# Module SIT151 : Articulation machine-langage-OS

UV FIP SIT150 : Outils informatiques pour l'ingénieur

Recueils de Cours-TP Introduction à Unix/Linux

Christophe Lohr

Automne 2023

## Sommaire



**IMT Atlantique**
Bretagne-Pays de la Loire
École Mines-Télécom

# FIP SIT151 Introduction à Linux

Christophe Lohr

Automne 2023

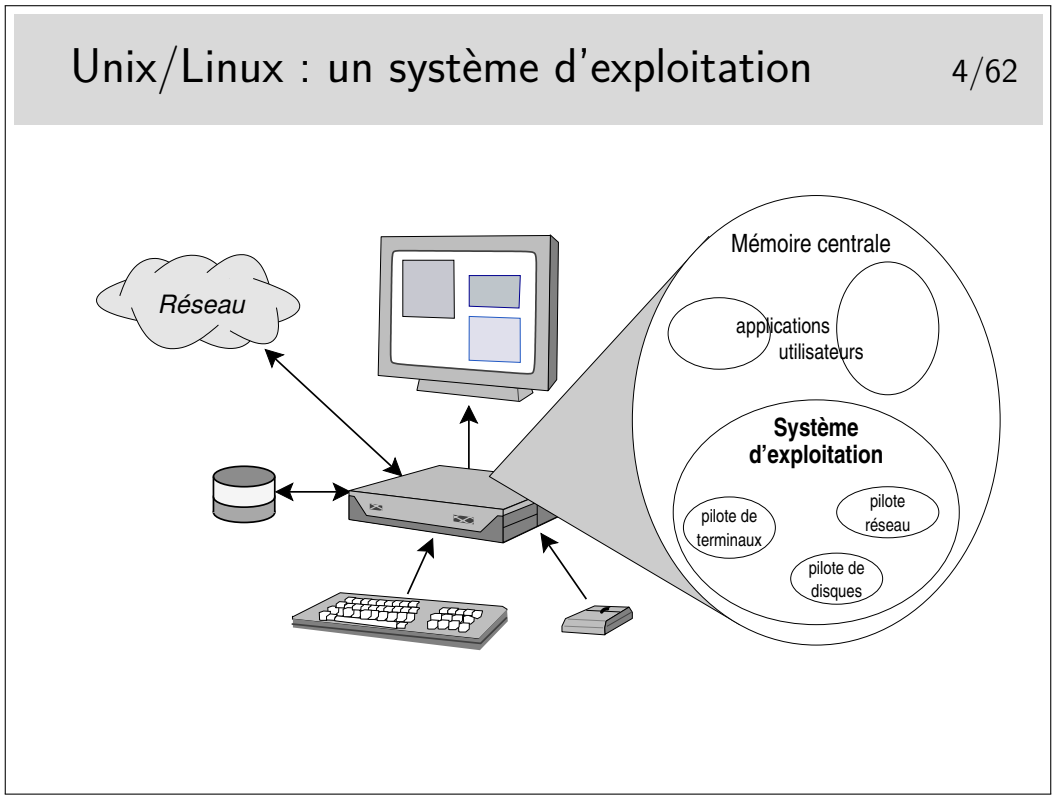
Grace Hopper - 1960

Dennis Ritchie & Ken Thompson - 1972

## Sommaire

# 1 Introduction

## 1.1 Le système d'exploitation




Le système d'exploitation (SE, OS en anglais) est une application chargée en mémoire centrale peu après la mise sous tension de la machine (juste après que que programme résidant en mémoire EEPROM se soit exécuté, ce qu'on appelle le BIOS dans l'architecture PC).

