Cours d'introduction à UNIX

Gildas PERROT
Institut de Génie Biomédical
Local A630.5.5
Tél : 340-4184
Fax : 340-4611
E-mail : perrot@grbb.polymtl.ca

École Polytechnique
Octobre 1994

1- Présentation d'UNIX

1.1- Historique

L'histoire d'UNIX débute dans les années 60 et peut être résumée de la façon suivante :

- **1966** : les laboratoires Bell (filiale d'AT&T) ont besoin pour leur usage interne, d'un système d'exploitation pour le traitement de textes et le développement d'applications. Ken Thomson et son équipe sont chargés de ce travail.

- **1969** : apparition de la 1ère version d'UNIX. Le nom UNIX provient de UNICS (UNiplexed Information and Computing System), système d'exploitation qui a succédé à Multics dans les laboratoires Bell.

- **1973** : nécessité de rendre UNIX portable sur d'autres ordinateurs. Denis Ritchie réécrit alors entièrement UNIX en langage C qui a d'ailleurs été créé dans ce but précis. Ceci explique les liens profonds entre le langage C et UNIX.

- **1974** : AT&T propose les 1ères licences aux universités ce qui apporta un enrichissement en extensions et en utilitaires variés à UNIX (en particulier, l'Université de Berkeley). Cette date correspond au début de la popularité et de la diversité d'UNIX.

- **1978** : AT&T présente à l'industrie les 1ères versions commerciales.

- **années 80** : AT&T autorise le clonage d'UNIX par d'autres constructeurs. Ainsi, apparaissent ULTRIX sur DEC, BSD sur SUN, AIX sur IBM, etc. Ces versions constructeur dérivent toutes des 2 versions présentes à l'époque et qui sont :
    - System V pour des configurations moyennes et petites ; USL (Unix Systems Labs, filiale d'AT&T) en est responsable actuellement ;
    - BSD (Berkeley Software Distribution) pour des configurations importantes dans le domaine scientifique).

À noter qu'il existe quelques différences de syntaxe entre certaines commandes UNIX de type System V et celles de type BSD. Dans le document suivant, la syntaxe utilisée est celle utilisée par la version BSD sur SUN (stations utilisées pour ce cours).

1.2- Fonctionnalités

Les 4 fonctions principales d'UNIX sont :

1.2.1- Gestion des ressources de l'ordinateur

Ce qui a fait le grand succès d'UNIX, c'est le fait d'être un système d'exploitation multi-tâches et multi-utilisateurs. En effet, sous UNIX, le temps d'utilisation du processeur de l'ordinateur est réparti entre différentes tâches ce qui se traduit par l'exécution simultanée de programmes. D'autre part, UNIX traite les commandes de plusieurs utilisateurs en même temps. Dans ce contexte, il doit répartir les ressources entre les différentes tâches et utilisateurs de façon transparente pour ces derniers.

1.2.2- Gestion des données

Celle-ci consiste en l'organisation, la maintenance et l'accès aux unités de stockage (mémoire, disques durs, bandes magnétiques, etc.)

1.2.3- Communication entre utilisateurs

C'est par exemple le courrier électronique ou les transferts de fichiers dont l'utilisation est expliquée plus loin.

1.2.4- Environnement de programmation

Ce sont les compilateurs (C et quelquefois Fortran), éditeurs de textes, outils d'aide à la programmation (débogueurs, etc.).

2- Connexion et déconnexion

On suppose ici, que l'utilisateur a déjà été enregistré sur le système et que donc, son identité est présente dans des fichiers particuliers gérés par l'administrateur du système.

2.1- Procédure de connexion

Celle-ci peut être scindée en 2 phases :

2.1.1- Connexion logique

La connexion logique doit avoir lieu entre le terminal (écran + clavier) et l'ordinateur sur lequel on veut travailler. Le terminal peut être aussi un autre ordinateur (stations UNIX, PCs, Macintosh, etc.). Cette connexion dépend de la liaison matérielle entre le terminal et l'ordinateur. Dans le cas le plus simple de la liaison directe, il suffit d'allumer le terminal.

2.1.2- Initialisation de la session de travail

C'est durant cette étape que l'utilisateur va devoir s'identifier auprès du système. Cette identification a lieu de la façon suivante :
- affichage du message `login:` après lequel il faut rentrer son nom d'utilisateur (ou logname) ;
- affichage du message `Password:` après lequel il faut rentrer son mot de passe. Celui-ci n'est pas affiché pendant la frappe pour éviter bien sûr que quelqu'un d'autre puisse l'apercevoir.

Puis, souvent, apparaît un message demandant de spécifier le type de terminal utilisé (ex : `vt100`) et ceci de façon à ce que les éditeurs fonctionnent correctement ; si ce message n'est pas affiché, un type de terminal est pris par défaut.
Après la connexion, différents messages en provenance de l'administration du système (mots du jour, présence de courrier dans la boîte à lettres, etc.) sont affichés.
L'utilisateur est effectivement prêt à travailler quand il reçoit l'invite du système consistant en un marqueur en début de ligne. Ce marqueur est variable selon les machines (ex : `$` ou `nom_utilisateur@nom_machine>`)

2.2- Mot de passe

2.2.1- Changement de passe

Le changement de mot de passe s'effectue avec la commande `passwd` (ou `yppasswd`). L'utilisation de l'une ou l'autre commande dépend du type de configuration du système d'enregistrement des utilisateurs d'un système UNIX.
Si ceux-ci sont les utilisateurs locaux d'une machine donnée, ils doivent utiliser la commande `passwd` . Dans le cas où les utilisateurs sont enregistrés par le système des pages jaunes (Yellow Pages), encore appelé NIS (Network Information Service), ils doivent utiliser la commande `yppasswd.` Le système "Yellow Pages" permet entre autres à l'utilisateur, d'accéder à un compte unique et ceci à partir de différentes machines.

Le changement de mot de passe s'effectue en entrant d'abord le mot de passe actuel puis en entrant le nouveau mot de passe que l'on doit retaper pour confirmation.

```
perrot@von-neumann>yppasswd
Changing NIS password for perrot on von-neumann.
Old password:
New password:
Retype new password:
NIS entry changed on von-neumann
```

Si le nouveau mot de passe est rejeté, la raison peut être que:
- la 2ème entrée du nouveau mot de passe (pour confirmation) était différente de la première ;
- le nouveau mot de passe ne répond pas à des exigences de sécurité (voir ci-dessous).

