Petit viatique pour survivre en UNIX

Valérie Ménissier-Morain

Septembre 1998

1 Généralités

UNIX est un « système d'exploitation », c'est-à-dire un ensemble de programmes permettant d'utiliser l'ordinateur. Entre autres c'est lui qui gère l'accès aux différentes ressources (mémoire, affichage, ...) de l'ordinateur. On dit qu'il est « multi-tâches » car il permet de faire fonctionner plusieurs programmes en même temps, et « multi-utilisateurs » car il permet à plusieurs utilisateurs de travailler en même temps sur la même machine.

Dans ce petit document il est hors de question d'être exhaustif, ce qui nécessiterait plusieurs livres. Pour chaque commande mentionnée, on ne présente que le comportement de base, les options fondamentales et comment obtenir de l'aide ou en apprendre davantage en ligne.

2 Comment commencer et terminer une session ?

2.1 Connection

Pour pouvoir travailler sur une machine UNIX il faut d'abord s'identifier. Les machines connaissent les utilisateurs par un nom appelé « nom de login ».

2.1.1 Nom de login

Pour se connecter (ou se « loguer ») il faut donner son nom de login et son mot de passe dans la fenêtre présente à l'écran (si l'écran n'affiche pas de fenêtre, il suffit de taper une touche pour la faire apparaître). Votre « nom de login » ainsi que votre mot de passe vous sont communiqués au début de l'année scolaire.

Attention, Unix est un système dit *case sensitive*, c'est-à-dire qu'un caractère en minuscule est différent de ce même caractère en majuscule.

2.1.2 Mot de passe et sécurité

Votre mot de passe est SECRET, vous êtes la SEULE personne censée le connaître. C'est le moyen pour la machine de vérifier qui vous êtes et cela vous engage LÉGALEMENT. Vous pourriez avoir de graves ennuis si quelqu'un d'autre se faisait passer pour vous à des fins délictueuses.

Vous devez vous conformer à la charte que vous avez signée à l'ouverture de votre compte. Vous vous êtes engagé vis-à-vis de l'université et au regard de la loi.

Il faut donc ne jamais donner votre mot de passe à quelqu'un d'autre et éviter de s'éloigner de la station sur laquelle on est connecté. Une personne malveillante peut profiter de votre éloignement pour vous nuire ou nuire en votre nom, sous votre responsabilité.

2.2 Déconnection

Avant de partir, on se « déconnecte » en cliquant sur le bouton de gauche de la souris (soit sur l'étiquette Start en bas à gauche de l'écran, soit sur le fond d'écran) et en sélectionnant `Exit Fvwm` dans le menu puis `Yes, Really Quit` dans le sous-menu **et on laisse la machine allumée comme elle l'était en entrant. On n'éteint pas une machine sous Linux en s'en allant.**

3 Ce que vous voyez sur l'écran

Lorsque vous êtes connecté, apparaissent un certain nombre de fenêtres : la fenêtre « Console » sert à afficher les différents messages envoyés par le système. Vous pouvez ne pas la regarder, sauf si vous avez l'impression que quelque chose ne se passe pas normalement. Dans la plupart des cas vous aurez un message d'erreur dans cette fenêtre. L'autre fenêtre est une fenêtre qui vous permettra de lancer des programmes, elle se comporte comme un « terminal » indépendant. C'est dans cette fenêtre (ou dans une autre de même type) que vous lancerez les programmes qui ne sont pas dans les menus en tapant simplement leur nom suivi des arguments éventuels. Ainsi pour commencer une session CAML, vous tapez `camllight`. Pour compiler un programme C ce sera du genre `cc` *`arguments`*.

3.1 La manipulation des fenêtres

Attention, a priori chaque bouton de la souris active une action différente quand on clique dessus.

Vous pouvez bien entendu manipuler les fenêtres à l'aide de la souris : en haut de la fenêtre se trouve une *barre de titre* où vous pouvez voir le titre de la fenêtre et 3 boutons.

- Le bouton le plus à droite permet de donner à la fenêtre la taille de l'écran puis de revenir à la taille normale lorsqu'on clique dessus avec le bouton de gauche de la souris.
- Le bouton juste à gauche de ce dernier permet d'*iconifier* la fenêtre. Pour la *désiconifier*, il suffit de cliquer 2 fois sur l'icône.
- En cliquant dans la barre de titre et en maintenant le bouton de la souris enfoncé, vous pouvez déplacer la fenêtre.
- Enfin le bouton le plus à gauche fait apparaître un menu permettant de manipuler la fenêtre. Ce menu apparaît d'ailleurs en cliquant avec le bouton droit de la souris n'importe où sur la bordure de la fenêtre.

On change la taille de la fenêtre en cliquant sur le bord du côté que l'on veut déplacer et en tirant ce bord dans la direction souhaitée en maintenant le bouton de la souris enfoncé.

Vous pouvez essayer de voir ce qui arrive si l'on clique avec un des boutons de la souris sur la barre de titre simplement, puis en maintenant la touche `Ctrl` enfoncée.

De même, essayer de voir ce qui arrive si l'on clique sur l'un des boutons de la souris sur le fond d'écran.

4 Ordres UNIX de base

4.1 Le système de fichiers

Les informations sont rangées sur le disque dans des *fichiers*. Les fichiers sont eux-mêmes rangés dans des répertoires (ou *directory* en anglais). En fait les répertoires sont des fichiers particuliers contenant une liste de fichiers, ils sont donc eux aussi rangés dans des répertoires. Le seul répertoire qui ne soit pas dans un autre répertoire est appelé *racine* et noté `/`. Cette organisation est représentable par une structure d'« arbre » renversé esquissé sur la figure **??**.