Le SE gère les applications qui s'y déroulent, et sert d'interface entre elles et les périphériques matériels. La mémoire centrale est chargée avec le (1) Système d'exploitation et avec (2) les programmes applicatifs lancés par les utilisateurs. Le système d'exploitation contient des modules spécifiques aux périphériques qu'il sait contrôler, ce sont les pilotes (*drivers* en anglais)

## 1.2 Historique

### Historique 6/62

- Unix naît officiellement le 1er janvier 1970 dans les laboratoires Bell AT&T : Ken Thompson et Dennis Ritchie
- Années 1970 : développement d'Unix, langage C (1973)
- Années 1980 : deux filières
  - Univ. Berkeley : système Unix BSD (Berkeley Software Development)
  - AT&T : Unix Système V, version commerciale standard
  - 1984 : Richard Stallman crée la Free Software Foundation et la Licence Publique Générale (GNU GPL)
- Années 1990 :
  - 1994 : Linus Torvalds écrit le noyau Linux
  - Les versions BSD continuent en logiciel libre : FreeBSD, OpenBSD...

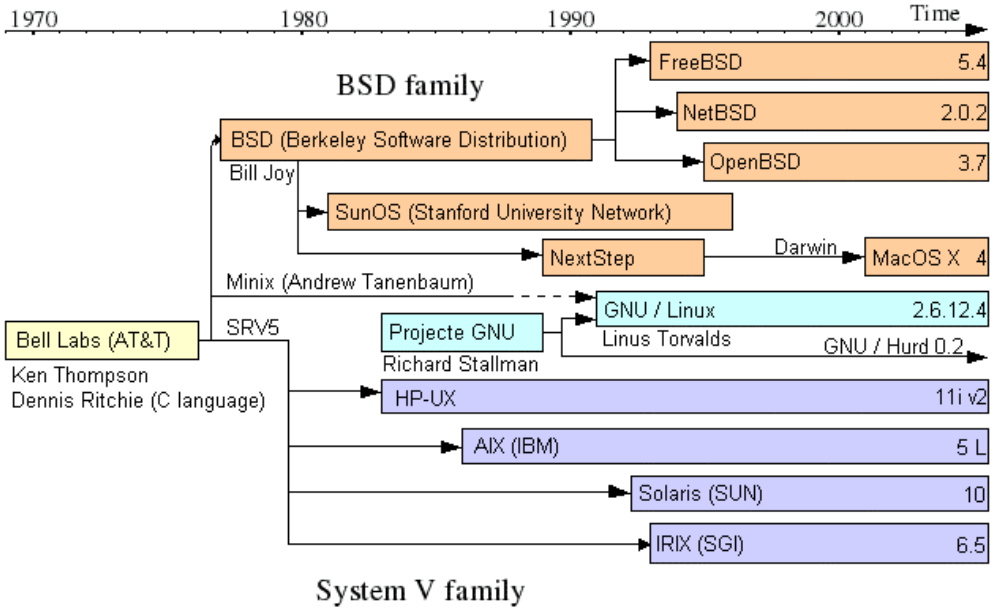
Pointeurs :

— `http://en.wikipedia.org/wiki/Unix`

— `http://virtual.park.uga.edu/hc/unixhistory.html`

— `http://www.princeton.edu/~hos/mike/transcripts/thompson.htm` (interview de Ken Thompson)

— `http://www.unix.org/what_is_unix/history_timeline.html`



## 1.3 De Unix à Linux

### Linux : un système Unix 8/62

- Philosophie d'Unix :
  - (presque) tout s'utilise comme un fichier
  - "Do one thing, do it well" (Doug McIlroy, l'inventeur des *pipes Unix*) :
    - Write programs that do one thing and do it well.
    - Write programs to work together.
    - Write programs that handle text streams, because that is a universal interface.
- Caractéristiques d'un système d'exploitation Unix :
  - Multitâche (multi processus)
  - Multi utilisateurs
- Spécificités (de Linux et de tous les Unix) :
  - Son Système de Gestion de Fichiers
  - La gestion des processus
- → Linux c'est (une implémentation) Unix

### ...vocabulaire... 9/62

**Linux** (p.ex. Linux 5.7.0)

Le noyau, uniquement !