2.2.2- Qualités d'un "bon" mot de passe

Des personnes peuvent essayer d'accéder de façon illégale à un système UNIX. Un des moyens dont ils disposent est d'entrer sur un compte utilisateur en déterminant son mot de passe. Pour cela, ils utilisent des chaînes de caractères appartenant à des dictionnaires ou correspondant à des informations personnelles sur l'utilisateur (prénom, nom, numéro de téléphone, etc.). Il est donc nécessaire que votre mot de passe respecte certaines règles de sécurité :

- il doit posséder au moins 7 caractères et contenir au moins une lettre majuscule, un chiffre et un caractère de ponctuation, et ceux-ci à l'intérieur et non en début ou fin de mot de passe. Généralement, seuls les 8 premiers caractères sont pris en compte ;
- il ne doit pas contenir des données relatives à votre identité comme votre nom d'utilisateur ou une information livrée par la commande `finger` (voir plus loin) ;
- il ne doit pas appartenir à des dictionnaires, tel quel ou sous sa forme canonique (c'est à dire, épuré de tous les caractères non-alphabétiques), à moins qu'il contienne des majuscules autres que le premier caractère ;
- il ne doit pas contenir des répétitions de caractère ;
- il doit être suffisamment simple pour s'en rappeler ; il ne faut pas le noter sur papier ou dans un fichier ni le donner à quelqu'un d'autre.

Ex. de "bons" mots de passe (2 mots courts séparés par un ou plusieurs caractères de ponctuation ou chiffres) : ble!1the ou si%@sol.

2.3- Procédure de déconnexion

Celle-ci dépend du type de session qui a été ouverte. Si un environnement graphique (tel que celui créé par X Windows, l'environnement multi-fenêtrage) est en place, il existe généralement un menu "logout" (ou "exit") qui permet de quitter cet environnement et ainsi de terminer la session de travail. Parfois aussi, ce menu ne permet simplement que de quitter l'environnement graphique. Il faut ensuite procéder à l'étape de déconnexion ci-dessous.

En l'absence d'un environnement graphique, une simple commande telle que `logout` ou `exit`, entrée après l'invite du système, suffit pour terminer la session de travail.

3- Commandes

3.1- Interpréteur de commandes

L'utilisateur peut taper des commandes lorsqu'il se trouve au niveau commande, c'est-à-dire lorsque l'invite du système apparaît. Toute commande entrée sera interprétée par l'interpréteur de commandes (ou shell). Le terme shell veut dire coquille pour exprimer l'idée d'interface entre utilisateurs et système UNIX et a été donné par opposition au noyau du système.
Il existe des dizaines d'interpréteurs de commandes sous UNIX mais les 2 principaux (qu'on retrouve sur la plupart des systèmes) sont le Bourne-shell (sh) et le C-shell (csh). Le choix de l'interpréteur activé à la connexion est fait à l'enregistrement de l'utilisateur dans le système. Dans la plupart des cas, c'est le C-shell.

3.2- Syntaxe des commandes

Celle-ci est généralement la suivante :

```
nom_commande [options] [arguments]
```

. le caractère séparateur entre les différents éléments de la commande est le blanc (ESPACE) ;
. les options commencent habituellement par le caractère - (signe moins) suivi d'une ou plusieurs lettres-clés. Ces options vont modifier le comportement de la commande ;
. les arguments spécifient les objets (fichiers ou variables) sur lesquels la commande va s'appliquer.
Note : les crochets autour des arguments et des options signifient que ceux-ci sont optionnels.

Ex : liste de fichiers avec les commandes suivantes :
`ls`
`ls -l` (l comme long, donne tous les attributs des fichiers)
`ls -la` (a comme all, liste aussi les fichiers commençant par le caractère .)
`ls -l rep1`

Ex : visualisation d'un ou plusieurs fichiers :
`cat fic1`
`cat fic1 fic2`

À noter que tous les shells font la distinction entre les lettres minuscules et majuscules pour les commandes et les noms de fichiers contrairement au MS-DOS.

3.3- Manuel des commandes

Si on veut de l'aide sur les règles d'utilisation ou encore sur les fonctionnalités d'une commande, on peut utiliser l'aide en ligne grâce à la commande `man` (comme MANual pages) de la façon
suivante : `man nom_commande`

Ex :
```
man ls
man man
```

Note : l'affichage des "manual pages" se fait à l'aide de la commande `more,` afficheur page par page dont on verra l'utilisation plus loin. Pour faire avancer l'affichage, il suffit de taper la barre d'espace.

Si on ne connaît pas la syntaxe de la commande, il est possible de faire une recherche par mot-clé dans le système de "manual pages") à l'aide de la commande suivante :
`man -k mot-clé`

Ex : `man -k list` ; `man -k directories`

3.4- Caractères génériques

Certaines commandes acceptent plusieurs noms de fichiers en arguments aussi, il était intéressant d'avoir des notations permettant de raccourcir l'écriture d'une telle liste. Ainsi, il existe plusieurs caractères génériques qui, incorporés dans les noms de fichiers, ont la signification suivante :
- le caractère `?` qui peut remplacer n'importe quel caractère ;
- le caractère `*` (astérix) qui peut remplacer n'importe quelle chaîne de caractères, y compris la chaîne vide.

Ex :

```
perrot@von-neumann>ls
fic   fic1   fic2   fic3   fic33   mbox   rep1   rep2   rep3
perrot@von-neumann>ls fic?
fic1  fic2  fic3
perrot@von-neumann>ls fic*
fic    fic2   fic33   fic1   fic3
perrot@von-neumann>ls fic??
fic33
```

Attention aux erreurs de frappe ou d'appréciation : `rm *.o` et `rm * .o` ont des résultats bien différents !

3.5- Redirection des entrées/sorties

Généralement, les commandes lisent l'entrée standard et/ou écrivent sur la sortie standard. Normalement, l'entrée standard est le clavier et la sortie standard est l'écran. Il est possible de rediriger ces entrée et sortie standards vers des fichiers.

Pour que le résultat d'une commande soit rangé dans un fichier au lieu d'apparaître à l'écran, il faut utiliser la syntaxe :

```
nom_commande [options] [arguments] > fichier_sortie
```

Ex : `ls -l > poub` et `date > poub`

On voit que si le fichier de redirection existe déjà, son contenu est écrasé avec la redirection >. Si on veut que ce contenu soit préservé et y ajouter des résultats d'une commande, il faut utiliser la redirection >>.

Ex : `ls -l > poub` et `date >> poub`

Au lieu de fournir des données en les entrant au clavier, ces données peuvent être lues dans un fichier avec la syntaxe :

```
nom_commande [options] [arguments] < fichier_entrée
```

Ex :

`wc` (imprime le nombre de lignes, de mots et de caractères fournis à l'entrée standard)

```
ls -l fic? > poub ; wc < poub
```

3.6- Tube («pipe»)

Dans le même ordre d'idées, on peut utiliser le mécanisme de «pipe» qui permet de prendre la sortie standard d'une première commande et de la rediriger sur l'entrée standard d'une 2ème commande. La syntaxe est :

```
nom_commande1 [options] [arguments] | nom_commande2
[options] [arguments]
```

Ex :

