulement `lpc` :

```
lpc>
```

Il faut alors lancer une des commandes spécifiques à `lpc`. Par exemple, pour avoir de la liste des commandes disponibles[10] :

```
lpc> ?
Commands may be abbreviated.  Commands are:

abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc>
```

et avoir plus de précisions sur une commande :

```
lpc> help restart
restart         kill (if possible) and restart a spooling daemon
lpc>
```

Pour mettre une tache en tête de la queue, utiliser la commande `lpc` avec l'option *topq* :

```
topq lp 30 mathieu
```

Si on n'indique aucun numéro de Job, tous les fichiers à imprimer de l'utilisateur spécifié seront déplacés en tête de queue. On peut vérifier que la requête a bien été déplacée avec la commande `lpq` :

```
lp is ready and printing
Rank   Owner      Job  Files                  Total Size
1st    mathieu    30   /tmp/y.out             1652 bytes
1st    mathieu    27   /tmp/sort.tmp          313482 bytes
2nd    mathieu    28   /tmp/Disquette         1024 bytes
3rd    mathieu    31   /tmp/SuperProbe.txt    933 bytes
```

On peut autoriser ou non un utilisateur à imprimer avec une imprimante particulière :

- `lpc enable simple` **autorise** l'impression sur l'imprimante *simple*.
- `lpc disable simple` **interdit** l'impression sur l'imprimante *simple*.

L'option *stop* permet de **désactiver** une impression. L'imprimante termine la tâche en cours et les tâches en attente où à venir demeurent dans la queue jusquà ce que l'impression soit **réactivée** avec l'option *start*. Avec l' option *abort*, la tâche en cours est interrompue, et ne sera relancée que

[10] On aurait tout aussi bien pu entrer *help*. . .

lorsque la queue sera réactivée. L'option *down* combine les actions des commandes *disable* et *stop*. L'option *up* fait l'inverse (annule *stop*), et combine les actions des commandes *enable* et *start*.

Bien entendu, il existe également les options *status* et *restart* déjà évoquées, qui sont accessibles à n'importe quel utilisateur et qui permettent respectivement d'avoir des informations sur l'état de l'ensemble des imprimantes et de relancer le démon d'impression. Noter que les options *stop* et *down* peuvent prendre en argument une notification à l'ensemble des utilisateurs.

L'utilitaire `tunelp` permet de configurer plusieurs paramètres pour le périphérique `lp`. Par exemple, pour requérir l'annulation notifiée d'une tâche en cas d'erreur de l'imprimante :

```
tunelp /dev/lp1 -a on
```

et si l'imprimante parallèle est reliée à un port qui dispose d'une ligne IRQ (IRQ 7 dans l'exemple qui suit), on peut accélérer la transmission :

```
tunelp /dev/lp1 -i7
```

le message suivant s'affiche alors :

```
/dev/lp1 using IRQ 7
```

et pour annuler (en le réinitialisant) :

```
tunelp /dev/lp1 -r -i0
```

le message suivant s'affiche alors :

```
/dev/lp1 using polling
```

et pour savoir quel port est utilisé :

```
tunelp /dev/lp1 -q on
```

le message suivant s'affiche alors :

```
/dev/lp1 using polling
```

ou

```
/dev/lp1 using IRQ 7
```

Pour accélerer l'impression, on peut choisir le nombre de tentatives pour transmettre un caractère avec l'option *-c* (250 tentatives par défaut), et le temps de pause après la série d'essais avec l'option *-t* (0,1 par défaut, soit $0,1 \times 0,01 = 0,001$ seconde, la valeur numérique spécifiée étant un multiple de 0,01 seconde) :

```
tunelp /dev/lp1 -c10 -t1
```

Si l'imprimante se bloque lorsqu'elle recoit des fichiers graphiques, il faut augmenter le temps de pause qui sépare les essais successifs (2 secondes dans l'exemple qui suit) et le nombre de boucles de temporisation entre deux transferts d'octets avec l'option *-w* :

```
tunelp /dev/lp1 -t200 -w5
```

9.15.5 Informations complémentaires concernant l'impression

Cette section présente des informations qui ne seront pas toujours utiles, mais qui peuvent améliorer l'impression ou permettre d'imprimer dans des conditions particulières.