**GNU/Linux**

+ commandes Unix de base (implémentation de GNU)
copier un fichier, répertoire, permissions utilisateur..

**une distribution Linux** (p.ex. Ubuntu 22.04, Debian Buster)

+ organisation des fichiers, outils d'administration, applications

Sur clef USB :

— LinuxLive USB Creator `http://www.linuxliveusb.com/` (sous Windows)

— PenDriveLinux YUMI `http://www.pendrivelinux.com/yumi-multiboot-usb-creator/` (sous Windows)

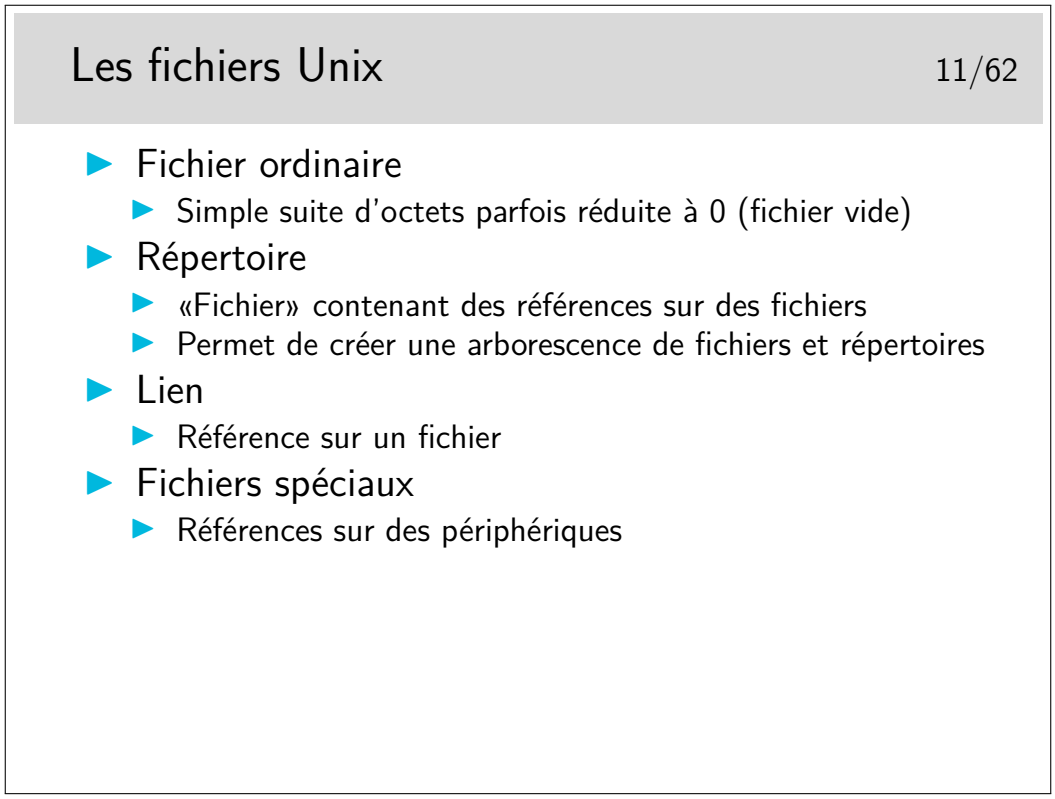
— Xboot `https://sites.google.com/site/shamurxboot/` (sous Windows)

— MultiBoot `LiveUSBhttp://liveusb.info/dotclear/` (sous Linux)

— etc.