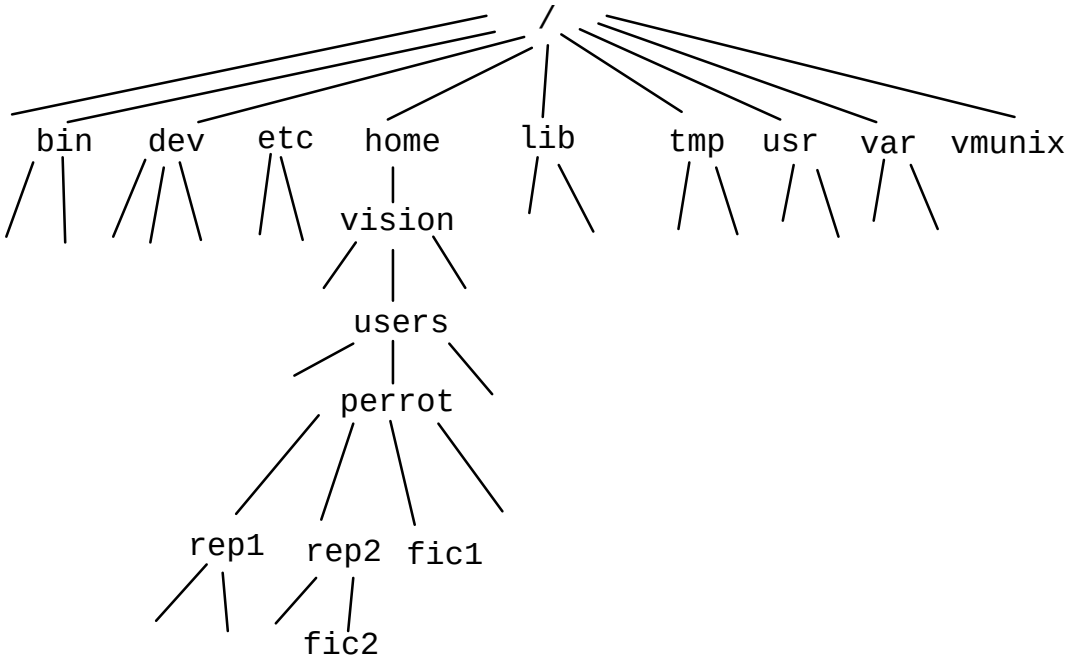
`ls -l fic? | wc` (raccourci de la série de redirections vu précédemment)

```
ls /etc | more
```

4- Organisation des fichiers

4.1- Arborescence d'UNIX

L'unité d'information gérée par le système est le fichier et celui-ci peut contenir n'importe quoi et être éventuellement vide. Selon leur utilisation, les fichiers sont appelés répertoires («directories») ou fichiers tout court («files»). Un répertoire est un catalogue de fichiers contenant leurs caractéristiques comme les droits d'accès, la taille, la date de création, etc.
L'ossature du système est une structure arborescente de fichiers et de répertoires. Chaque utilisateur peut ajouter dans son coin de nouvelles branches.
/ est le nom du répertoire racine («root») de l'arbre.



4.1.1- Nom absolu d'un fichier

Le nom absolu d'un fichier est formé de tous les noms de répertoires traversés depuis la racine pour l'atteindre, noms séparés par des obliques avant `/` (contrairement au MS-DOS qui utilisent des obliques arrière \). Chaque fichier a un nom absolu unique dans le système.

Ex : 2 fichiers de même nom, `fic1`, peuvent coexister s'ils sont dans 2 répertoires différents et ont donc un nom absolu différent :
`/home/vision/users/perrot/fic1` et
`/home/vision/users/perrot/rep1/fic1`.

4.1.2- Nom relatif

Le répertoire courant («working directory») permet de raccourcir la notation d'un nom de fichier en utilisant des noms relatifs à ce répertoire courant. Au début de la session, le répertoire courant est le répertoire personnel de l'utilisateur. Le répertoire courant est noté . et peut être en tout temps identifié grâce à la commande `pwd`. Le père du répertoire courant est noté `..` et correspond à un niveau supérieur dans l'arborescence.
Ex :

```
perrot@von-neumann>pwd
/home/vision/users/perrot
perrot@von-neumann>cd rep3
perrot@von-neumann>pwd
/home/vision/users/perrot/rep3
perrot@von-neumann>ls -la
total 3
drwx------  2 perrot        512 Sep 21 21:30 .
drwxr--r-x  8 perrot        512 Sep 21 21:30 ..
-rw-------  1 perrot         50 Sep 21 21:30 fic1
perrot@von-neumann>cd ../rep2
perrot@von-neumann>pwd
/home/vision/users/perrot/rep2
perrot@von-neumann>cd /home/vision/users/perrot/rep1
perrot@von-neumann>pwd
/home/vision/users/perrot/rep1
```

Note : la commande `cd` (Change Directory) permet de changer de répertoire. En argument, on lui passe le nom du répertoire dans lequel on veut se déplacer. Sans argument, on revient au répertoire personnel.

4.2- Droits d'accès des fichiers

Chaque fichier (ou répertoire) possède un ensemble d'attributs définissant les droits d'accès à ce fichier pour tous les utilisateurs du système.

4.2.1- Classes d'utilisateurs

Il existe 3 classes d'utilisateurs pouvant éventuellement accéder à un fichier :
- le propriétaire du fichier (User) ;
- le groupe dans lequel appartient le propriétaire (Group) ;
- les autres (Others).

A sa création, un fichier appartient à son auteur. Le propriétaire du fichier peut ensuite distribuer ou restreindre les droits d'accès sur ce fichier (voir plus loin).

4.2.2- Types d'accès

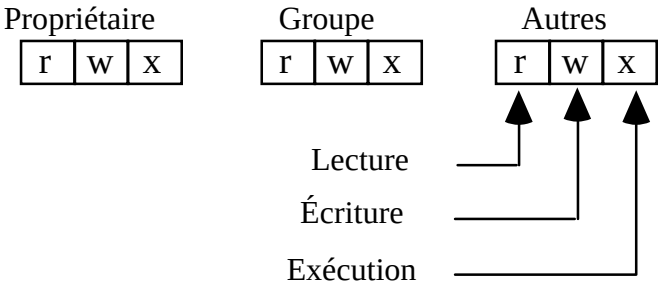
Pour chaque classe d'utilisateurs, il y a 3 types d'accès à un fichier donné :
- `r` : en lecture (Read) ;
- `w` : en écriture (Write) ;
- `x` : en exécution (eXecute).

Au niveau répertoire, ces droits signifient :
- droit de lister les fichiers présents dans ce répertoire (Read) ;
- droit de créer ou de détruire un fichier qui s'y trouve (Write) ;
- droit de traverser ce répertoire (eXecute).

En combinant les 3 types d'accès et les 3 classes d'utilisateurs, il y a donc 9 droits d'accès différents qui existent sous UNIX.