- La commande `lpr` accepte entre autres l'option *-#N*, qui permet d'imprimer N fois les données spécifiées, et l'option *-d* qui permet de supprimer le fichier dès que la demande d'impression est éxécutée.
- La commande `pr` permet d'**ajouter des informations à chaque page** du fichier imprimé, comme un en-tête avec l'option *-h*, ou permet certains réglages comme régler le nombre de lignes par pages avec l'option *-l* :

  ```
  pr -h "Fichier bidon" -l60 /tmp/x.out | lpr -Psimple
  ```

  Le tableau 9.14 présente les autres options de la commande `pr`.

Option	Signification
-n	Numérote chaque ligne, avec une tabulation après le numéro. L'option `-n:` numérote chaque ligne avec le caractère "`:`" après le numéro.
-3	Affiche sur 3 colonnes.
-w80	Fixe le nombre de caractères à **80** par ligne (72 par défaut).
-l60	Fixe le nombre de lignes à **60** par page (66 par défaut).
-h*texte*	Insère un en-tête avec le texte spécifié.
-o10	Définit 10 caractères en retrait par rapport à la marge de gauche.

TAB. 9.14 – Principales options de **pr**.

- S'il n'y a **pas beaucoup de place sur le disque**, la copie des gros fichiers lors de l'impression peut poser problème. On peut faire créer un lien du répertoire de spool vers ce fichier avec l'option *-s* :

```
lpr -Psimple -s /tmp/x.out
```

On ne peut cependant pas éditer ou supprimer ce fichier tant que l'impression n'est pas terminée. Si l'imprimante est distante (en réseau), le fichier sera quand même copié dans le répertoire de spool.

– Pour **imprimer en réseau**, il faut rajouter une entrée au fichier de configuration `/etc/printcap` avec le champ *lp=* vide, le champ *rm=* contenant le nom du système distant, et le champ *rp=* le nom de l'imprimante distante :

```
# Imprimante simplet.
# Imprime sur l'imprimante "simple" connectee sur "cameleon"

simplet|simple-printer|Une simple imprimante en reseau:\
        :sd=/usr/spool/lpd:\
        :lp=:\
        :rm=cameleon:\
        :rp=simple:\
        :mx#0:\
        :sh:\
        :sf:
```

Il faut ensuite éditer le fichier `/etc/hosts.lpd` du système distant (celui sur lequel est connecté *physiquement* l'imprimante, **cameleon** dans l'exemple précédent), et rajouter une ligne contenant le nom du système à partir duquel on veut imprimer (celui d'où on lance l'impression, et qui peut très bien n'avoir aucune imprimante physiquement connectée). Par exemple, si on lance sur la machine **zecastor** l'ordre de lancer l'impression sur l'imprimante connectée *physiquement* sur **cameleon**, le fichier `/etc/hosts.lpd` de **cameleon** contiendra la ligne :

```
zecastor
```

Il faut bien sûr que le fichier `/etc/hosts` du système distant (**cameleon** dans l'exemple précédent) contienne l'adresse IP du système à partir duquel on veut imprimer (**zecastor** dans l'exemple précédent). Pour l'imprimante particulière *simple*, un fichier `/tmp/simple.out` sur le système distant (**cameleon** dans l'exemple précédent) doit apparaître et contenir le résultat de l'impression suivante :

```
ls -l | lpr -Psimplet
```

Le champ *rg=* permet de spécifier quels groupes sont autorisés à imprimer sur l'imprimante distante, et le champ *rs=* impose que l'utilisateur ait un compte sur le système distant pour imprimer sur l'imprimante distante.