# 2 Le système de fichiers

## 2.1 Structure, nommage, droits

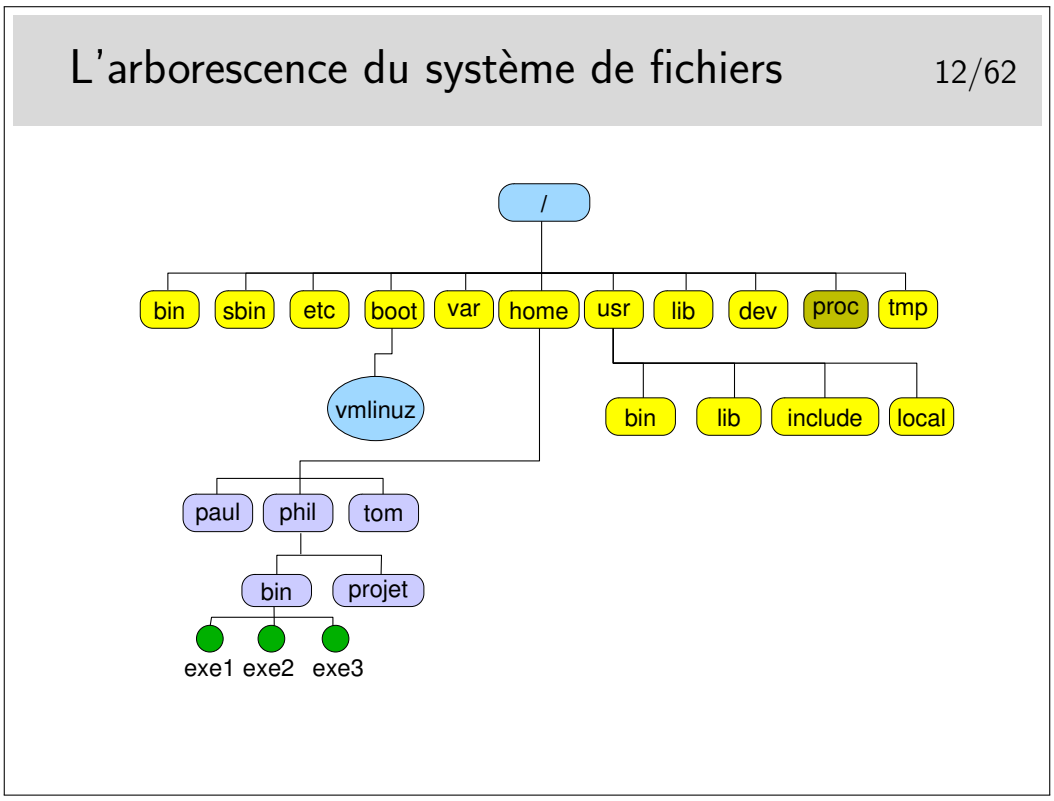



Il existe quelques autres types de fichiers pouvant apparaître dans l'arborescence :

— les sockets (type `s`) permettant la communication entre processus

— les tubes nommés (type `p` comme *pipe*) permettant aussi la communication entre processus

Les liens peuvent être de deux types : liens durs (version originale des liens sur Unix) et les liens symboliques (type l), version apportée par l'Unix de Berkeley (BSD) pour augmenter la portée des liens natifs (durs). Les liens durs ne peuvent être réalisés qu'entre références (nom dans un répertoire, on dit aussi entrée dans un répertoire) sur un même système de gestion de fichiers. Les liens symboliques peuvent passer les frontières physiques des système de fichiers.




— `/bin` (binaires) et `/usr/bin` les commandes standards (usr : UNIX System Resources)

— `/etc` fichiers d'administration (editable text configuration)

— `/sbin` commandes d'administration (binaires système)

— `/lib` et `/usr/lib` les bibliothèques (libraries ; fichiers contenant les fonctions appelées par les commandes pour réaliser les opérations avec le système, exemple libc.a est la bibliothèque standard du langage C).

— `/dev` les fichiers spéciaux, référencent les périphériques du système (les *devices*).

— `/tmp` répertoire où sont créés les fichiers temporaires, il peut exister aussi /usr/tmp et /var/tmp.