4.2.3- Visualisation des droits d'accès

Pour cela, on utilise la commande `ls -l`. Le 1er caractère spécifie si le fichier est un répertoire (caractère `d`) ou un fichier tout court (caractère `-`). Les 9 caractères suivants identifient les droits d'accès (présence du droit si lettre `r`, `w` ou `x` ; absence de droit si caractère `-`) et sont structurés de la façon suivante :



Ex :

```
perrot@von-neumann>ls -l
total 7
-rw-------  1 perrot         49 Sep 18 12:37 fic
-rw-------  1 perrot         50 Sep 18 12:35 fic1
-rw-------  1 perrot         50 Sep 18 12:37 fic2
-rw-------  1 perrot         50 Sep 18 12:37 fic3
-rw-------  1 perrot         51 Sep 18 12:38 fic33
```

4.2.4- Modification des droits d'accès

Seul, le propriétaire d'un fichier peut modifier ses droits d'accès. Pour cela, il utilise la commande `chmod` avec la syntaxe suivante :

`chmod mode nom_fichier`

`mode` indique de quelle façon les droits d'accès doivent être modifiés. Il se décompose en `[qui] op accès.`

`qui` (optionnel) indique quelles classes sont concernées par la commande `chmod` et est composé de 1 ou plusieurs lettres parmi `u`, `g` et `o`. Si aucune classe n'est spécifiée, toutes les classes sont concernées.

`op` peut être :

`+` pour ajouter des droits d'accès

`-` pour enlever des droits d'accès

`accès` est une combinaison des lettres `r`, `w` et `x` qui spécifient les types d'accès.

Ex :

```
perrot@von-neumann>ls -la rep3
total 3
drwx------  2 perrot        512 Sep 21 21:30 .
drwxr--r-x  8 perrot        512 Sep 21 21:30 ..
-rw-------  1 perrot         50 Sep 21 21:30 fic1
perrot@von-neumann> chmod u-r rep3
perrot@von-neumann>ls -la rep3
rep3 unreadable
perrot@von-neumann>ls -la fic2
-rw-rw----  1 perrot        101 Sep 18 19:15 fic2
perrot@von-neumann>fic2
fic2: Permission denied.
perrot@von-neumann> chmod +x fic2
perrot@von-neumann>ls -la fic2
-rwxrwx--x  1 perrot        101 Sep 18 19:15 fic2
perrot@von-neumann>fic2
Il est : 11:44:37
```

Note : les droits du propriétaire d'un fichier sont uniquement déterminés par la partie propriétaire de la protection (même s'il est membre d'un groupe).

4.2.5- Initialisation des droits d'accès

Au moment où l'utilisateur crée un fichier, des droits d'accès par défaut sont donnés à ce fichier. C'est la commande `umask` qui permet de définir la protection maximale que l'on donne par défaut aux nouveaux fichiers. La syntaxe est la suivante : `umask masque.`

`masque` est une valeur octale qui joue le rôle de masque sur les droits d'accès d'un fichier à sa création. Les droits d'accès sont déterminés sont obtenus après l'opération logique suivante : mode & (~masque) avec &, le ET logique et ~ le NON logique.
En standard, une commande shell de création de fichier, comme `cat > nom_fichier` (ce fichier n'existant pas) initialise le mode de ce fichier à `rw-rw-rw-`. C'est à ce mode par défaut qu'est appliqué le masque.

Ex :
`umask 22` (le masque est ici, 022, soit en codage binaire, 000 010 010 => les fichiers créés auront la protection `rw-r--r--`).
`umask 77` (le masque est ici, 077, soit en codage binaire, 000 111 111 => les fichiers créés auront la protection `rw-------`).

Note : au moment de l'initialisation de la session de travail, une valeur de masque est définie. Pour connaître la valeur du masque, il suffit d'utiliser la commande `umask` sans arguments.
La commande `chmod` n'est pas affectée par la valeur du masque.

5- Manipulation des fichiers

5.1- Liste des fichiers

On utilise pour cela la commande `ls` avec ses nombreuses options.

5.2- Création d'un fichier

Habituellement, on utilise un éditeur de textes. Ici, on suppose que l'utilisateur ne sait utiliser aucun des éditeurs et qu'en attendant, il a la recette suivante pour créer un petit fichier :

```
cat > nom_fichier
1ère ligne
2ème ligne
.
.
dernière ligne
^D
```

Avec cette recette, si le fichier existe déjà, son précédent contenu est détruit.
La procédure ci-dessus est simple mais manque de fonctionnalités en ce qui concerne l'édition du fichier (correction d'erreurs impossible sur les lignes précédentes).

5.3- Destruction d'un fichier

```
rm nom_fichier
```

Note :
Quelquefois, le système demande confirmation avant d'effacer un fichier et si la réponse de l'utilisateur ne commence pas par 'y' ou 'Y' alors le fichier est préservé. Toutefois, la commande Undelete sous DOS n'existe pas sous UNIX. Il est en effet impossible de récupérer un fichier effacé et ceci rend les sauvegardes sur bandes magnétiques

indispensables (il faut donc contacter l'administrateur du système pour s'informer à ce sujet). Cela veut dire aussi qu'il faut être très prudent avec la commande `rm`.

5.4- Visualisation d'un fichier

Plusieurs commandes à différentes fonctionnalités permettent la visualisation d'un fichier mais les plus usuelles sont :

`cat nom_fichier`
`more nom_fichier` qui permet un affichage page par page (h pour connaître les commandes possibles de `more`, `RETURN` pour la ligne suivante, `ESPACE` pour la page suivante et `q` pour terminer l'affichage).

5.5- Copie d'un fichier

```
cp nom_fichier1 nom_fichier2
cp nom_fichier1 nom_répertoire
```

5.6- Renommage ou déplacement d'un fichier

`mv ancien_nom nouveau_nom` (renommage)
`mv nom_fichier nom_répertoire` (déplacement)

5.7- Création d'un répertoire

`mkdir nom_répertoire` (possible uniquement si celui-ci est inexistant)

5.8- Destruction d'un répertoire

`rmdir nom_répertoire` (si le répertoire est vide)
`rm -r nom_répertoire` (`r` comme récursif : effacement du contenu du répertoire et de lui-même)

6- Communication

6.1- Le réseau Internet

Avant de parler des outils de communication d'UNIX, il était important de présenter brièvement le réseau Internet. En effet, ces outils de communication (courrier électronique, transfert de fichiers, etc.) peuvent fonctionner sur ce réseau qui relie des ordinateurs dans le monde entier et c'est là que réside un de leurs principaux intérêts.
Le réseau Internet est en effet constitué de milliers d'ordinateurs du monde entier, connectés entre eux par divers types de liaisons matérielles. Le point commun entre toutes ces machines est qu'elles communiquent entre elles avec le protocole TCP/IP (Transmission Control Protocol / Internet Protocol).
Internet est né il y a 20 ans au moment où le US Department of Defense créa ARPAnet (Advanced Research Projects Agency NETwork) pour les employés du gouvernement. En 1984, il y avait déjà plusieurs réseaux locaux et NSF (National Science Foundation) développa son propre réseau pour connecter entre elles les différentes universités aux US (au début pour accéder à des supers machines de calcul puis pour partager de l'information).