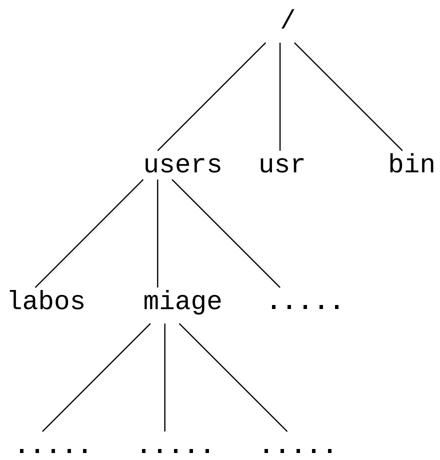


FIG. 1 – Un extrait de l'arborescence des fichiers gèrés par Linux dans les salles DEUG

4.1.1 Noms de fichiers, chemins

On désigne un fichier soit de manière « absolue », soit de manière « relative ». Quand on donne son nom de manière absolue, on donne la liste des répertoires et sous-répertoires pour accéder au fichier depuis la racine séparés par des `/`. Par exemple, dans l'arborescence de la figure **??**, `/home/Etu/mio1/mio1m1`) ; quand on le donne de manière relative on donne de la même manière le chemin depuis le répertoire de travail, sans commencer par `/`. Par exemple, si le répertoire de travail est `/home/Etu/`, le nom relatif du fichier précédent sera `mio1/mio1m1`.

4.1.2 Répertoires

À tout moment, chaque utilisateur se situe dans un *répertoire de travail*, dont on peut avoir le nom par la commande `pwd` (comme *print working directory*).

Chaque utilisateur dispose d'un répertoire principal, où il se trouve par défaut en début de session (par exemple, dans l'arborescence de la figure **??**, `/home/Etu/mio1/mio1m1`) dans lequel il met ses fichiers. Ce répertoire est aussi appelé *home directory* et noté « ~ ».

On obtient la liste des fichiers contenus dans un répertoire par la commande `ls` (comme *list*). Pour se déplacer dans l'arborescence, on dispose de la commande `cd` (comme *change directory*).

Nous reviendrons plus tard sur ces trois commandes.

4.1.3 Les droits d'accès

UNIX étant un système multi-utilisateurs, il dispose d'un système permettant de contrôler qui a accès à quels fichiers.

Pour cela, comme le montre la figure **??** chaque fichier dispose :
- d'un propriétaire
- d'un groupe
- de différents droits d'accès.

De plus chaque utilisateur appartient à un ou plusieurs groupe(s). Pour voir les droits d'un fichier il suffit d'utiliser l'option `-l` de `ls`, c'est-à-dire taper `ls -l` *`argument`*. On obtient alors, pour chaque fichier, une ligne de la forme :

`-rw-r--r-- 1` *`user`* `6876 Sep 10  11:04` *`fichier`*

Cette ligne donne un certain nombre d'informations : en partant de la droite

1. Le nom du fichier (*`fichier`*)

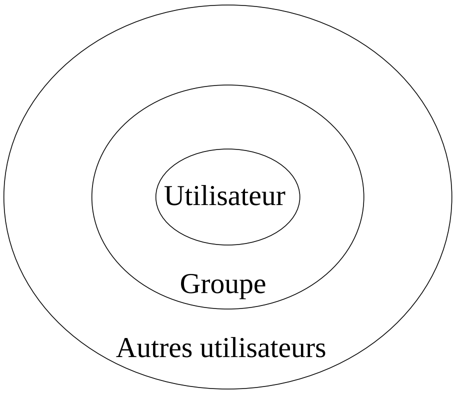


FIG. 2 – Les différentes classes d'utilisateurs vis-à-vis des droits d'accès à un fichier.

2. La date et l'heure de dernière modification, ou éventuellement seulement la date si elle est assez éloignée. (`Sep 10 11 :04`)
3. La taille en octets (`6876`)
4. Le nom du propriétaire du fichier. (*`user`*)
5. Un nombre que vous pouvez ignorer (`1`)
6. Les droits et le type du fichier (`-rw-r--r--`). Ce champ est représenté par une suite de dix caractères. Le plus à gauche donne le type du fichier : `-` si c'est un fichier « normal » et `d` si c'est un répertoire (`d` comme *directory*). Ensuite viennent 9 caractères donnant les droits proprement dits. Les 3 premiers donnent les droits pour le propriétaire du fichier, les 3 suivants pour les gens appartenant au même groupe que le propriétaire du fichier, et les 3 derniers pour les autres utilisateurs. Ces trois caractères sont :
   (a) `r` ou `-`. Un `r` indique que le fichier est accessible en lecture, un `-` indique que le fichier n'est pas permis en lecture.
   (b) `w` ou `-`. Un `w` indique que l'utilisateur concerné peut écrire sur ce fichier.
   (c) `x` ou `-`. Un `x` indique que le fichier contient un programme que l'utilisateur peut exécuter.

   Ces permissions ont une signification spéciale quand le fichier est un répertoire : écrire dans un répertoire signifie y créer ou y détruire des fichiers. Exécuter un répertoire signifie y accéder par la commande `cd`. En général, les droits sur un répertoire seront `drwxr-xr-x` ce qui signifie que seul le propriétaire peut y créer ou détruire des fichiers, mais que tout le monde peut aller voir ce qui se trouve dans ce répertoire, ou (mais très rarement) `drwx------`, ce qui signifie que c'est un répertoire où seul le propriétaire peut aller. Il s'agit typiquement de répertoires confidentiels tel que celui où vous sauverez les courriers électroniques que vous souhaiterez conserver.

   Par défaut, vos fichiers (excepté les fichiers de courrier créés par certains utilitaires de courrier électronique) sont créés avec les droits `-rw-r--r--`.

   Vous pouvez toujours modifier ces droits *a posteriori* avec la commande `chmod` (pour *change mode*). On peut appliquer cette commande à un fichier unique ou une liste de fichiers.

   On indique les modifications à apporter aux droits de la façon suivante : `chmod` *`mode fichier`*. L'argument *`mode`* est décrit par trois composantes : la catégorie des gens dont on modifie les droits sur le fichier, la nature de la modification (ajout, retrait, redéfinition) des droits, la nature des droits à modifier (lecture, écriture ou exécution). La figure **??** récapitule le contenu de cet argument. Par exemple si le fichier *`fichier`* a été créé avec les droits par défaut `-rw-r--r--` et si on veut retirer les droits de lecture de ce fichier aux membres du groupe et aux autres, on écrit `chmod go-r` *`fichier`*.