— `/var/spool` répertoire où sont créées les files d'attente pour différents services tels que l'impression, le courrier électronique, etc. (var : variable)

— `/var/spool/cron` contient les travaux du service *cron* (voir cette référence dans le manuel de référence, faire man cron).

— `/usr/include` contient les fichiers d'entête standards des programmes en langage C.

— `/boot` répertoire contenant les composants du noyau du système (bootstrap : amorçage du système).

— `/boot/vmlinuz` le noyau (le système-lui même en quelque sorte).

— etc.

Une organisation standardisée sous Unix : Filesystem Hierarchy Standard (`http://www.pathname.com/fhs/`).

## Les fichiers - Le nommage 13/62

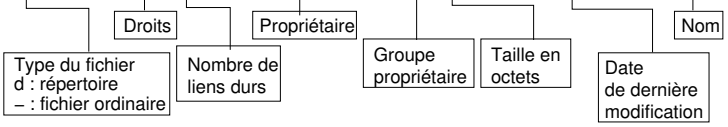
- Nommage absolu
  - par rapport à la racine, le nom commence par /
  - /home/phil/bin/exe1
- Nommage relatif
  - relatif au répertoire dans lequel on est :

    home/phil/bin/exe1 si on est dans /
    phil/bin/exe1 si on est dans /home
    bin/exe1 si on est dans /home/phil
    exe1 si on est dans /home/phil/bin

## Visualiser le contenu d'un répertoire 14/62

- La commande ls
  - Exemple :

```
[bash]$ ls -l
total
-rwxr-xr-x    1   clohr   ens-rec     7790   avr 11 2020 essai
-rw-rw-r--    1   clohr   ens-rec     1122   avr 24 2020 essai.c
-rw-rw-r--    1   clohr   ens-rec  9869052   avr 25 2020 test
```



## Les fichiers cachés 15/62

- Ce sont les fichiers dont le nom commence par un point
  - Exemple : .bashrc
- Par défaut les outils d'affichage du contenu des répertoires n'affichent pas les fichiers cachés
- Ce sont généralement des fichiers de configuration d'applications
- On peut les comparer à la base de registres sur des systèmes concurrents à Unix/Linux
- Il existe aussi des répertoires cachés, leur nom commence aussi par un point

Pour visualiser la liste des fichiers cachés il faut utiliser l'option `-a` de `ls`. Pour les outils graphiques de gestion de fichiers (les «explorateurs» dirions nous sous un autre système d'exploitation bien connu), il existe une option de paramétrage dans les menus de configuration (parfois appelés «*préférences*»).

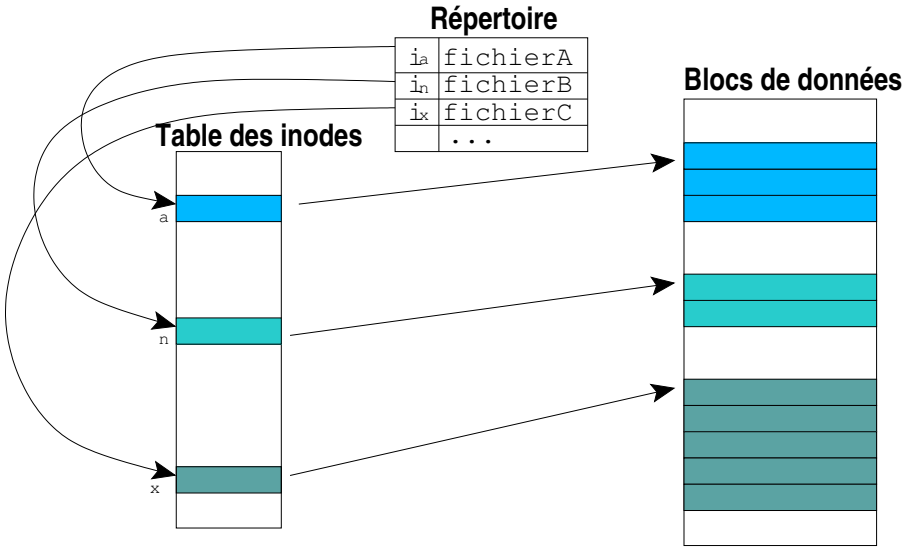
## Structure et contenu d'un répertoire 16/62