6.2- Adresse d'une machine

L'adresse d'une machine connectée au réseau Internet est unique et s'écrit sous 2 formats possibles :
- numérique : xxx.xxx.xxx.xxx. (4 nombres correspondants à un code 32 bits = 4 octets ; chaque nombre étant donc compris entre 1 et 255) ;

Note : le début de cette adresse donne le numéro du réseau alors que la fin de l'adresse correspond au numéro de la machine sur ce réseau.

Ex : la machine `von-neumann` a pour adresse Internet numérique 132.207.12.1. avec 132.207 qui identifie le réseau Internet de l'École Polytechnique, 132.207.12, le réseau local du département de Génie Informatique et `1` le numéro de cette machine sur ce réseau local.

- symbolique : nom_machine.domaine_internet. A l'adresse numérique est associée cette adresse symbolique qui est généralement plus facile à retenir. Souvent, les domaines Internet aux USA se terminent par un suffixe désignant le type d'organisation : `.com` pour les entreprises commerciales, `.org` pour les entreprises à but non-lucratif, `.gov` et `.mil` pour les agences militaires ou du gouvernement, `.net` pour les compagnies qui utilisent des réseaux de grande taille. Les autres domaines Internet utilisent habituellement un suffixe à 2 lettres désignant le pays concerné. Ex : `.ca` (CAnada), `.fr` (FRance), `.us` (USa), etc.

Ex : l'adresse symbolique complète de la machine `von-neumann` est `von-neumann.info.polymtl.ca`.
Note : on peut obtenir le nom de la machine sur laquelle on travaille avec la commande `hostname`.

Ex :
```
perrot@von-neumann>hostname
von-neumann
```

6.3- Adresse d'un utilisateur

L'adresse d'un utilisateur sur une machine (son adresse e-mail, voir plus loin) est elle aussi unique et s'écrit : `nom_utilisateur@domaine_internet` ou parfois `nom_utilisateur@adresse_machine`.

Ex : `perrot@info.polymtl.ca`.

Notes : généralement, il importe peu que les noms de machines et d'utilisateurs soient en majuscules et minuscules (à part de rares exceptions pour certains noms d'utilisateurs). D'autre part, il n'est pas nécessaire de spécifier le domaine internet d'une machine ou d'un utilisateur lorsque l'on veut effectuer une requête à ces derniers si ceux-ci sont dans le même domaine.

6.4- Courrier électronique («e-mail»)

Chaque utilisateur a à sa disposition une boîte à lettres où sont stockées les lettres qui lui sont envoyées par d'autres utilisateurs. Il existe différents logiciels pour lire et envoyer du courrier électronique mais on va parler ici de l'outil standard présent sous tout système UNIX : `mail`.

6.4.1- Envoi de courrier

Pour cela, la commande à utiliser est : `mail adresse_destinataire.` Si le destinataire du courrier est sur le même réseau local que l'expéditeur, son nom d'utilisateur suffit comme adresse. Dans le cas contraire, l'adresse Internet complète est nécessaire.
Après cette commande, l'utilisateur spécifie le sujet (court si possible) de son e-mail après le champ `Subject:` puis entre la lettre à envoyer. Cette dernière se termine par une ligne commençant par un point ou par la marque de fin de fichier soit `^D`.

Ex : `mail perrot`
`1ère ligne`
`2ème ligne`
`.`
`dernière ligne`
`^D`

6.4.2- Lecture de courrier

On est informé de la présence de courrier dans la boîte à lettres au moment de la connexion par le système grâce à un message à l'écran. Sur la plupart des systèmes, si on est déjà logé au moment où arrive une nouvelle lettre, celle-ci est affichée. Sur certains systèmes, seul un message signalant l'arrivée de courrier est affiché. Il existe de nombreux utilitaires d'avertissement (textes, graphiques ou sonores) de l'arrivée de courrier.
Pour consulter son courrier, il faut taper la commande : `mail [options].`

Ex :
```
perrot@von-neumann>mail
Mail version SMI 4.0 Wed Oct 13 18:37:02 PDT 1993  Type ?
for help.
```

```
"/usr/spool/mail/perrot": 2 messages 2 new
>N  1 perrot             Sun Sep 18 20:43   11/316   test
 N  2 perrot             Sun Sep 18 20:45   11/312   test2
&
```

Diverses informations sont affichées :
- la version de l'outil `mail` ;
- le nom du fichier où les lettres sont contenues ainsi que le nombre de lettres et leurs types (ex : `new` si lettre jamais lue) ;
- la liste numérotée des lettres reçues avec leur type (N comme `new`), le numéro dans la liste (dans l'ordre chronologique d'arrivée), le nom de l'expéditeur, la date et l'heure d'expédition, le nombre de lignes et de caractères constituant la lettre et enfin le sujet de la lettre.

Le curseur `>` pointe vers la lettre courante, celle sur laquelle les commandes `mail` agissent par défaut. Si on tape sur la touche `RETURN`, le contenu de cette lettre courante est affichée.
Les 1ères lignes, dites d'en-tête ("headers"), de la lettre donnent de l'information de base sur la lettre et sont directement utilisées par l'outil `mail` (pour afficher la liste des lettres par ex.).
En tapant une seconde fois sur la touche RETURN, la seconde lettre dans la liste (elle est devenue la lettre courante) est affichée.

Pour connaître les commandes disponibles sous l'outil `mail`, il faut utiliser la commande `help` (ou `?`). On obtient alors la liste des commandes suivantes :

```
& ?
cd [directory]                  chdir to directory or home
if none given
d [message list]                delete messages
e [message list]                edit messages
f [message list]                show from lines of messages
h                               print out active message
headers
m [user list]                   mail to specific users
n                               goto and type next message
p [message list]                print messages
pre [message list]              make messages go back to
system mailbox
q                               quit, saving unresolved
messages in mbox
r [message list]                reply to sender (only) of
messages
R [message list]                reply to sender and all
recipients of messages
s [message list] file           append messages to file
t [message list]                type messages (same as
print)
top [message list]              show top lines of messages
u [message list]                undelete messages
v [message list]                edit messages with display
editor
w [message list] file           append messages to file,
without from line
x                               quit, do not change system
mailbox
z [-]                           display next [previous] page
of headers
!                               shell escape

A [message list] consists of integers, ranges of same, or
user names separated
by spaces.  If omitted, Mail uses the current message.
&
```

La liste des commandes couramment utilisées est :

`h` : (h comme headers) pour voir la liste des lettres ;
`num_lettre` : pour afficher le contenu de la lettre `num_lettre` ;
`d` : pour effacer la lettre courante;
`u` : pour annuler l'effacement de la lettre courante (avant de quitter l'outil `mail`) ;
`s nom_fichier` : pour sauver la lettre courante dans le fichier `nom_fichier` ;
`q` : pour quitter l'outil `mail` ; les lettres lues et non-effacées (ou sauvegardées) sont rangées par défaut dans le fichier `mbox` du répertoire personnel.