Si un filtre est spécifié sur le système local, il ne sera pas pris en compte par le système distant (le fichier sera transféré sur le système distant, et c'est le processus `lpd` du système distant qui se charge d'imprimer le fichier). Seuls les filtres spécifiés dans le fichier `/etc/printcap` du système distant seront pris en compte. On peut toutefois filtrer le fichier avant de l'envoyer en créant un script qui envoit la sortie à l'imprimante distante :

```
#!/bin/sh
{
#
# Commandes qu'il y aurait dans le filtre...
#
} | lpr -Psimplet -h -l
```

– Pour **éviter l'effet d'escalier** (le caractère saut de ligne **LF** en fin de ligne sous UNIX est remplacé par le caractère retour chariot **CR** plus saut de ligne **LF**), lorsqu'on imprime des fichiers MS-DOS sous UNIX :

```
Premiere ligne
              Seconde ligne
                           Troisieme ligne
                                          Quatrieme ligne
```

il faut essayer de configurer l'imprimante (si possible), ou insérer un filtre en entrée du type suivant :

```
#!/bin/sh

# Envoi de la commande (\033 correspond a Esc en octal)

echo -ne \\033\&k2G

# Lit stdin et redirige sur stdout

cat

# Envoie le caractere nouvelle page a la fin du fichier

echo -ne \\f
```

ou le script suivant (équivalent, mais qui permet d'inhiber les CR par la commande `lpr -l fichier.txt`).

```
!/bin/sh
```

```
# Lit stdin et redirige sur stdout

if [ "$1" = -l ]; then
    cat
else
# Envoie le caractere nouvelle page a la fin du fichier (^M=CR)
    sed -e s/$/^M
fi

echo -ne \\f
```

Il ne reste plus qu'a rendre ce script exécutable, et à l'invoquer dans le fichier `/etc/printcap` par le champ *if=*.
- **Les fichiers graphiques sont tronqués**. Il faut alors spécifier dans le fichier `/etc/printcap` la ligne **mx#0** (sinon tout fichier dépassant 1000 blocs sera automatiquement tronqué).
- **Le texte comporte des signes "^H"**. Il s'agit d'un texte "enrichit" (qui contient des éléments en gras, tels qu'ils seraient affichés par une commande comme `less`). Il faut alors filtrer le fichier :

```
man printcap | col -b | lpr
```

Et pour remplacer les caractères soulignés par des éléments en gras :

```
man printcap $\mid$ colcrt lpr
```

9.16 Réglage des préférences sur les terminaux

Le programme `setterm` permet de modifier les caractéristiques de chaque terminal virtuel. Par exemple, pour changer les couleurs :

```
setterm -foreground white -background blue
```

et pour que les modifications ne soient pas changées par les applications (certaines veulent assigner les couleurs par défaut une fois leur tâche terminée) :

```
setterm -store
```

Le fichier `/etc/DIR_COLORS` permet de définir les couleurs d'affichage.

9.17 Gestion des traces du système avec syslogd

Le démon `syslogd` permet d'enregistrer diverses activités du système, comme les messages de déboguage ou les avertissements affichés par le noyau. Son fichier de configuration, `/etc/syslog.conf`, permet de spécifier l'endroit où les informations doivent apparaître. Par exemple :

```
kern.warn;*.err;authpriv.none   /dev/tty10
*.emerg                         *
mail.*                          -/var/log/mail
*.warn                          /var/log/warn
```

La syntaxe de ce fichier est la suivante :

dispositif.niveau ; dispositif.niveau...

où *dispositif* représente l'application ou le service d'où le message émane (ce peut être le démon courrier *mail*, le noyau *kern*, les programmes utilisateurs *user*, les programmes d'identification *auth*, tels que `login` ou `su`), et *niveau* la gravitée du message :
- `debug` pour déboguage.
- `info` pour information.
- `notice` pour notification.
- `warning` pour avertissement.
- `err` pour erreur.
- `crit` pour critique.
- `alert` pour "panique à bord"...
- `emerg` pour "rien n'va plus"...

Les messages envoyés à `/dev/console` seront envoyés à la console virtuelle courante, où une fenêtre **xterm** lancée avec l'option *-C*.

On peut effacer les fichiers de compte rendu si ils devient trop imposants, mais certains systèmes risquent de se plaindre. Il faut alors recréer un fichier vide, à l'aide de la commande `touch`, par exemple[11].

