Cours Unix 1

Michel Mauny

ETGL

D'après http://www.francenet.fr/~perrot/cours_unix.html

Cours disponibles sur http://quincy.inria.fr/courses/unix/

Organisation du cours

- Intervenants
- Salles
- Horaires
- Objectifs du cours
  - Acquérir une aisance dans un environnement Unix
  - Savoir lire/écrire des scripts shell
  - Comprendre différents aspects d'Unix au travers d'un shell
- Examen
  - Compréhension
  - Scripts ou éléments de scripts

Plan du cours 1

1. Présentation d'Unix
   (a) Historique
   (b) Fonctionnalités
   (c) Unix aujourd'hui
2. Connexion et déconnexion
3. Principales commandes
   (a) Organisation et manipulation des fichiers
   (b) Éditeurs
   (c) Communications

Présentation d'UNIX

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.

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.

Historique (suite)

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: System V et BSD.

Fonctionnalités

Un système d'exploitation fournit:

- Isolation du matériel vis-à-vis des applications
- Bibliothèque commune pour les programmes

Les fonctions principales d'UNIX sont:

- Gestion des ressources de l'ordinateur
- Gestion des données
- Communication entre utilisateurs
- Environnement de programmation

Il donne de plus accès à un riche langage de commandes: le *shell*.

Gestion des ressources de l'ordinateur

Système d'exploitation multi-tâches et multi-utilisateurs:

- temps d'utilisation du processeur de l'ordinateur réparti entre différentes tâches (exécution simultanée de programmes)
- traitement des commandes de plusieurs utilisateurs en même temps.

Unix doit répartir les ressources entre les différentes tâches et utilisateurs de façon transparente pour ces derniers.

Gestion des ressources de l'ordinateur

Gestion des données Organisation, maintenance et accès aux unités de stockage (mémoire, disques durs, bandes magnétiques, etc.)

Communication entre utilisateurs le courrier électronique, les transferts de fichiers dont l'utilisation est expliquée plus loin.

Environnement de programmation Compilateurs (C, Fortran, ...), éditeurs de textes, outils d'aide à la programmation (débogueurs, etc.).

Unix aujourd'hui

Les Unix libres:

- Linux
- FreeBSD
- ...

OS pour programmeurs. Serveurs. Extrêmement configurable. Documenté. Fournit aujourd'hui des interfaces conviviales et comprend des outils bureautiques.

Linux

Fonctionne sur la majorité des architectures.

Commun sur PC, avec souvent un *dual-boot*:

- au démarrage, Master Boot Record: Linux ou Windows?
- si Linux, exécution du *boot sector* de la partition Linux qui lance Linux
- sinon, lancement de Windows.

Permet d'avoir les 2 OS sur la même machine.

Connexion et déconnexion

Hypothèse: l'utilisateur a déjà été enregistré sur le système. Son identité est donc présente dans des fichiers particuliers gérés par l'administrateur du système.

Procédure de connexion

Celle-ci peut être scindée en 2 phases:

- connexion logique
- initialisation de la session de travail

Connexion logique

La connexion logique doit avoir lieu entre le

- terminal (écran + clavier, ou autre ordinateur (stations UNIX, PCs, Macintosh, etc.))
- et l'ordinateur sur lequel on veut travailler.

Dans le cas le plus simple de la liaison directe, il suffit d'allumer le terminal.

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;
- 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 ne puisse l'apercevoir.

Initialisation de la session de travail (suite)

On a alors accès ou bien:

- à un environnement graphique, qui permet notamment d'avoir accès à des fenêtres *shell*;
- à un mode *console* (un *shell*)

En shell, 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: `$` ou `nom_utilisateur@nom_machine$`)

Changement de mot passe

Avec la commande `passwd` (ou `yppasswd`). Dépend du type de configuration du système d'enregistrement des utilisateurs:

- Si utilisateurs locaux d'une machine donnée, ils doivent utiliser la commande `passwd`.
- Si utilisateurs enregistrés par le système des pages jaunes (Yellow Pages), encore appelé NIS (Network Information Service), ils doivent utiliser la commande `yppasswd`. Le système YP/NIS permet entre autres à l'utilisateur, d'accéder à un compte unique et ceci à partir de différentes machines.