- Un répertoire est avant tout un fichier
- Ce fichier, de type particulier, contient des références à des fichiers ou sous-répertoires
- Une référence est composée de :
  - la longueur de la référence
  - la longueur du nom du fichier ($< 256$)
  - le nom du fichier
  - un nombre index appelé **numéro d'inode** qui identifie une structure dans une table appelée ***table des inodes***
  - Cette structure contient toutes les informations sur le fichier

## Système de fichiers - Contenu d'un inode 17/62

- mode et type du fichier (voir droits et protections)
- nombre de liens sur le fichier
- numéro d'identification du propriétaire du fichier (uid)
- numéro de groupe du propriétaire du fichier (gid)
- taille du fichier en octets
- adresse des blocs constituant le fichier
- date du dernier accès à ce fichier
- date de la dernière modification de mode
- date de création

## Répertoire / inodes / bloc de données 18/62



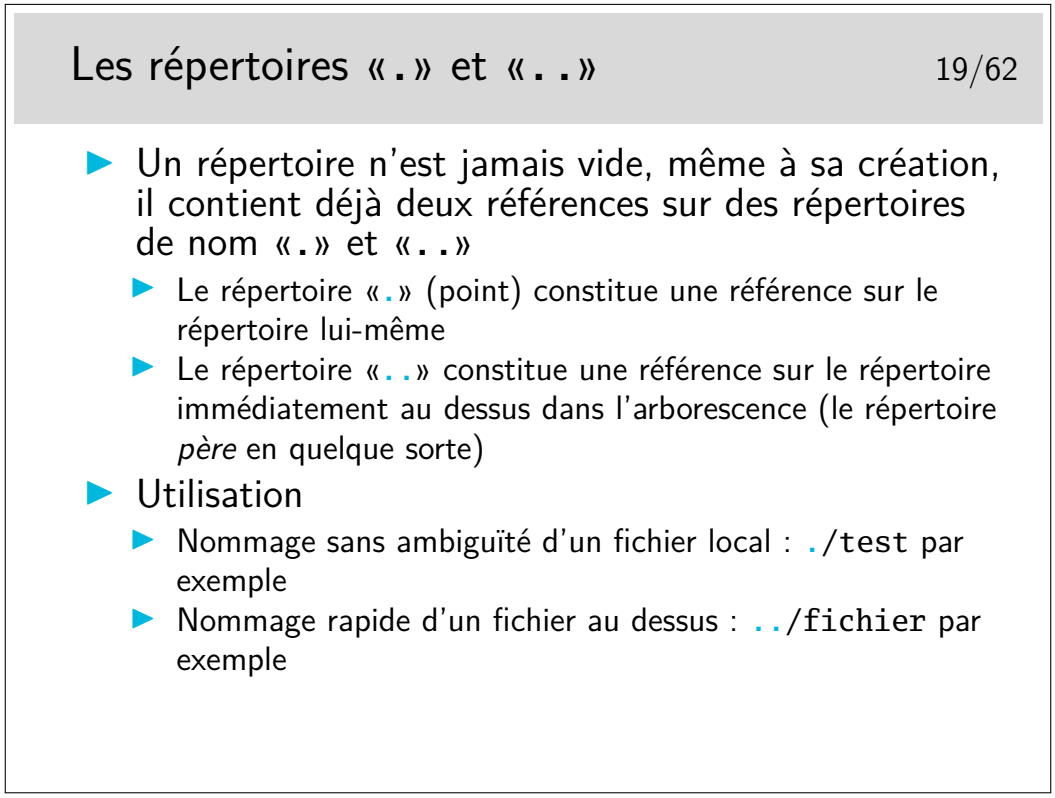