Pour faire en sorte que les modifications du fichier de configuration, `/etc/syslog.conf`, soient prisent en compte, il faut que `syslogd` recoive le signal **-1**, appelé `SIGHUP` :

```
kill -HUP `cat < /etc/syslog.pid`
```

Le fichier `/etc/syslog.pid` contient le numéro de processus de `syslogd`. Les fichiers suivants sont rattachés à `syslogd` :

[11] Cette commande n'est cependant pas destinée à ca au départ.

- `/var/adm/wtmp` contient la date et la durée des sessions de chaque utilisateur du système (utilisé par la commande `last`).
- `/var/run/utmp` contient des informations sur les utilisateurs actuellement connectés sur la machine (utilisé par les commandes `who`, `w` ou `finger`).
- `/var/log/lastlog` est lui aussi utilisé par divers programmes, dont `finger`.

9.18 Réparation d'un système de fichiers

Il existe deux commandes qui permettent de réparer un système de fichiers : `/sbin/e2fsck` qui resoud la plupart des problèmes, et `/sbin/debugfs` qui pourra être utilisé si la commande précedente n'a pu résoudre le problème. Pour ces deux commandes, le système de fichier (partition, disquette...) **ne doit pas être monté**.

Etant donné la complexité de la réparation d'un système de fichiers, il est conseillé de se "faire la main" sur un périphérique pour lequel une erreur n'aurait pas beaucoup de conséquences, comme une disquette par exemple.

Le tableau 9.15 présente les principales options de `/sbin/e2fsck`.

Option	**Signification**
-y	répond à toutes les questions par **oui**.
-n	répond à toutes les questions par **non**.
-p	répare les problèmes mineurs sans poser de questions.
-a	Répond automatiquement à toutes les questions posées.
-c	Marque les blocs défectueux (comme pour `mkfs`, qui sert à créer un système de fichiers).
-v	Affiche des informations plus détaillées pendant la vérification.

TAB. 9.15 – Principales options de **e2fsck**.

9.19 En cas de problème

9.19.1 Que faire ?

On peut tout réinstaller, ou tenter de réparer le problème. Dans ce cas, il faut :

1. avoir crée une **disquette d'amorcage**, qui contient une image du noyau qui se décompressera en mémoire (voir section 9.11.2 page 231).
2. avoir crée une **disquette outils** avec `mkfs`, qui contiendra quelque programmes essentiels tels que :
   - `fsck` et le ou les programmes qu'il appelle (comme `fsck.ext2`, ou `e2fsck`, par exemple).
   - `fdisk` pour gérer les partitions (taille, type...).
   - un petit éditeur comme `vi`, par exemple.
   - le programme `mount` qui sert à accéder aux données.
   - le répertoire `/dev` du système au grand complet.
   - les modules ou pilotes nécessaires (les modules sont situés en général dans le répertoire `/lib/modules/`).
   - le programme `rdev` pour monter un système de fichiers à la racine.
   - les programmes servant à restaurer les sauvegardes, comme `tar` ou `gzip`, par exemple (et éventuellement `insmod` pour les sauvagardes sur bandes, ainsi que le module `ftape.0`).
   - les commandes comme `ls, dir, ln`...

   On devrait pouvoir se passer de programmes tels que `init, getty` ou `login` (en mettant en place sur la disquette un lien entre `/etc/init` et `/sbin/bash`, ou en utilisant un paramètre LILO `init=/bin/bash`).

Noter qu'en l'absence de disquette de boot, on peut souvent s'en sortir en passant *init=/bin/sh* lors de l'amorcage. Ensuite, on peut monter la racine et éditer à la main le fichier `/etc/passwd`.

9.19.2 Quel est mon problème ?

Parmis les problèmes les plus courant, il y a :
- la **table des partitions logiques est corrompue**. On peut alors essayer de lancer le programme `fdisk` et entrer à la main les limites (*Start* et *End*) de chacune des partitions du disque. Bien sûr, il faut en avoir fait une copie papier au préalable (sur le fameux petit carnet utilisé lors de l'installation...).
- **impossible de booter**. Dans ce cas, le mieux est d'utiliser les disquettes de boot qui ont servies à l'installation. Une fois la machine

amorcée, monter la partition racine et copier le noyau sur une nouvelle disquette en lancant :

```
cat /mnt/vmlinuz > /dev/fd0
```

par exemple. Normalement, ca devrait alors fonctionner si le noyau est correct. Eventuellement, spécifier au noyau quelle partition est la racine du système de fichiers (**/dev/hda4** dans l'exemple qui suit) et lui dire de monter celle-ci en lecture seule pour permettre à `fsck` de faire son travail au moment du boot :

```
rdev /dev/fd0 /dev/hda4
rdev -R /dev/fd0 1
```