Changement de mot passe

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.

```
$ yppasswd
Changing NIS password for tom on sparc34.
Old password:
New password:
Retype new password:
NIS entry changed on sparc34
```


Changement de mot passe

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).

Qualités d'un ≪bon≫ mot de passe

Sécurité! (En environnement multi-utilisateurs ou lorsque la machine est connectée au réseau.)

Il est 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 appartenir à aucun dictionnaire;

Qualités d'un ≪bon≫ mot de passe

- 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 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`.

Procédure de déconnexion

Dépend du type de session qui a été ouverte.

- Si environnement graphique, menu `logout` (ou `exit`). Parfois étape de déconnexion ci-dessous nécessaire.
- En mode console, `logout` ou `exit`.

Commandes

Interpréteur de commandes

Le *shell*. *shell* = coquille

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 ses variantes `ksh`, `bash`, ...) et le C-shell (`csh`).

Le choix du (la variante du) shell activé à la connexion est fait à l'enregistrement de l'utilisateur dans le système.

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 (<SPACE>);
- 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.

Syntaxe des commandes

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`

Syntaxe des commandes

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.

Manuel des commandes

Aide (sur les règles d'utilisation, sur les fonctionnalités d'une commande): on peut utiliser l'aide en ligne grâce à la commande `man` (comme `man`*ual pages*).

`man nom_commande`

Ex:

- `man ls`
- `man man`

Manuel des commandes

Note: l'affichage des pages de manuel se fait à l'aide de la commande `more`, afficheur page par page. 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 pages de manuel à l'aide de la commande suivante:

`man -k mot-clé`

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

Confort: édition de ligne

On peut configurer `ksh` pour qu'il soit doté d'édition de ligne:

`set -o emacs`

Il est alors aisé de:

- remonter dans l'historique `^P` (*Previous*)
- descendre dans l'historique `^N` (*Next*)
- se déplacer vers le début de la ligne `^B` (*Backward*)
- se déplacer vers la fin de la ligne `^F` (*Forward*)

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 `*` (astérisque) qui peut remplacer n'importe quelle chaîne de caractères, y compris la chaîne vide.

Caractères génériques

Ex:

```
$ ls
fic fic1 fic2 fic3 fic33 mbox rep1 rep2 rep3
$ ls fic?
fic1 fic2 fic3
$ ls fic*
fic fic2 fic33 fic1 fic3
$ ls fic??
fic33
```

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

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`

Redirection des entrées/sorties

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`

  `date >> poub`

- bénéficier de l'édition de ligne dans chaque instance de `ksh`:

  `echo "set -o emacs" >> ~/.kshrc`

Redirection des entrées/sorties

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`

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 vue précédemment)
- `ls /etc | more`

Organisation des fichiers

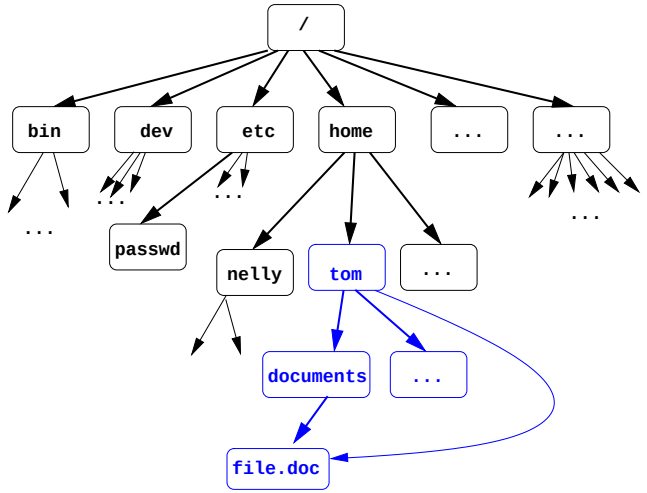
Arborescence d'UNIX

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.