Pour lire le courrier rangé dans un fichier particulier autre que `/usr/spool/mail/nom_utilisateur`, il faut utiliser la commande : `mail -f nom_fichier`.

Notes :
- il est conseillé de ranger son courrier dans plusieurs fichiers spécifiques, de façon à retrouver facilement un ancien courrier pour une consultation ultérieure, plutôt que d'avoir un fichier `mbox` qui contient des dizaines de lettres sans lien entre elles ;
- des logiciels plus conviviaux (`elm`, `pine`, etc.) que `mail` existent souvent sur les systèmes UNIX.

6.5- Connexion sur une machine distante

On a la possibilité, à partir d'une machine sur laquelle on a déjà ouvert une session de travail, de se connecter à un autre ordinateur via le réseau (local ou Internet). La commande à utiliser est la suivante :

```
telnet adresse_machine
```

Pour pouvoir se connecter, il faut bien sûr avoir un compte sur cette machine distante. Il faut alors s'identifier de la même façon qu'on le fait pour se connecter une première fois sur une machine. Une fois logé sur la machine distante, les commandes entrées sont exécutées sur cette machine.

Ex :

```
telnet von-neumann
Trying 132.207.12.1 ...
Connected to von-neumann.
Escape character is '^]'.

SunOS UNIX (von-neumann)

login: perrot
Password:
```

6.6- Transfert de fichiers

FTP (File Transfer Protocol) est un programme permettant de transférer des fichiers entre 2 machines.

6.6.1- Procédure générale

FTP fonctionne un peu comme la commande `telnet`. Il faut d'abord lui fournir en argument l'adresse de la machine avec laquelle on désire effectuer des transferts de fichiers de la façon suivante :

```
ftp adresse_machine
```

Puis, on s'identifie auprès du système avec un nom d'utilisateur et un mot de passe.

6.6.2- FTP anonymes

Par contre, contrairement au service Telnet, FTP peut donner un accès (restreint) à des machines où l'on n'a pas de compte utilisateur. Pour cela, il faut que la machine permette ce qu'on appelle les FTP anonymes. Dans ce cas, on fournit comme nom d'utilisateur, le nom `anonymous` et comme mot de passe, son adresse e-mail (ce qui permet de faire des statistiques sur l'utilisation du site FTP). L'accès au site FTP est alors restreint puisque que l'on n'a pas de compte utilisateur dessus. Il est possible par exemple que l'on ne puisse que retirer des fichiers sur cette machine (généralement présents dans le répertoire `/pub).`

Ex :

```
ftp ftp.umontreal.ca
login: anonymous
Password: adresse_e-mail
```

Une fois logé sur le site à l'aide de FTP, on accède aux commandes FTP quand on voit l'invite `ftp>`. La liste de ces commandes est obtenue avec la commande `help`. Pour obtenir de l'aide sur une commande en particulier, il faut taper la commande :
`help nom_commande`.
Les commandes importantes à connaître sont :
- `ls` (liste des fichiers du répertoire courant); les caractères génériques sont utilisables ici aussi ;
- `cd nom_répertoire` (changement de répertoire) ;
- `get nom_fichier` : récupère un fichier sur la machine distante. Il y a possibilité de sauver le fichier sous un autre nom sur la machine locale ;
- `put nom_fichier` : transfert un fichier de la machine locale à la machine distante ;
- mode de transfert : `ascii` (par défaut) ou `binary` ; en effet, en mode ascii, certains caractères sont traduits entre 2 systèmes différents pour les rendre plus lisibles. Par contre, pour les fichiers binaires (pas de caractères ascii), cette traduction ne doit pas avoir lieu ;
- transfert multiple : `mget` et `mput.` Avec ces 2 commandes (dont la fonction est identique à celle de leur équivalents, `get` ou `put`), on peut transférer plusieurs fichiers à la fois. L'utilisation des caractères génériques est alors intéressante ;
- confirmation : `prompt.` Dans le cas de l'utilisation de `mget` et `mput`, le système demande, pour chaque fichier concerné, de valider ou non son transfert. Si on veut annuler cette demande systématique de confirmation, on utilise la commande : `prompt`.

6.7- Commande `finger`

La commande `finger` permet d'avoir de l'information sur un utilisateur qui a un compte sur une machine particulière. On l'utilise ainsi :

```
finger nom_utilisateur@adresse_machine
```

Ex :

```
perrot@von-neumann>finger perrotg@eole.ere.umontreal.ca
[eole.ere.umontreal.ca]
Login name: perrotg           In real life: Perrot Gildas
Mail to perrotg goes to perrotg@eole.ERE.UMontreal.CA
Project: Traitement d'images par methodes statistiques.
Plan:
Cours UNIX du 17 au 29 octobre.
```

Remarques :

- la commande `finger` affiche en plus du nom de la personne, le contenu des fichiers textes `.plan` (plan de travail) et `.project` (projet sur lequel travaille cette personne) s'ils sont présents dans le répertoire personnel de l'utilisateur ;
- pour des raisons de sécurité, certaines machines ne permettent pas l'accès par `finger` aux informations concernant les utilisateurs qui y ont un compte.

6.8- Commande `ping`

Cette commande permet de vérifier s'il existe une connexion possible entre la machine où l'on travaille et celle spécifiée en argument : `ping adresse_machine.`

Ex : `ping von-neumann`

```
von-neumann is alive
```

Elle permet aussi de déterminer l'adresse numérique d'une machine avec la syntaxe suivante : `ping -s adresse_machine` (envoi d'un paquet par seconde et affichage de la présence de réponse de la machine).

```
Ex :
perrot@von-neumann>ping -s vision
PING vision: 56 data bytes
64 bytes from vision (132.207.12.129): icmp_seq=0. time=2.
ms
64 bytes from vision (132.207.12.129): icmp_seq=1. time=4.
ms
```

^C

6.9- Commande `talk`

Parfois, on peut préférer au courrier électronique, le principe de conversation interactive. Ce principe est mis en place sous UNIX grâce à la commande `talk`. Pour cela, `utilisateur1` qui travaille sur `machine1` utilise la syntaxe :
`talk utilisateur2` (si les 2 correspondants sont sur la même machine) ou
`talk utilisateur2@machine2` (si elles sont sur 2 machines différentes). `utilisateur2` reçoit un message du type :

```
Message from TalkDaemon@machine2...
talk: connection requested by utilisateur1@machine1.
talk: respond with: talk utilisateur1@machine1.
```

`utilisateur2` accepte la communication avec `utilisateur1` en entrant elle-même la commande de réponse `talk` spécifiée.
La fenêtre de travail de chacun des 2 correspondants se partagent alors en 2 et chacun voit dans la 1ère moitié, ce qu'il entre et dans la 2ème moitié, ce que son correspondant entre au clavier. La fin de la communication est généralement signalée par l'un ou l'autre des correspondants par la commande d'interruption `^C`.

7- Recherches

On va examiner ici plusieurs commandes effectuant des recherches de fichiers ou dans des fichiers.