– **formatage accidentel d'une partition**. La première solution de récupération est de faire la commande suivante :

```
strings /dev/hda3 > Recup
```

Il reste ensuite à parcourir le fichier et à récupérer ce qu'il est possible de récupérer. Une seconde solution est d'utiliser ce script Perl suivant :

```
# Auteur : aubert@titan.enst-bretagne.fr
#!/usr/local/bin/perl

$maxlines = 20;

@before = ();
$syntaxe = "Syntaxe: cgrep.pl terme_a_rechercher
fichier_a_parcourir\n";

$terme = shift(@ARGV) || die $syntaxe;
$fichier = shift(@ARGV) || die $syntaxe;

open(F, $fichier) || die "Cannot read $fichier: $!\n";

# On remplit @before jusqu'a sa capacite maximales ($maxlines)
while (($_ = <F>) && (scalar(@before) < $maxlines))
{
  if (/$terme/o)
  {
    print @before;
    print $_;
    & print_next_lines;
  }
```

```
  push(@before, $_);
}
# Le tableau @before contient la bonne quantite d'elements, donc on
# passe maintenant dans une partie ou @before garde une taille
# constante
while (<F>)
{
  if (/$terme/o)
  {
    print @before;
    print $_;
    & print_next_lines;
  }
  push(@before, $_);
  shift(@before);
}

close(F);
exit 1;

sub print_next_lines
{
  for ($i = 0; $i < $maxlines; $i++)
  {
    print scalar(<F>);
  }
  exit 0;
}
```

– le ***super-block*** (qui contient les informations globales vitales) **est endommagé**, et le système de fichier ne peut donc pas être monté. Un message d'erreur du type :

```
Couldn't find valid filesystem superblock.
```

s'affiche alors. Pour le réparer, il faut d'abord avoir déterminé sur un système "sain" (partition sans problème de superblock) combien de blocks contient un "groupe de block". Si `/dev/hda2` est la partition abîmée, de type ext2 (voir section 9.8 page 226), entrer la commande suivante :

```
/sbin/dumpe2fs /dev/hda2 | less
```

et relever la ligne indiquant le nombre de blocks par groupe de block :

```
Blocks per group:         8192
```

La valeur standard est 8192. Le système fait des copies régulières du super-block au blocks $8192+1 = 8193$, $2\times8192+1 = 16385$, $3\times8192+1 = 24577$...si le nombre de blocks par groupe de block est bien 8192 (sinon, s'inspirer de ce calcul pour déterminer les endroits où résident les copies du super-block). Lancer alors la commande :

```
e2fsck -f -b 8193 /dev/hda2
```

Si ca ne fonctionne toujours pas, essayer de lancer la commande `mke2fs` avec l'option *-S*. Attention à ne pas l'oublier, sinon la partition sera reformatée ! Cette option provoque la regénération des copies du superbloc et des descripteurs du système de fichiers. Elle n'écrase pas les fichiers existants.
Il n'y a plus qu'à remonter de répertoire :

```
mount -t ext2 /dev/hda2 /mnt
```

On peut alors accéder aux fichiers de cette partition. Mettre le répertoire `/mnt` dans la variable d'environnement **$PATH** peut faciliter les opérations. Si la commande `shutdown` ne peut redémarrer le système, il faut bien **penser à démonter manuellement les partitions** avant de relancer le système.

– les **blocs** du disque dur sont **défectueux**. Il faut alors :
  1. booter sur une disquette si le problème est à la racine.
  2. lancer la commande `badblocks > BLOCKS`.
  3. lancez la commande `e2fsck -f -L BLOCKS -p -y`.