Un dessin



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 utilise 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/tom/fic1` et `/home/tom/rep1/fic1`.

Nom relatif

Le répertoire courant (*working directory*):

- noté ≪.≫
- identifié par la commande `pwd`
- au début de la session c'est le répertoire personnel de l'utilisateur.

Le ≪père≫:

- noté ≪..≫
- correspond à un niveau supérieur dans l'arborescence

Nom relatif

Ex:

```
$ pwd
/home/tom
$ cd rep3
$ pwd
/home/tom/rep3
$ ls -la
total 3
drwx------ 2 tom 512 Sep 21 21:30 .
drwxr--r-x 8 tom 512 Sep 21 21:30 ..
-rw------- 1 tom 50 Sep 21 21:30 fic1
$ cd ../rep2
$ pwd
/home/tom/rep2
$ cd /home/tom/rep1
$ pwd
/home/tom/rep1
```


Nom relatif

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.

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.

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).

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*).

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:

`r w x r w x r w x`

user *group* *others*

Visualisation des droits d'accès

Ex:

```
$ ls -l
total 7
-rw------- 1 tom 49 Sep 18 12:37 fic
-rw------- 1 tom 50 Sep 18 12:35 fic1
-rw------- 1 tom 50 Sep 18 12:37 fic2
-rw------- 1 tom 50 Sep 18 12:37 fic3
-rw------- 1 tom 51 Sep 18 12:38 fic33
```

Modification des droits d'accès

Seul le propriétaire d'un fichier (ou l'administrateur) 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`.

Modification des droits d'accès

`chmod [qui]opaccès nom_fichier`

- `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.

Modification des droits d'accès

Ex:

```
$ ls -la rep3
total 3
drwx------ 2 tom 512 Sep 21 21:30 .
drwxr--r-x 8 tom 512 Sep 21 21:30 ..
-rw------- 1 tom 50 Sep 21 21:30 fic1
$  chmod u-r rep3
$ ls -la rep3
rep3 unreadable
```


Modification des droits d'accès

```
$ ls -la fic2
-rw-rw---- 1 tom 101 Sep 18 19:15 fic2
$ fic2
fic2: Permission denied.
$  chmod +x fic2
$ ls -la fic2
-rwxrwx--x 1 tom 101 Sep 18 19:15 fic2
$ fic2
Il est: 11:44:37
```

Modification des droits d'accès

Notes:

- 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).
- il existe d'autres modes:
  - `s` (*set user (or group) id on execution*): permet à l'exécutant de la commande contenue dans le fichier de se sustituer à son propriétaire;
  - `t` *sticky bit* sur un répertoire: son sens dépend de la version d'Unix.

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 que l'on donne par défaut aux nouveaux fichiers.

Initialisation des droits d'accès

La syntaxe est la suivante: `umask masque`.

`masque`: valeur octale qui joue le rôle de masque sur les droits d'accès d'un fichier à sa création. Les droits d'accès sont calculés par:

`mode-par-défaut & (~masque)`

avec `&`, le ET logique et `~` le NON logique.

	`mode-par-défaut`
Fichiers:	`rw-rw-rw-` = 110110110
Répertoires:	`rwxrwxrwx` = 111111111

Initialisation des droits d'accès

Ex:

- `umask 22`. Le masque est ici 022, soit en codage binaire 000 010 010. Les fichiers créés auront la protection `rw-r--r--`, les répertoires `rwx-r-xr-x`.
- `umask 77`. Le masque est ici 077, soit en codage binaire 000 111 111. Les fichiers créés auront la protection `rw-------`, les répertoires `rwx------`.

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.