Un ou plusieurs	Un seul	Un ou plusieurs
`u` (utilisateur) `g` (groupe) `o` (autres)	`+` (ajouter) `-` (retirer) `=` (définir)	`r` (lire) `w` (écrire) `x` (exécuter)

FIG. 3 – L'argument *mode* de la commande `chmod` à appliquer à un fichier.

4.1.4 Quelques commandes de base sur les répertoires et les fichiers

- `pwd` : Indique le nom du *répertoire de travail.*
- `cd` : Permet de se déplacer dans un fichier.
  - `cd` vous renvoie dans votre *home directory*,
  - `cd ..` renvoie au répertoire « père » de votre répertoire courant (c'est-à-dire celui situé juste au-dessus dans la hiérarchie)
  - `cd` *`répertoire`* va dans le répertoire de nom *`répertoire`*.
- `ls` : Donne la liste des fichiers.
  - Sans argument, donne la liste des fichiers du répertoire courant.
  - Avec une liste de noms de fichiers comme arguments, donne la liste des fichiers, et pour les répertoires, la liste des fichiers qu'ils contiennent.

  Cette commande dispose de nombreuses options. Voici les plus couramment utilisées :
  - `-l` (pour *long*) donne en plus les droits et propriétaires des fichiers listés.
  - `-a` (pour *all*) fait apparaître un certain nombre de fichiers et répertoires supplémentaires dont le nom est préfixé par un point tel que `.twmrc`. Il s'agit de fichiers de configuration d'un certain nombre de programmes. On les trouve essentiellement à la racine de son répertoire.
  - `-F` suffixe le nom des répertoires par le caractère `/`, les fichiers exécutables par le caractère `*` et les liens symboliques par le caractère `@`.
  - `-t` (pour *time*) affiche les fichiers selon leur date de dernière modification, du plus récemment modifié au plus anciennement utilisé.
  - `-r` (pour *reverse*) permet d'afficher les fichiers dans l'ordre inverse (que le critère d'ordonnancement soit l'ordre alphabètique, la date de dernière modification, etc.)
  - `-R` (pour *Récursif*) affiche tous les fichiers ordinaires puis le contenu des sous-répertoires, et ainsi de suite.

  On peut utiliser plusieurs options simultanément comme le montre la session suivante :

```
$ ls
$ touch f1 f2 f3 f4 f5
$ chmod +x f3 f5
$ mkdir D1 D2
$ mkdir D1/D11 D1/D12
$ ln -s f1 f6
$ ls
D1  D2  f1  f2  f3  f4  f5  f6
$ ls -F
D1/  D2/  f1   f2   f3*  f4   f5*  f6@
$ ls -l -F
total 2
drwxrwxr-x   2 user   group      1024 Oct  3 20:24 D1/
drwxrwxr-x   2 user   group      1024 Oct  3 20:24 D2/
-rw-rw-r--   1 user   group         0 Oct  3 20:24 f1
-rw-rw-r--   1 user   group         0 Oct  3 20:24 f2
```

```
-rwxrwxr-x   1 user  group        0 Oct  3 20:24 f3*
-rw-rw-r--   1 user  group        0 Oct  3 20:24 f4
-rwxrwxr-x   1 user  group        0 Oct  3 20:24 f5*
lrwxrwxrwx   1 user  group        2 Oct  3 20:24 f6 -> f1
$ ls -a
.   ..  D1  D2  f1   f2   f3  f4   f5  f6
$ ls -t
f6  D1  D2  f1   f2   f3  f4   f5
$ ls -r -t
f1   f2   f3  f4   f5  D1  D2  f6
$ ls -R -F
D1/  D2/  f1   f2   f3*  f4   f5*  f6@

D1:
D11/  D12/

D1/D11:

D1/D12:

D2:
```

- `cat` *fichier* : affiche le contenu du fichier à l'écran.
- `more` *fichier* : affiche le contenu du fichier à l'écran en s'arrêtant quand l'écran est plein. On repart en tapant sur la barre d'espace, ou on arrête en tapant sur `q`.
- `mkdir` *nom* (pour *make directory*) : crée le répertoire *nom*. Il est très fortement recommandé d'avoir une bonne organisation de fichiers. Ne mélangez pas des fichiers n'ayant rien à faire ensemble, créez des répertoires différents pour les différents cours et groupez-les de manière logique. Par exemple, si vous programmez en plusieurs langages, ayez un répertoire `Programmes`, puis à l'intérieur de ce répertoire des répertoires `Pascal`, `C`, `Caml`, etc.
- `rm` *fichier* (pour *remove*) : efface le ou les *fichier(s)* spécifiés.
  `rm` n'efface pas les répertoires. Deux options utiles :
  - `-i` (pour *interrogation*) demande confirmation. Il est recommandé de mettre la ligne suivante :
    `rm () { /bin/rm -i "$@"; }`*
    dans son fichier `~/.bashrc`*personnel, ce qui fait que le système vous demandera confirmation chaque fois que vous utiliserez la commande `rm`, ce qui pourra vous évitez de faire des bêtises et aussi de supprimer tous les fichiers d'un répertoire sauf certains.
  - `-r` (pour *récursivement*) supprime récursivement les fichiers et répertoires indiqués.
- `rmdir` *répertoire* (pour *remove directory*) : efface le ou les répertoire(s) spécifiés. Il faut que ces répertoires soient vides. Il est souvent plus rapide d'utiliser `rm -r`.
- `cp` *nom1 nom2* (pour *copy*) : copie le fichier de nom *nom1* sous le nom *nom2*.
  L'option `-R` (pour *Récursif*) est très utile pour copier récursivement un répertoire.
- `mv` *nom1 nom2* (pour *move*) : change le nom du fichier *nom1* en *nom2*.
  nom1 et *nom2* peuvent être des répertoires. Il est alors plus naturel de considèrer l'action de `mv` comme un déplacement.