– **le mot de passe de *root* est oublié**, ou le fichier `/etc/passwd` est abîmé. Il faut alors démarrer avec la disquette de secours, se loguer en root (là, pas de mot de passe), monter la partition dans `/mnt` :

```
mount -t ext2 /dev/hda1 /mnt
```

et modifier le fichier `/mnt/etc/passwd` pour supprimer le mot de passe de **root** :

```
root::0:0:Big Brother:/:/bin/bash
```

Il ne reste plus qu'à relancer le système, se connecter en tant que **root** et appeler la commande `passwd` pour définir un nouveau mot de passe. Une autre méthode consiste à rebooter en mode dit “single-user”. Pour cela, lors du boot avec LILO, fournir LILO *linux single* (remplacer ici “linux” par le nom sous lequel LILO connaît le noyau). Un shell root va apparaître. Attention : le clavier est en *qwerty* et la partition en lecture seule. Pour y remédier, taper :

```
loadkeys /usr/lib/kbd/keytables/fr.map
mount -w -n -o remount /
```

La suite est comme dans la solution précédente pour avoir un nouveau mot de passe.

- **l'image d'une bibliothèque partagée n'est pas accessible**. Normalement, il existe un lien, comme `/lib/libc.so.4`, qui pointe vers le fichier `/lib/libc.so.4.6.18`, par exemple (le numéros indiquent les versions). Il faut alors créer ce lien, s'il n'existe pas ou qu'il pointe vers un mauvais fichier :

  ```
  cd /mnt/lib; ln -sf libc.so.4.6.18 libc.so.4
  ```

  et appeler `ldconfig`. Pour plus d'informations, voir section 8.5 page 197.

9.19.3 Récupération d'un disque

Les instructions qui suivent effectuent une sauvegarde du disque (enfin, de ce qu'il est possible de récupérer), testent la qualité du disque, recréent un système de fichiers en retirant les blocs défectueux et enfin restaurent la sauvegarde.

```
cd /fs
find . -depth -mount -print | cpio -ovB > $TAPE
cd /
tail -f /usr/adm/syslog &
umount /fs
badblocks -w -o /autre_filesystem/bb_list.tmp /dev/hda4
mke2fs -l /autre_filesystem/bb_list.tmp /dev/hda4
mount /dev/hda4 /fs
dd if=/dev/zero of=/fs/test.tmp
rm test.tmp
cd /fs
cpio -ivBmd < $TAPE
```

A partir du `mke2fs`, aucun message d'erreur ne devrait plus se produire.

Chapitre 10

Autres

10.1 mtools

Les commandes `mtools` sont la plupart de celles disponibles sous DOS, en ajoutant devant le nom de la commande la lettre *m*. Elles sont disponibles pour le super-utilisateur. La plus pratique est `mdir`, qui permet d'afficher le contenu d'une disquette (par exemple) sans avoir à la monter :

```
mdir a:
```

On peut également formater une disquette DOS, à condition de l'avoir pré-formatée avec la commande UNIX `fdformat` :

```
fdformat /dev/fd0H1440
mformat a:
```

ou

```
fdformat /dev/fd0H1440
mkfs -t ext2 /dev/fd0 1440
```

L'option *-c* permet de rechercher les blocs défectueux (à la suite de l'énoncé du type) :

```
mkfs -t ext2 -c /dev/fd0 1440
```

pour formater une disquette Linux.

Au cas où il faudrait modifier le fichier de configuration `/etc/mtools.conf` :

```
drive a: file="/dev/fd0" exclusive
drive b: file="/dev/fd1" exclusive

# 1er disque Dur
```

```
drive c: file="/dev/hda1"

# 2nd disque Dur
drive d: file="/dev/sda1"

mtools_lower_case=1
```

10.2 dosemu, un émulateur dos

Le fichier de référence est `QuickStart`, qui doit normalement se trouver dans la documentation du système. Pour installer `dosemu`, il faut d'abord s'assurer que les lignes suivantes du fichier `/etc/dosemu.conf` ne soient pas en commentaire :

```
disk { image "/var/lib/dosemu/hdimage.first" }

disk { partition "/dev/hda1" readonly } ## 1st partition on 1st IDE.

floppy { device /dev/fd0 threeinch }
```

Ensuite, sous DOS, créer une disquette système (`FORMAT A:/S`) et copier les fichiers `C:\DOS\FDISK.EXE` et `C:\DOS\SYS.COM`.