7.1- Recherche de chaîne dans un fichier : `grep`

UNIX offre la commande grep pour afficher les lignes de fichiers donnés en arguments et qui contiennent un motif donné. La syntaxe à utiliser est :
grep [option] motif [nom_fichier]
Ex :

```
perrot@von-neumann>grep son fic?
fic1:Ceci est le fichier fic1 du repertoire personnel.
fic3:Ceci est le fichier fic3 du repertoire personnel.
```

options :
-v : affichent les lignes qui ne contiennent pas le motif ;
-i : ignore la distinction minuscule/majuscule dans les comparaisons.

Ex :

```
perrot@von-neumann>grep -i est fic?
fic1:Ceci est le fichier fic1 du repertoire personnel.
fic2:echo Il est : `date | awk '{printf "%s\n", $4}'`
fic3:Ceci est le fichier fic3 du repertoire personnel.
fic4:CECI EST LE FICHIER fic4.
perrot@von-neumann>grep -v est fic*
fic2:#!/bin/sh
fic4:CECI EST LE FICHIER fic4.
fic4:
```

7.2- Recherche d'un fichier : `find`

`find` descend récursivement dans des sous-arborescences de répertoires, en cherchant à appliquer à des fichiers précisés par un ou plusieurs critères de sélection (nom, type, date de modification, etc.), une commande donnée. `find` s'utilise de la façon suivante :

```
find liste_de_répertoires expression
```

liste_de_répertoires est la liste des racines des arborescences à parcourir ;
expression est une suite d'options exprimant les critères de sélection des fichiers et les actions à leur appliquer. Lorsque que le critère est vrai, l'action est exécutée. Dans la suite, on appelle fichier courant, le fichier examiné par la commande find à un moment donné.

Voici quelques unes des options de sélection :
- name motif : vrai si le motif s'applique sur le nom du fichier courant ;
- user nom_utilisateur : vrai si le fichier courant appartient à l'utilisateur nom_utilisateur ;
- mtime n : vrai si le fichier a été modifié dans les n derniers jours (+n pour exprimer "n et plus" et -n pour exprimer "n et moins").

Les éléments de l'expression peuvent être connectés par les opérateurs logiques suivants :
- négation : ! ;
- et : simple juxtaposition des éléments ;
- ou : -o ;

Dans ce cas, il est nécessaire d'entourer l'expression complète avec des parenthèses (qu'il faut précédées d'une contre-barre pour qu'elles ne soient pas interprétées par le shell).

Les actions à effectuer sur les fichiers sélectionnés sont :
- print : affiche le nom du fichier courant ;
- exec com : exécute la commande sur le fichier courant ; com est terminée par le marqueur
\; et le paramètre spécial {} désigne le fichier courant.

Ex :

```
perrot@von-neumann>find . -name 'fic?' -print
./rep1/fic1
./rep1/fic4
./fic1
./fic2
 ./rep3/fic1

perrot@von-neumann>find rep1 rep2 \( -name 'fic?' -perm 755
\) -print -exec rm{} \;
rep1/fic1
rep2/fic5
```

Note : on peut utiliser la commande grep comme filtre pour éliminer les informations inintéressantes produites par une commande précédente comme dans l'exemple suivant :

```
find . -name fic* -exec cat {} \; | grep est
```

(affiche les lignes des fichiers dont les noms commencent par la chaîne `fic`, qui appartiennent à l'arborescence de racine définie par le répertoire courant et qui contiennent le motif `est`.

7.3- Recherche des utilisateurs connectés : `who`

La commande who affiche la liste des utilisateurs connectés sur la machine où l'on travaille. Le nom d'utilisateur, le nom du terminal utilisé et la date de début de session de travail sont fournis.

Ex :

```
perrot@von-neumann>who
Lartis   ttyp1    Oct 23 22:25 (mistral.ERE.UMon)
perrot   ttyp3    Oct 24 08:52 (San-A.grbb.polym)
```

Une utilisation pratique de la commande who est la commande who am i qui permet de connaître le nom de l'utilisateur connecté à partir d'un terminal et qui l'a quitté.

Ex :

```
perrot@von-neumann>who am i
von-neumann!perrot   ttyp3    Oct 24 08:52 (San-
A.grbb.polym)
```

8- Éditeurs de textes

8.1- Présentation générale

Le but d'un éditeur de textes est de créer et de modifier des fichiers textes qui peuvent être des documents ou des programmes sources (programmes C, Fortran, etc.).
Il existe de nombreux éditeurs sous UNIX mais les 2 éditeurs standards généralement disponibles sont `ed` et `vi`. Le premier est un éditeur interactif fonctionnant en mode ligne : le fichier traité n'est pas affiché à l'écran et l'utilisateur doit donc fournir des requêtes en indiquant les adresses des lignes que l'éditeur doit traiter. On n'étudiera pas cet éditeur vu son manque de convivialité. Il faut quand même savoir que `ed` est un vieil éditeur qui a servi de référence pour d'autres éditeurs comme `vi`.

8.2- L'éditeur de textes vi

`vi` (Visual), contrairement à `ed`, est un éditeur pleine page puisqu'il affiche le fichier traité sur une page d'écran (au moins 24 lignes de 80 caractères).

Il possède 2 modes de travail :
- un mode commande dans lequel l'utilisateur spécifie les requêtes de traitement du fichier ;
- un mode insertion dans lequel tout ce qui est entré au clavier est écrit dans le tampon de mémoire associé au fichier.

Pour ouvrir un fichier existant ou pour créer un nouveau fichier, il suffit d'utiliser la syntaxe suivante : `vi nom_fichier.`
À l'appel de l'éditeur, on se trouve dans le mode commande. Plusieurs commandes d'insertion de texte permettent de passer en mode insertion, alors que pour passer du mode insertion au mode commande, on tape le caractère d'échappement `ESC (ESCAPE)`.
Les commandes sont nombreuses et ici, on ne verra que les plus importantes (voir en annexe, le manuel d'utilisation de `vi`). Elles sont de plusieurs types :

8.2.1- Commandes de déplacement du curseur

On parlera ici de ligne courante, la ligne dans laquelle se trouve le curseur.

`l` (ou ESPACE ou ∅) : déplacement d'un caractère vers la droite ;
`h` (ou BACKSPACE ou ♦) : déplacement d'un caractère vers la gauche ;
`k` (ou ¯) : déplacement d'un caractère vers le haut ;
`j` (ou ¬) : déplacement d'un caractère vers le bas ;
`0` : déplacement sur le caractère de la colonne 1 de la ligne courante ;
`$` : déplacement sur le dernier caractère de la ligne courante ;
`[n]G` : déplacement sur la n^{ème} ligne du fichier. Par défaut, n est le numéro de la dernière ligne ;
`/motifRETURN`: descendre sur la 1ère ligne contenant `motif` ;
`?motifRETURN`: remonter sur la 1ère ligne contenant `motif` ;
`n` : répète la dernière opération /motif ou ?motif ;
`CTRL-f` (Forward) : page suivante ;
`CTRL-b` (Backward) : page précédente ;
`w` (Word) : déplacement sur le début du mot suivant ;
`b` (Back) : déplacement sur le début du mot précédent ;
e (End) : déplacement sur la fin du mot courant ;

Toutes ces commandes (sauf `0, $` et `G`) peuvent être précédées d'un entier qui sert de facteur de répétition.