4.2 Communication entre programmes

UNIX permet à plusieurs programmes de s'exécuter simultanément, et dispose de moyens pour les faire communiquer. Le moyen le plus simple est le « pipe » (`|`). En gros quand on tape *commande1* `|` *commande2*, le résultat de *commande1* est envoyé comme entrée à *commande2*. Cela permet de faire des *filtres*. L'utilisation la plus courante est avec la commande `more` qui, lorsqu'elle n'a pas d'argument, affiche le texte qu'on lui entre en s'arrêtant lorsque l'écran est plein. Par exemple, `ls -l /bin|more` affiche page par page la liste des fichiers contenus dans le répertoire `/bin` avec leurs droits.

Pour rediriger le résultat d'une commande vers un fichier *`commande > fichier`*. Pour rediriger les messages d'erreurs d'une commande vers un fichier *`commande 2> fichier`**. Si on veut jeter quelque chose, on dispose du fichier `/dev/null` qui absorbe toute information sans la stocker.

4.3 Commandes souvent utilisées comme filtres

- `head` *`fichier`* affiche les 10 premières lignes du fichier. `head -`*`nb fichier`* affiche les *`nb`* premières lignes du fichier.
- `tail` *`fichier`* affiche les 10 dernières lignes du fichier. `tail -`*`nb fichier`* affiche les *`nb`* dernières lignes du fichier. `tail -f` *`fichier`* affiche les 10 dernières lignes du fichier à tout instant. Cette commande est utile pour surveiller le contenu d'un fichier qui se remplit (par exemple si on a redirigé le résultat d'une commande longue à exécuter vers ce fichier). On l'interrompt en appuyant simultanément sur la touche `Ctrl` et la touche `c`.
- `wc` *`fichier`* (pour *word count*) affiche le nombre de lignes, de mots (suites de caractères séparées par des blancs), de caractères du fichier. On peut obtenir une seule information avec l'option `-l` pour les lignes, `-w` pour les mots et `-c` pour les caractères.
- `file` *`fichier`* indique quelle est la nature du fichier. On obtient un certain nombre d'informations précieuses comme le montrent les exemples suivants :

```
$ cat f1
Ceci est du texte.
$ file f1
f1: ASCII text
$ cat f3
#! /bin/sh

echo $SHELL
$ ls -F f3
f3*
$ file f3
f3: Bourne shell script text
$ ls -lF f6
lrwxrwxrwx   1 user  group         2 Oct  3 20:24 f6 -> f1
$ file f6
f6: symbolic link to f1
$ file /bin/arch
/bin/arch: ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked, stripped
$ file /dev/null
/dev/null: character special (1/3)
$ file /dev/fd0
/dev/fd0: block special (2/0)
$ file /dev/ttyp0
/dev/ttyp0: character special (3/0)
$ tar -cf f1.tar f1
$ file f1.tar
f1.tar: GNU tar archive
$ gzip f1.tar
$ file f1.tar.gz
f1.tar.gz: gzip compressed data, deflated, original filename, last
modified: Sun Oct  4 13:16:00 1998, os: Unix
```

- `sort` *`fichier`* trie les lignes d'un fichier par ordre alphabétique. Quelques options :
  - `-r` pour inverser l'ordre d'affichage

- `-k` *`pos`* pour ordonner par rapport au mot numéro *`pos`* de la ligne
- `-n` pour ordonner selon l'ordre numérique : il faut se souvenir que par ordre alphabètique 10 se trouve avant 2, ce qui nest pas toujours ce que l'on souhaite
- `-u` élimine les lignes identiques :

Voici quelques exemples :

```
$ cat etdts
Victor Hugo
Andre Gide
Gustave Eiffel
$ sort etdts
Andre Gide
Gustave Eiffel
Victor Hugo
$ sort -k2 etdts
Gustave Eiffel
Andre Gide
Victor Hugo
$ cat tutu
1
3
10
5
3
$ sort tutu
1
10
3
3
5
$ sort -u tutu
1
10
3
5
$ sort -n tutu
1
3
3
5
10
$ sort -n -u tutu
1
3
5
10
```

5 Autres commandes utiles

- `date` : donne la date et l'heure du système.
- `clear` : efface la fenêtre dans laquelle vous tapez cette commande. On peut aussi taper simultanément sur le modificateur « `Ctrl` » et sur `l` pour obtenir le même résultat sans que ceci soit considéré comme une commande (affiché à l'écran, placé dans l'historique des commandes, etc.)

5.1 Compression et commandes associées

Quand un problème de place surgit, on compresse les fichiers avec la commande `gzip` *`fichier`* qui comprime le (ou les) fichiers passés en argument. Leur nom se voit augmenter du suffixe `.gz`.

Vous pouvez aussi recevoir ou récupérer sur le réseau de tels fichiers, pour les décompresser, on utilise la commande `gunzip` *`fichier`*`.gz` qui décomprime le fichier comprimé par `gzip`.

On veut souvent compresser tout un répertoire en un seul fichier, il faut alors que le répertoire soit vu comme une entité en soi. Pour cela, on utilise la commande `tar` qui servait à l'origine à stocker et déstocker des informations sur bande. Pour ne faire qu'un d'un ensemble de fichiers `tar -cf` *`résultat liste des fichiers`*. Pour retrouver l'ensemble de fichiers à partir du fichier *`résultat`*, `tar -xf` *`résultat`*. Pour voir ce qui se passe, ajouter `v` (comme *verbose*) aux options ; pour voir ce qui va apparaître sans tout déplier remplacer `x` par `t`.

Vous pouvez aussi utiliser une commande très simple lorsqu'elle est disponible sur votre système, la commande `shar`. Pour ne faire qu'un seul fichier `shar` *`liste des fichiers`* `>` *`résultat`* et pour déplier l'ensemble des fichiers `sh résultat`.

5.2 Comment retrouver un fichier ?

On dispose de deux outils, `grep` et `find`, pour retrouver un fichier ou des lignes d'un fichier.