Un lien est un nom de fichier, une référence dans un répertoire. Un fichier peut être référencé plusieurs fois, on dit alors qu'il a plusieurs liens. Ces liens peuvent être symboliques ou physiques (on dit alors «lien dur»). Un lien dur est tout simplement une référence dans un répertoire sur un fichier déjà référencé. Les deux références sont associées au même inode. Dans le schéma ci-dessus on pourrait par exemple créer un lien sur `fichierB` dans un répertoire quelconque. `fichierB` serait alors accessible par deux chemins. Les deux noms auraient le même inode.

La commande `ln` permet de créer des liens durs ou symboliques.

Les liens symboliques sont des fichiers presque normaux, seul leur type est spécifique. Leur contenu est le chemin d'accès au fichier réel.

Les liens durs ne peuvent être effectués qu'à l'intérieur d'un même système de fichiers (point commun : l'inode). Les liens symboliques peuvent sans problème franchir les frontières physiques des systèmes de fichiers.

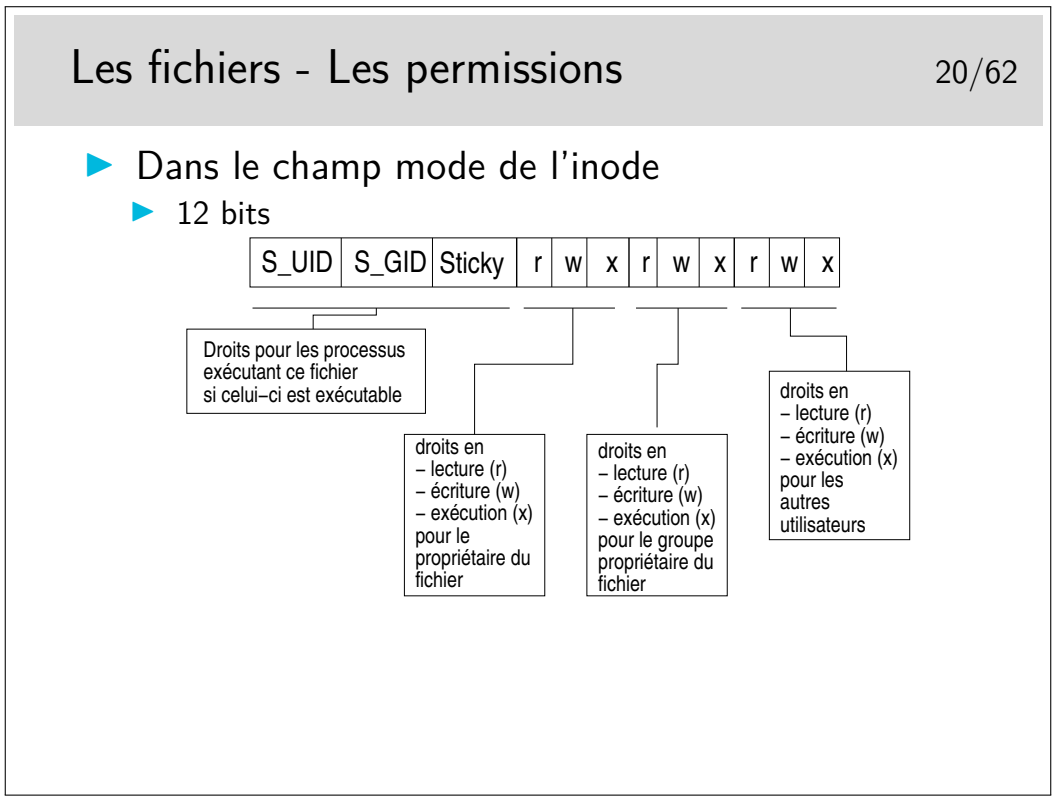
Conceptuellement, un lien peut être considéré comme un raccourci sous Windows (si vous connaissez W...)