Sous UNIX, lancer les commandes suivantes :

```
dos -A
fdisk /mbr
sys c:
```

Pour quitter `dosemu`, taper :

```
c:\exitemu
```

10.3 Timezone

Timezone est un système qui permet de gérer le changement d'horaires d'été et d'hiver. La France se trouve dans une zone horaire **MET** (Medium European Time, soit GMT + 1). La zone **"MET DST"** correspond à l'heure d'été active (GMT + 2). Le répertoire `/usr/lib/zoneinfo` contient les fichiers binaires qui indiquent les régles de calcul de l'heure dans différentes zones du globe. Le fichier `time.doc` contient plus de précisions que la présente section.

Pour configurer la zone horaire, il faut copier le fichier `MET` sous le nom de `/usr/lib/zoneinfo/localtime`, puis faire un lien symbolique de ce fichier vers `/usr/lib/zoneinfo/posixrules`. Voici les commandes à passer :

```
cd /usr/lib/zoneinfo
cp MET localtime
ln -sf localtime posixrules
```

Utiliser ensuite la commande `clock` pour mettre le système a l'heure. Il y a alors deux possibilités :

- la machine est à l'heure GMT.
- la machine est à l'heure locale.

La première solution est préférable, mais MS-DOS ne gère pas correctement cette approche. L'horloge sera donc faussée sur ce système. Par contre, tous les changements d'heure, deux fois par an, seront pris en charge par Linux. Dans ce cas, il faut rajouter dans le fichier `/etc/rc.d/rc.local` la commande suivante :

```
clock -u -s
```

Linux s'ajustera sur l'horloge sauvegardée. Si la CMOS est à l'heure locale, la commande dans le fichier `/etc/rc.d/rc.local` devient `clock -s`, et les changements d'heure ne seront automatiques que si la machine est allumée au moment des changements d'heure. Il faudra manuellement réécrire la nouvelle heure dans la CMOS par la commande `clock -w`, ou mettre l'horloge à l'heure avec le setup. La commande `date` permet de vérifier la validité de l'heure : elle renvoie heure et timezone et `date -u` donne toujours l'heure en GMT : la commande `date` renverra :

```
Sat Jan  9 20:57:22 MET 1999
```

tandis que la commande `date -u` renverra :

```
Sat Jan  9 19:58:19 GMT 1999
```

En été, *MET DST* (Daylight Savings Time) serait indiqué.

10.4 Nouveaux changements d'heure

S'il s'avère qu'une année la date de changement d'heure soit décalée, et pour que Linux s'y retrouve, il est nécessaire de faire certaines modifications. La manipulation se base sur l'utilisation du programme `zic`, le **time zone compiler**. Dans un fichier appellé `europe`, il faut mettre :

```
Zone  MET     1:00    M-Eur MET%s
Link  localtime MET
Rule  M-Eur   1986  max   -     Mar   lastSun 2:00s 1:00  " DST"
Rule  M-Eur   1986  1995  -     Sep   lastSun 2:00s 0     -
Rule  M-Eur   1996  max   -     Oct   lastSun 2:00s 0
```

Et pour tout mettre en place, il faut lancer la commande :

```
zic europe
```

Pour vérifier que tout s'est bien passé, il faut, suite à la commande :

```
zdump -v MET | grep 1999
```

obtenir un résultat du type suivant :

```
MET  Sun Mar 28 00:59:59 1999 GMT = Sun Mar 28 01:59:59 1999 MET isdst=0
MET  Sun Mar 28 01:00:00 1999 GMT = Sun Mar 28 03:00:00 1999 MEST isdst=1
MET  Sun Oct 31 00:59:59 1999 GMT = Sun Oct 31 02:59:59 1999 MEST isdst=1
MET  Sun Oct 31 01:00:00 1999 GMT = Sun Oct 31 02:00:00 1999 MET isdst=0
```

10.5 Accounting et lastcomm

Le système d'accounting permet d'avoir un historique des programmes invoqués. Lancer la commande :

```
lastcomm | less
```