`grep` *`str fichier`* affiche toutes les lignes du fichier (ou des fichiers) qui contiennent la chaîne de caractères *`str`*. Cette commande dispose de plusieurs options :
- `-i` pour afficher le résultat que chaque caractère soit en majuscule ou en minuscule
- `-c` pour n'afficher que le nombre de lignes trouvées
- `-l` pour n'afficher que le nom des fichiers où la chaîne de caractères apparaît
- `-n` pour afficher en plus le numéro des lignes trouvées.

En fait, on peut rechercher quelque chose de beaucoup plus compliqué qu'une chaîne de caractères avec `grep` (des *expressions régulières*) mais la maîtrise de cette notion complexe dépasse le cadre de ce document.

`find` examine récursivement à partir du point de départ qu'on lui fournit si les fichiers satisfont les critères décrits. Cette commande dispose de nombreuses options. En voici les principales :
- `-name` *`critère`* recherche le fichier de nom *`critère`*. Ce nom peut contenir des caractères de substitutions ;
- `-print` pour imprimer le résultat de la recherche ;
- `-exec` *`commande`* parcourt récursivement l'arborescence et applique la commande à chaque fichier. Il s'agit d'un moyen puissant d'appliquer une commande à tous les fichiers d'une arborescence. `{}` représente alors l'argument de la commande. On termine une telle expression par `\;`.

Voici quelques exemples d'utilisations de cette commande :

```
$ find . -name biblio.bib -print
./TeX/Document/biblio.bib
./Biblio/biblio.bib
$ find . -exec grep -l bonjour {} \; -print
./TeX/Livre/Ctex/premieres_expressions.ctex
./TeX/Livre/Mltex/premieres_expressions.mltex
./TeX/Livre/TeX/livre.dvi
./TeX/Livre/Ftex/premieres_expressions.ftex
$ find . -name *.tex -print
./TeX/Livre/TeX/livre.tex
./TeX/Livre/TeX/titre.tex
./Biblio/test.tex
./Enseignement/98-99/Programmation/Viatique_UNIX/unix.tex
```

Ici « `.` » désigne le répertoire courant.

Attention, le résultat peut être très long à imprimer sur votre écran. N'oubliez pas que le disque que vous utilisez est partagé par toute la licence pour le moins, que lancer la commande `find` est coûteux en temps et en vitesse pour vous sur votre machine, mais aussi pour tous les autres étudiants qui utilisent ce disque ! Il faut toujours donner un point de départ le plus précis possible pour les recherches pour limiter la portion de l'arborescence à parcourir. Par exemple, il ne faut JAMAIS lancer « `find /` » !

Vu le coût de cette commande, il est dommage d'avoir fait une grosse recherche et d'avoir oublié de l'imprimer,donc pensez à ajouter `-print` à la fin de votre commande.

5.3 La commande `locate`

Certains UNIX sont dotés de la commande `locate`, qui utilise une base de données actualisée chaque nuit pour stocker un certain nombre d'informations sur le contenu de l'arborescence. On peut alors effectuer des recherches très rapides et peu coûteuses grâce à cette base de données.

5.4 Impression

On dispose d'un certain nombre de commandes pour gérer l'impression. Il faut d'abord que le système sache sur quelle imprimante vous voulez imprimer. Pour cela vous disposez de la variable d'environnement `PRINTER`. Vous pouvez voir sa valeur à l'aide de la commande `echo $PRINTER` et changer sa valeur parla commande `export PRINTER=`*`valeur`**. Cette valeur sera accessible à toutes les commandes de gestion de l'impression qui utiliseront l'imprimante adéquate. Les commandes sont :

- `lpr` *`fichier`* : C'est la commande qui envoie un fichier à l'imprimante. Ses arguments sont les noms des fichiers à imprimer.
  On peut cependant l'utiliser sans arguments, par exemple après un « pipe », dans ce cas la commande imprime ce qu'elle reçoit sur son entrée standard.
- `lpq` : Cette commande vous donnera la liste des fichiers en attente d'impression, avec leur numéro d'ordre dans la file d'attente et leur propriétaire.
- `lprm` *`num`* : Cette commande demande d'enlever de la file d'attente le fichier dont le numéro est *`num`*. Bien entendu vous ne pouvez enlever que vos propres fichiers.

5.5 Les caractères de substitutions et le shell

Les commandes tapées au claviers sont interprétées par un programme appelé *shell* (on parle aussi d'*interprète de commandes* en français). Ce programme permet d'employer des raccourcis claviers quand on veut effectuer la même action sur plusieurs fichiers. Ces caractères sont `*` qui remplace un nombre quelconque de caractères et `?` qui remplace un et un seul caractère. Leur mécanisme est le suivant : quand le shell les rencontre sur la ligne de commande, il remplace le mot dans lequel ils sont par la liste de tous les fichiers dont le nom peut être représenté par ce mot. Ainsi, si on se trouve dans un répertoire contenant les fichiers suivant :

```
emacs-dep.fig   hierarchie.fig  pres-mat.dvi    pres-mat.tex~
emacs-dep.tex   hierarchie.tex  pres-mat.log
essai.ps        pres-mat.aux    pres-mat.tex
```

l'ordre `ls e*` donnera comme résultat :

```
emacs-dep.fig   emacs-dep.tex   essai.ps
```

l'ordre `ls *.tex` donnera pour résultat

```
emacs-dep.tex   hierarchie.tex  pres-mat.tex
```

l'ordre `ls ??e*` donne

```
hierarchie.fig  pres-mat.aux    pres-mat.log    pres-mat.tex~
hierarchie.tex  pres-mat.dvi    pres-mat.tex
```

Ces caractères de substitutions préfigurent la notion d'expressions régulières que l'on retrouve dans un certain nombre d'outils d'édition.

Le shell dont vous disposez s'appelle `bash`, comme vous le montre la commande `echo $SHELL`. Les commandes suivies d'uneastérisque*dans ce document dépendent du shell utilisé. Il existe deux familles de shells :
- la famille *Bourne shell*, avec `sh`, `ksh`, `psh`, `bash`, `zsh`, etc.
- la famille *C shell*, avec `csh`, `tcsh`, etc.