Manipulation des fichiers

Liste des fichiers

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

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
```


Création d'un fichier

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).

Destruction d'un fichier

`rm nom_fichier`

Notes:

- Quelquefois, le système demande confirmation avant d'effacer un fichier.
- Il est impossible de récupérer un fichier effacé et ceci rend les sauvegardes indispensables (il faut donc contacter l'administrateur du système pour s'informer à ce sujet).
- Il faut être *très prudent* avec la commande `rm`.

Visualisation d'un fichier

Plusieurs commandes 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, <SPACE> pour la page suivante et q pour terminer l'affichage).

Copie d'un fichier

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


Renommage ou déplacement d'un fichier

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

Création d'un répertoire

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

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)

Éditeurs de textes

Présentation générale

But: créer et de modifier des fichiers texte 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`.

- `ed` éditeur interactif (on ne l'étudiera pas) fonctionnant en mode ligne
- `vi` pleine page

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.

L'éditeur de textes vi

Ouvrir un fichier existant ou pour créer un nouveau fichier:
`vi nom_fichier`.

`vi`*: modes*

En mode commande

- `i`, `a`: passe en mode insertion
- flèches: navigation
- `:` attend une commande en bas de la fenêtre.

En mode insertion:

- <ESC> (échappement): repasser en mode commande,

Commandes nombreuses: on ne les détaille pas toutes oralement.

- Commandes générales: voir la sprochaine planche
- Autres commandes: les 10 planches suivantes

`vi`*: 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 la ligne courante, etc.);
- `:!nom_commande` exécution d'une commande shell.

vi: commandes de déplacement du curseur

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

- `l` (ou <SPACE> ou $\rightarrow$): déplacement d'un caractère vers la droite;
- `h` (ou <BACKSPACE> ou $\leftarrow$): déplacement d'un caractère vers la gauche;
- `k` (ou $\uparrow$): déplacement d'un caractère vers le haut;
- `j` (ou $\downarrow$): déplacement d'un caractère vers le bas;

vi: commandes de déplacement du curseur

- `0`: déplacement sur le 1er caractère de la colonne (colonne 0) de la ligne courante;
- `$`: déplacement sur le dernier caractère de la ligne courante;
- `[n]G`: déplacement sur la `n`-ième ligne du fichier. Par défaut, `n` est le numéro de la dernière ligne;

vi: commandes de déplacement du curseur

- `/motif<RETURN>`: descendre sur la 1ère ligne contenant `motif`;
- `?motif<RETURN>`: remonter sur la 1ère ligne contenant `motif`;
- `n`: répète la dernière opération `/motif` ou `?motif`;

vi: commandes de déplacement du curseur

- `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;

vi: commandes de déplacement du curseur

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.

vi: 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.

vi: 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);

vi: commandes de suppression

- `[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.

vi: commandes de suppression

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 (comprise));
- d/fic (efface les caractères entre la position du curseur et la 1ère chaîne fic).

vi: autres commandes

- remplacement: r, R, s, c
- déplacement (sorte de copier/couper/coller): d, p, P, y
- recherche, recherche/remplacement: /motif, ?motif, s, ...

  :1,$s/motif1/motif2/g

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 quelquefois 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 notamment l'utilisation de la souris et de menus.

Autres éditeurs: emacs/xemacs

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.

Communication

Le réseau Internet

Communication rime avec Internet

Internet:

- constitué de millions d'ordinateurs interconnectés entre eux par divers types de liaisons matérielles
- communiquent entre eux à l'aide du protocole TCP/IP (Transmission Control Protocol / Internet Protocol)
- né il y a une vingtaine d'années

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` $(1 \geq \texttt{xxx} \geq 255)$

  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.

- symbolique: `nom_machine.domaine_internet`. Associée à l'adresse numérique.

  `www.lip6.fr`, `www.inria.fr`, `www.google.com`

Adresse d'une machine

Note: on peut obtenir le nom de la machine sur laquelle on travaille avec la commande `hostname`.

Ex:

```
$ hostname
sparc34
```

Adresse d'un utilisateur

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

Ex:

`tom.pouce@inria.fr`, `tom@pouce.net`, `tom@machine.domain.edu`

Notes:

- insensibles à la casse
- domaine non nécessaire si sur le même domaine

*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`.

Envoi de courrier

Pour cela, la commande à utiliser est:
`mail adresse_destinataire`

Entrer le texte. Terminer par un point en début de ligne (ou `^D`).

Ex:

```
mail tom
1ère ligne
2ème ligne
.
dernière ligne
^D
```


Lecture de courrier

Pour consulter son courrier, il faut taper la commande:
`mail [options]`.

Ex:

```
$ mail
Mail version SMI 4.0 Wed Oct 13 18:37:02 PDT 1993
Type ? for help.
"/var/spool/mail/tom": 2 messages 2 new
>N 1 tom Sun Sep 18 20:43 11/316 test
N 2 tom Sun Sep 18 20:45 11/312 test2
&
```

Lecture de courrier

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 ?).

Lecture de courrier

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
```


Lecture de courrier

```
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.
&
```


Lecture de courrier

Pour lire le courrier rangé dans un fichier particulier autre que `/var/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.

Connexion sur une machine distante

```
telnet adresse_machine
rlogin adresse_machine
```

Avoir un compte sur cette machine distante, et s'identifier.

Ex:

```
$ telnet sparc34
Trying 123.234.12.34 ...
Connected to sparc34.
Escape character is '^]'.
SunOS UNIX (sparc34)
login: tom
Password:
```


Transfert de fichiers

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

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.

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.

- on fournit comme nom d'utilisateur, le nom `anonymous` (ou simplement `ftp`)
- et comme mot de passe, son adresse e-mail

Accès restreint: vision d'une partie spécifique de la hiérarchie de fichiers.

FTP anonymes

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.
```


FTP anonymes

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;

FTP anonymes

- mode de transfert: `ascii` (par défaut) ou `binary`;
- transfert multiple: `mget` et `mput`. 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`.

Commande finger

Permet d'avoir de l'information sur un utilisateur qui a un compte sur une machine particulière.

```
finger nom_utilisateur@adresse_machine
```

Ex:

```
$ finger tom@sparc34.pouce.net
[sparc34.pouce.net]
Login name: tom In real life: Tom Pouce
Mail to tom goes to tom.pouce@hautemail.com
Project:
Plan: No plan
```

Commande finger

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.

Commande ping

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 sparc34
sparc34 is alive
```


Commande ping

Permet aussi de déterminer l'adresse numérique d'une machine avec la syntaxe suivante: `ping -s adresse_machine`

Ex:

```
$ ping -s sparc44
PING sparc44: 56 data bytes
64 bytes from sparc44 (123.234.12.129): icmp_seq=0. time=2. ms
64 bytes from sparc44 (123.234.12.129): icmp_seq=1. time=4. ms
^C
```


Recherche des utilisateurs connectés: who

Affiche la liste des utilisateurs connectés sur la machine.

Ex:

```
$ who
max ttyp1 Oct 23 22:25 (sparc33.pouce.net)
tom ttyp3 Oct 24 08:52 (wstation.business.com)
```

Une utilisation pratique de la commande `who` est la commande `who am i` qui permet de connaître le nom de l'utilisateur connecté au terminal courant.

Ex:

```
$ who am i
sparc34!tom ttyp3 Oct 24 08:52 (wstation.business.com)
```

Divers

Arrêter la commande courante taper le caractère `^C`

Lancer une application en tâche de fond: `command ... &`

Voir les processus `ps`

Tuer un processus `kill numéro_de_processus`

Imprimer `lpr`

Bibliographie

- La programmation sous UNIX de Jean-Marie Rifflet, 2ème édition chez Mc-Graw-Hill
- Le système UNIX de Steve Bourne aux éditions InterEditions;
- Probablement beaucoup d'autres ...