Ex : `3k` (fait remonter le curseur de 3 lignes) ; `2w` (déplace le curseur de 2 mots en avant).

Note : on voit en fin de fichier, la présence de tildes (~) en première colonne de chaque ligne : c'est pour indiquer que ces lignes ne font pas partie du fichier édité.

8.2.2- Commandes d'insertion

Toutes les commandes d'insertion de chaînes de caractères font passer en mode insertion et doivent être terminées par le caractère `ESC` pour repasser en mode commande.

`a chaîne` : `chaîne` insérée immédiatement après le curseur (Append);
`A chaîne` : `chaîne` insérée en fin de la ligne courante ;
`i chaîne` : `chaîne` insérée devant le curseur (Insert) ;
`I chaîne` : `chaîne` insérée au début de la ligne courante ;
`o chaîne` : `chaîne` insérée après la ligne courante (Open) ;
`O chaîne` : `chaîne` insérée avant la ligne courante.

8.2.3- Commandes de suppression

`x` : supprime le caractère pointé par le curseur ;
`X` : supprime le caractère précédant le curseur ;
`dd` : supprime la ligne courante ;
`D` : supprime la fin de la ligne courante (y compris le caractère pointé) ;
`[n]d[adr]` (Delete) : `n` est un éventuel facteur de répétition et `adr` est une éventuelle adresse dans le texte. Dans ce cas, la destruction de texte a lieu depuis le caractère sous le curseur jusqu'à :
- le caractère adressé ou le dernier caractère de l'entité adressée (si l'adresse est celle d'un mot par ex.) ;
- la ligne adressée en cas d'adressage de ligne (y compris cette ligne) ;
- le caractère précédent l'adressage par un motif.

Ex : `3dw` (détruit la fin du mot courant et les 2 mots suivants) ; `dG` (détruit la fin du fichier à partir de la ligne courante y compris) ; `d/fic` (efface les caractères entre la position du curseur et la 1ère chaîne `fic`).

8.2.4- Commandes de remplacement

`r` : remplace le caractère sous le curseur par le caractère tapé ensuite ;
`R` : remplace les caractères à partir du curseur par le texte entré après `R` et terminé par `ESC` ;
`s` : remplace le caractère sous le curseur par le texte qui suit `s`, terminé par `ESC`.
`[n]c[adr]` (Change) : fonctionne sur le même principe que la commande `d` (voir plus haut). Un caractère `$` est positionné pour indiquer la fin du texte à remplacer, matérialisant `adr`. Si la fin de la modification (`ESC`) est frappé avant d'atteindre le dollar, tous les caractères situés entre le curseur et le dollar sont effacés tandis que si le curseur dépasse le dollar, celui-ci s'efface et les caractères qui le suivent se poussent.
Ex : `2cw` (remplace depuis le curseur et le mot suivant); `c$` (remplace depuis le curseur jusqu'à la fin de la ligne).

8.2.5- Commandes de copie et de déplacement de blocs

Déplacer un bloc se fait par une requête `d` où il est placé dans un tampon. Puis, le curseur est positionné là où on veut insérer le bloc, insertion qui se fait avec la requête `p` (Put) après le curseur (ou sur la ligne suivante) ou avec la requête `P` avant le curseur (ou sur la ligne précédente).

Copier un bloc se fait par une requête `y` (Yank), d'utilisation similaire à la commande `d`, qui initialise le tampon sans ôter le bloc adressé du texte puis par une requête `p` ou `P` d'insertion à l'endroit où le curseur a été positionné.

8.2.6- Commandes de recherche et de remplacement

Pour effectuer une substitution dans tout le texte (ou seulement une partie), on peut procéder de 2 façons :

- avec une combinaison de recherche sur `motif` (`/motif` ou `?motif`), modification (`c`) et répétition (`n` pour répéter la dernière recherche et . pour répéter la dernière modification)
- avec une requête de substitution en mode ligne (empruntée à l'éditeur `ed`) : la requête `s`.

Ex : pour remplacer dans tout le texte `motif1` par `motif2`, on utilise la commande :

`:1,$s/motif1/motif2/g`
Ceci se fait sans demande de confirmation, demande de confirmation qui peut activée par l'utilisation de `/cg` au lieu de `/g`.

8.2.7- Commandes générales

`:w` : sauvegarde du fichier sous son nom d'entrée ;
`:w nom_fichier` : sauvegarde du fichier sous le nom `nom_fichier` ;
`:q` : sortie de l'éditeur (seulement possible si le texte n'a pas été modifié depuis la dernière sauvegarde ;
`:q!` : sortie de l'éditeur (même si le texte a été modifié) ;
`:r nom_fichier` : introduction du fichier nom_fichier dans le texte après la ligne courante ;
`:u` (Undo) : annulation la dernière modification du texte ;
`:f` : obtention des caractéristiques du fichier édité (nom du fichier, numéro de ligne pointée, etc.) ;
`:!nom_commande` : exécution d'une commande shell.

8.3- Autres éditeurs

L'éditeur `vi` a une mauvaise réputation : il est jugé difficile et mal commode. Cette idée vient surtout du fait qu'aucune commande n'est disponible en mode insertion à part la commande `ESC` de sortie du mode d'insertion. En particulier, on ne peut déplacer le curseur en mode insertion. Toutefois, il ne faut pas perdre de vue que l'éditeur `vi` est présent sur la plupart des systèmes UNIX et qu'il est quelque fois le seul éditeur accessible.
Mais, si on en a la possibilité, il est préférable d'utiliser des éditeurs graphiques (fonctionnant sous X Windows par ex.) qui permettent, entre autres, l'utilisation de la souris et de menus.
Un des éditeurs les plus puissants en terme de fonctionnalités est `emacs`. De plus, il peut être utilisé en mode texte (certains terminaux n'offrent que ce mode) ou sous environnement graphique.

9- Bibliographie

- La programmation sous UNIX de Jean-Marie Rifflet, 2ème édition chez Mc-Graw-Hill

- Le système UNIX de Steve Bourne aux éditions "InterEditions" ;

- la liste des livres sur UNIX : rtfm.mit.edu:/pub/usenet/news.answers/books/unix.

- Big Dummy's Guide to the Internet par EFF (Electronic Frontier Foundation) (version texte disponible sur de nombreux sites FTP anonymes comme ftp.eff.org:/pub/Net_info/EFF_Net_Guide/netguide.*) ;

- Zen and the Art of the Internet de Brendan P. Kehoe (version PostScript disponible sur de nombreux sites FTP anonymes : ftp.cs.widener.edu:/pub/zen/zen-1.0.PS).