Ces deux familles de shells sont de syntaxes différentes, aussi est-il recommandé quand on a pris l'habitude d'un shell d'une famille de ne pas changer pour un shell de l'autre famille.

Vous pouvez adapter votre environnement *shell* dans une certain mesure. Pour savoir quelles sont les valeurs des variables *shell*, `printenv` pour les voir toutes, `echo $`*`var`* pour voir la valeur de la variable *`var`*. Pour lesmodifier, comme on l'a vu pour l'impression *`var=valeur`**, directement sous le *shell* où l'on veut que la modification soit prise en compte, dans son `~/.profile`*si on veut retrouver cette nouvelle valeur à la prochaine connection. Pour tenir compte

tout de suite de plusieurs modifications apportées à votre `~/.profile`*, rechargez le par la commande `. .profile`*.

Ce *shell* dispose d'un historique des commandes que vous avez tapées précédemment. Vous pouvez les retrouver en appuyant simultanément sur les touches `Ctrl` et `p`, ou revenir en avant dans cette liste en appuyant simultanément sur les touches `Ctrl` et `n`. De même si vous commencez à écrire un nom de commande ou de fichier et que vous appuyez ensuite sur la touche `Tab`, alors le *shell* complètera ce nom dans la mesure du possible ou vous proposera les différentes réponses qu'il a trouvé.

6 Les processus

Nous avons dit au début de ce document qu'UNIX est multi-tâches.

6.1 Visualisation et destruction de processus

Pour visualiser les commandes que l'on a lancées, on dispose de la commande `ps`. Cette commande dispose de nombreuses options, parmi lesquelles :
- `-l` pour plus d'informations
- `-w` ou même `-ww` pour éviter que les lignes d'affichage ne soient tronquées à la fin de la ligne d'écran
- `-x` pour voir les commandes système
- `-u` pour disposer de plus d'informations dans un format plus lisible
- `-a` pour ne pas se cantonner à vos processus, mais aussi ceux des autres utilisateurs.

Parmi ces informations se trouve un numéro sous les lettres PID qui sert à désigner ce processus lors d'ordres tels que ceux qui suivent.

On peut envoyer des signaux à un processus par la commande `kill`. Son nom vient du fait qu'initialement il servait essentiellement à détruire les processus. Nous nous limiterons ici à cette fonction de la commande `kill`. `kill` *`pid`* ou plus fort `kill -9` *`pid`* détruisent complètement le processus.

6.2 Lancer un programme en tâche de fond

Si un programme n'a pas besoin d'écrire sur le terminal ni d'en recevoir des ordres, il peut être lancé en tâche de fond, c'est-à-dire qu'il s'exécute en rendant la main, ce qui permet de continuer à travailler dans la même fenêtre ; l'exemple le plus courant est un programme qui ouvre ses propres fenêtres et se sert uniquement d'elles pour communiquer avec l'utilisateur. On lance de tels programmes en faisant suivre la ligne de commande qui lance le programme d'un caractère `&` (par exemple, `emacs &` lance emacs en tâche de fond).

On peut aussi suspendre l'exécution d'un programme en tapant simultanément `Ctrl` et `z`. On peut alors soit la relancer normalement (en *foreground*), par la commande `fg`, soit la lancer en tâche de fond (en *background*, par la commande `bg`.

On peut visualiser tous ses programmes qui tournent en tâche de fond, par la commande `jobs`. Lorsqu'il y en a plusieurs, ils sont présentés suivis d'un numéro et les commandes que l'on peut envoyer à ces tâches doivent être suivi du caractère `%` et de ce numéro (différent du PID du processus).

Nous avons dit au début de ce document qu'UNIX est multi-utilisateurs. : plusieurs utilisateurs peuvent être connectés sur la même machine, même si un seul au plus utilise l'écran qui lui est associé (voir la section **??**).

On peut voir qui est connecté par `w`, `who`, `finger` (cette commande permet d'obtenir plus d'informations sur un utilisateur précis en indiquant son nom de login).

Comme dans toute utilisation de ressources limitées, un peu de bon sens, de tolérance et de convivialité aident à une bonne entente dans la communauté des utilisateurs.

7 L'aide

7.1 Le manuel en ligne `man`

`man` est un programme qui vous donnera une aide sur la plupart des programmes installés (programmes UNIX).

On l'appelle en indiquant le nom de la commande pour laquelle on souhaite avoir des informations (par exemple `man rmdir`) ou en indiquant un mot anglais qui correspond à ce que l'on cherche à faire et pas nécessairement un nom de commande (par exemple « `man -k caml` »). Voici le résultat de l'exécution de ces deux exemples d'appel de `man` :

```
$ man rmdir
RMDIR(1)                                                RMDIR(1)


NAME
       rmdir - remove empty directories

SYNOPSIS
       rmdir [-p] [--parents] [--help] [--version] dir...

DESCRIPTION
       This  manual  page  documents  the  GNU  version of rmdir.
       rmdir removes each given empty  directory.   If  any  non-
       option argument does not refer to an existing empty direc-
       tory, it is an error.

   OPTIONS
       -p, --parents
              Remove any parent directories that  are  explicitly
              mentioned  in  an  argument,  if  they become empty
              after the argument file is removed.

       --help Print a usage message on standard output  and  exit
              successfully.

       --version
              Print  version  information on standard output then
              exit successfully.
```

```
FSF                        GNU File Utilities                     1

$ man -k compiler
camlc (1)             - The Caml Light compiler
coqc (1)              - The Coq compiler
bootpef (8)           - BOOTP Extension File compiler
cccp, cpp (1)         - The GNU C-Compatible Compiler Preprocessor.
compile_et (1)        - error table compiler
fort77 (1)            - invoke f2c Fortran translator transparently, like a compi  ler
g++ (1)               - GNU project C++ Compiler
gcc, g++ (1)          - GNU project C and C++ Compiler (egcs - 1.0.3)
gcc, g++ (1)          - GNU project C and C++ Compiler (v2.7)
guavac (1)            - a free Java compiler
ocamlc (1)            - The Objective Caml bytecode compiler
ocamlcp (1)           - The Objective Caml profiling compiler
ocamlopt (1)          - The Objective Caml native-code compiler
tic (1m)              - the terminfo entry-description compiler
diagnostics (3)       - Perl compiler pragma to force verbose warning diagnostics
less (3)              - perl pragma to request less of something from the compiler
```

7.2 Les manuels en ligne

Le manuel en ligne de Caml Light est disponible en local sur votre machine, ainsi que ceux d'un certain nombre d'applications que vous êtes susceptibles d'utiliser. Pour cela, lancer `netscape` lors de votre premier TD et familiarisez-vous avec ce programme qui est facile d'utilisation et qui pourra vous servir à l'occasion d'outil pour lire ou envoyer du courrier électronique.