Ce système a tendance à prendre beaucoup de place. La solution pour résoudre ce problème est de lancer le système d'accounting de cette manire :

```
#!/bin/sh
#
# Lancement de l'accounting

accton /var/log/acct
accttrim -n 2000 /var/log/acct 2> /dev/null
```

10.6 Comment limiter le reboot en single user ?

Le problème du reboot en single user, c'est que n'importe qui peut alors réussir à passer sur la machine en **root**. Linux permet de demander le mot de passe **root**, même en single user. Pour cela, il faut récupérer les sources du programme `init` qui est lancé lors de l'amorage du système. Au début du programme `init.c`, il faut modifier la définition de la constante **SOME_USER** pour qu'elle ait la valeur **2**, recompiler `init`, et le ré-installer. Cette première solution peut toutefois s'avérer être insuffisante car une personne peut toujours booter sur un autre périphérique (en utilisant l'option *root = MonLinux*). En utilisant LILO, pas de problème ! Il suffit alors d'ajouter les lignes suivantes pour chacune des images dans le fichier `/etc/lilo.conf` :

```
password = le_mot_de_passe_en_clair
restricted
```

Il faut penser à mettre ce fichier en lecture seule pour le super-utilisateur, et aucun droit pour les autres ! Le boot normal de chaque image se passe sans problème, et sans demander le mot de passe (important si l'on veut que la machine redémarre seule en cas de pépin), mais si l'on veut passer des paramètres au noyau lors du boot, LILO demande alors le mot de passe.

10.7 Délai avant de première répétition et vitesse de répétition d'une touche

Le programme `kbdrate` permet de régler la vitesse de répétition (option *-r*) et le délai (option *-d*) d'une touche. Le programme `xset` permet de valider ou non la possibilité de répétition :

```
xset r off
```

empêche toute répétition de touche.

10.8 Clavier Francais

Une fois loggé en root, taper la commande suivante :

```
/usr/bin/loadkeys /usr/lib/kbd/keytables/fr-latin1.map
```

Pour un clavier suisse-romand, utiliser *sf-latin1.map*.

Maintenant, le clavier est francais ! Attention, cela reste temporaire. La solution la plus simple est de le rajouter dans le fichier `/etc/rc.local` avec la Slackware, mais on peut aussi utiliser le programme `/sbin/setup` ou directement `/usr/lib/setup/SeTkeymap`. Pour la Red Hat, réaliser l'opération avec le panneau de configuration `kbdconfig`.

10.9 Les accents sous bash

La lecture des lignes de saisie se fait souvent par le fonction centrale `readline`, qui consulte le fichier `.inputrc`. Il faut donc ajouter dans le fichier `.inputrc` les lignes suivantes :

```
set meta-flag on
set convert-meta off
set output-meta on
```

pour avoir la possibilité de traiter les caractères accentués.

Annexe A

Conversion des nombres en base décimale et en base héxadécimale

Un nombre en base **décimale** (base 10) s'écrit suivant les **puissances de 10** :

$$1492 = 1000 + 400 + 90 + 2$$

$$1492 = 1 \times 1000 + 4 \times 100 + 9 \times 10 + 2 \times 1$$

$$1492 = 1 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

Les coefficients devant les puissances de 10 sont compris entre 0 et 9 $(9 = 10 - 1)$.

De même, un nombre en base **héxadécimale** (base 16) s'écrit suivant les **puissances de 16**. Les coefficients devant les puissances de 16 sont compris entre 0 et 15 $(15 = 16 - 1)$. On remplace les nombres compris entre 10 et 15 par les lettres **a** à **f**. Ainsi, 3 b vaut 3 11 en décimal.

En **binaire**, on applique le même principe de comptage :

$$11 = 8 + 2 + 1$$

$$11 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$11 = 1011$$

Bibliographie

[1] M. DECORE : **article.dvi**, 3 novembre 1999

[2] M. WELSH et L. KAUFMAN : **Le système Linux**, O'REILLY, 1997

[3] M. DECORE d'après M. WELSH et L. KAUFMAN, **Vide**, 3 novembre 1999.

Index