Bien des commandes affichent une certaine quantité d'informations si on les appelle sans argument, ou avec l'option `-h` ou `-?` ou encore `-help`.

8 Réseau, Internet, convivialité

8.1 Réseau

UNIX utilise le réseau, entre autres choses pour partager les disques de manière transparentes entre plusieurs machines.

Pour se connecter sur une autre machine, on dispose des commandes `rlogin`, `telnet`.

8.2 Le courrier électronique ou messagerie

Un moyen simple de correspondre est le courrier électronique. Vous y avez accès par la commande `mail` (et aussi par les éditeurs et un certain nombre de commandes, une bonne occasion de lancer « `man -k mail` » !).

Pour envoyer du courrier électronique à un utilisateur de « nom de login » ***utilisateur*** qui dispose d'un compte sur une machine ***machine.domaine.pays*** , on tape

mail *utilisateur@machine.domaine.pays*

Le programme vous demande alors le sujet à indiquer pour le message puis le texte du message qui doit se terminer par un point en tête d'une ligne qui ne contient rien d'autre.

Pour voir si l'on a reçu du courrier électronique et pour le lire, on utilise la commande `mail` sans argument. On dispose d'une aide en tapant `help`.

Votre adresse électronique est la suivante : Demandez à votre chargé de TD/TP quelle est l'adresse électronique où vous pouvez lui écrire.

9 Sauvegarde sur disquette

Sous certains UNIX dont Linux on dispose d'un outil pour manipuler des disquettes formatées au format DOS usuel : il s'agit des `mtools`. On peut consulter le manuel en ligne globalement par la commande `man mtools` ou bien commande par commande (`man mcopy`, etc.). Le nom de ces commandes est généralement composé du nom usuel sous DOS préfixé de la lettre *m*. Les principales commandes sont les suivantes :

- `mcopy`. Les deux utilisations les plus fréquentes :
  - Pour sauvegarder sur disquette le fichier `filehd`, `mcopy` *`filehd`* `a:`
  - Pour recopier sur le disque le fichier `filefd` qui se trouve sur la disquette

    `mcopy a:`*`filefd nom_de_fichier_ou_répertoire`*

    ou plus simplement

    `mcopy a:`*`filefd`* `.`

    si on se trouve à l'endroit où l'on veut placer ce fichier.
- `mmove` et `mren` permettent de déplacer ou renommer des fichiers et répertoires.
- `mtype` affiche le contenu d'un fichier sur la disquette.
- `mdel` permet de supprimer un fichier sur la disquette.
- `mdir` permet de voir le contenu d'un répertoire sur la disquette.
- `mcd` permet de changer le répertoire courant sur la disquette.
- `mmd` et `mrd` permettent respectivement de créer et de détruire un répertoire sur la disquette.

10 Emacs

Emacs est un éditeur, c'est-à-dire un programme permettant de taper des textes. Une fois lancé, il ouvre une fenêtre dans laquelle vous tapez vos fichiers. Vous disposez de menus en haut de l'écran pour les manipulations de base. Vous disposez aussi en annexe de feuilles décrivant les commandes : toutes ces commandes ne vous seront pas utiles. Lorsqu'une commande est donnée sous la forme « C-k » cela signifie « Control-k » et veut dire que vous devez appuyer sur la touche marquée « Ctrl » puis en la gardant enfoncée sur la touche « k ». Pour les commandes marquées « M-k » (lire Méta-k) il faut appuyer sur la touche « Compose Character » ou « Esc » au lieu de la touche « Ctrl ».

10.1 Vocabulaire

Emacs est un éditeur multi-fichiers, c'est-à-dire qu'on peut éditer plusieurs fichiers simultanément. Chaque fichier se trouve dans un *Buffer*, on passe de l'édition d'un buffer à un autre à l'aide du menu `Buffer`. On peut aussi manipuler plusieurs fenêtres (pour voir simultanément plusieurs fichiers par exemple). Les fenêtres sont appelées *frames* et on les manipule par l'intermédiaire du menu `File`. On peut aussi couper chaque fenêtre en plusieurs sous-fenêtres appelées *windows*. Si jamais, suite à une fausse manœuvre vous vous trouvez avec trop de sous-fenêtres dans votre fenêtre, cliquez dans celle que vous voulez garder et tapez la commande `C-x 1`, les autres sous-fenêtres et les fichiers qui y étaient édités sont toujours accessibles par `C-x o` (les passer en revue successivement) ou `C-x b` (un buffer est proposé par défaut et on peut taper le nom de celui qui nous intéresse si ce n'est pas celui qui est proposé) ou `C-x C-b` (en voir apparaître la liste dans un buffer et choisir celle que l'on veut voir).

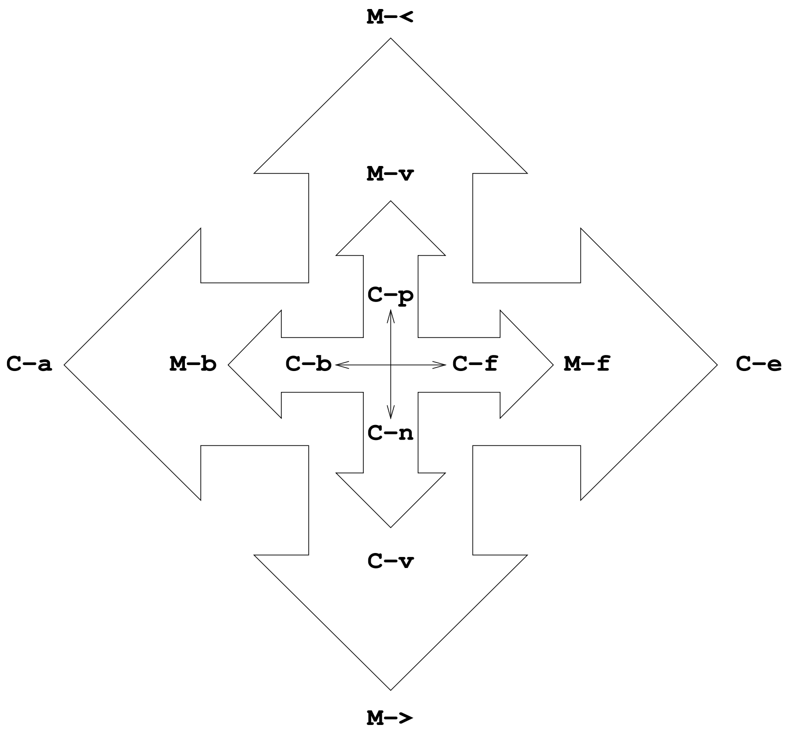
Si vous utilisez la possibilité d'appliquer Caml Light à une phrase que vous venez de taper, vous vous apercevrez que la fenêtre est alors coupée en deux parties : l'une contient le buffer que vous éditez et l'autre le toplevel Caml Light avec la phrase que vous lui avez envoyé et la réponse de Caml Light.

10.2 Un outil de base : le *copier-coller*

Pour sélectionner une partie de texte, placez la souris d'un côté de ce texte et cliquez avec le bouton gauche de la souris. Ensuite placez la souris de l'autre côté et cliquez avec le bouton droit de la souris (`emacs`) ou bien sans relâcher le bouton de la souris déplacer la souris jusqu'à ce que la zone coloriée coïncide avec la sélection souhaitée (`emacs`/`xemacs`). Cette portion de texte apparaît alors coloriée sur votre écran. On peut ensuite couper (le faire disparaître à cet endroit du texte et le garder en mémoire) ou coller (faire apparaître ce texte à un autre endroit) le texte sélectionné. On coupe la portion de texte sélectionnée soit à l'aide du menu `Edit` soit en cliquant 2 fois avec le bouton droit de la souris ou bien encore par `C-w`. On peut coller une sélection en cliquant au point d'insertion avec le bouton du milieu de la souris ou bien encore par `C-y`.

Voyez l'annexe pour les autres commandes de l'éditeur.

10.3 Déplacements



Dans ce diagramme, les flèches les plus internes représentent le déplacement d'un caractère, les moyennes représentent un déplacement d'un mot ou d'un écran, et les plus grosses le déplacement jusqu'au bout de la ligne ou du fichier. Bien entendu, vous pouvez utiliser les touches fléchées du clavier pour vous déplacer d'un caractère ou d'une ligne.

10.4 Quelques manipulations de fichiers

Vous disposez à tout instant d'un *buffer* nommé `*scratch*` où vous pouvez écrire sans qu'il soit sauvegardé. Vous pouvez y taper un texte. Il vous faudra ensuite le sauvegarder en le nommant par `C-x C-w`. Vous pourrez ensuite vous contenter de le sauver par `C-x C-s`. Si vous voulez visualiser un fichier, `C-x C-f` vous demandera le fichier que vous souhaitez lire.

10.5 L'aide en ligne sous emacs

Emacs dispose d'aide sous de nombreuses formes comme vous le montre le menu `Help` tout à droite de la barre de menu. On peut accèder à de la documentation sur emacs lui-même (« Describe Mode », « Commande Apropos », « List Keybindings », « Emacs Tutorial », etc), mais aussi sur des outils Unix ou des commandes complexes (« Man », « Browse Manuals »). On accède ainsi à de gros volumes de documentation structurée.

10.6 Philosophie de l'éditeur de texte

`emacs` a été conçu pour pouvoir être utilisé « en environnement hostile », sur de simples terminaux textuels sans confort. On peut donc tout faire sans souris, sans touches de flèches, sans touches de fonctions, sans pavé numérique et sans menus. Cela peut paraître ingrat au début, mais on s'aperçoit au bout d'un certain temps que c'est un moyen beaucoup plus rapide de manipuler du texte et des fichiers à condition d'en avoir fait l'investissement.

Bibliographie

Un grand nombre d'ouvrages sur le sujet sont disponibles dans les différentes bibliothèques du campus. Citons ici à titre d'exemples :

[1] J.-P. Armspach. *UNIX, Initiation et utilisation.* InterEditions, 1994. *(moins de 180F).*

[2] J.-M. Rifflet. *La programmation sous UNIX*, 1re éd. McGraw-Hill, 1986. *(plus disponible dans le commerce, mais 1 exemplaire à la bibliothèque INFO RECHERCHE et 4 exemplaires à la bibliothèque MATHS-INFO 2ecycle).*

[3] J.-M. Rifflet. *La programmation sous UNIX*, 2e éd. McGraw-Hill, 1986. *(plus disponible dans le commerce, mais 19 exemplaires à la bibliothèque MATHS-INFO 2ecycle).*

[4] J.-M. Rifflet. *La programmation sous UNIX*, 3e éd. Édiscience International, 1993. *(plus de 250F, mais 17 exemplaires à la bibliothèque 1ercycle, 1 exemplaire à la bibliothèque INFO RECHERCHE, 24 exemplaires à la bibliothèque MATHS-INFO 2ecycle).*

[5] D. Taylor et J. C. Armstrong, Jr. *Unix.* Le tout en poche, 1998. *(assez bien fait, un livre de poche à moins de 60F disponible à la librairie du campus).*

...et bien sur l'aide en ligne sur les stations avec la commande `man` et certaines informations sous `Emacs` (menu `Help`) et sous `netscape` sur la page d'accueil.