Il arrive à beaucoup de gens d'écrire un petit fichier d'essai et de le nommer `test`. Après qu'il soit rendu exécutable, il est essayé... Et...Surprise ! le résultat n'est pas du tout celui attendu... Car la commande lancée, de nom `test`, n'est pas celle qu'on croit, il s'agit de `/bin/test` et non le test du répertoire courant (si la variable d'environnement `PATH` ne contient pas le caractère «`.`» dans sa liste avant `/bin`).

Il suffit pour remédier à cela d'entrer alors la ligne de commande `./test` et le tour est joué.

En général, par défaut, le «`.`» ne figure pas dans la variable `PATH` pour des raisons de sécurité élémentaire et toute commande se situant dans le répertoire de travail (celui dans lequel on se trouve) doit se lancer avec la séquence `./nomFichier` (à moins que le répertoire de travail ne soit listé naturellement dans la variable `PATH`).




Si les droits en écriture/lecture/exécution sont assez évidents à comprendre il n'en va pas de même pour les trois premiers : `S_UID`, `S_GID` et le `Sticky bit` :

— Ces droits concernent les processus d'exécution du fichier, ils ne sont actifs qu'au moment où le fichier est en exécution. Ils précisent ce que le fichier (si l'on peut dire) a le droit de faire lorqu'il s'exécute.

— Lorsqu'une commande est lancée, si celle-ci correspond à un fichier exécutable, ces droits agissent comme suit :

  — si le bit *S_UID* est positionné, le processus d'exécution a les droits du propriétaire du fichier et non ceux de l'utilisateur qui a lancé la commande,

  — si le bit *S_GID* est positionné, le processus a les droits du groupe propriétaire du fichier

  — si le *sticky bit* est positionné, le processus ne sera pas purement et simplement effacé de la mémoire, il sera recopié en zone de swap et si la commande correspondant est rappelée, son rechargement en mémoire sera plus rapide. Ce n'est plus beaucoup utilisé.

Remarque importante : le *sticky bit* est maintenant utilisé sur des répertoires ouverts en lecture/écriture à tous pour restreindre le droit d'effacement des fichiers qui s'y trouvent au seul propriétaire de ces fichiers. Cas des répertoires `/tmp`, `/var/tmp`, `/usr/tmp`. Voir aussi la notion d'attribut avec la commande `chattr`.

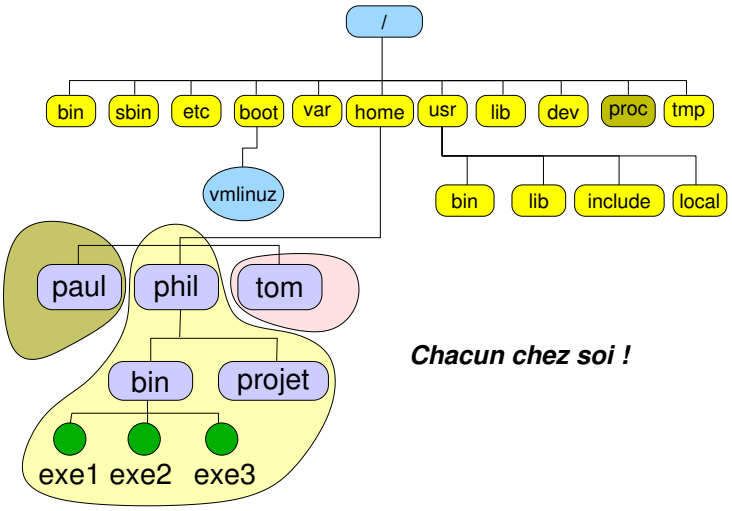
## Les droits sur les répertoires 21/62

- Ce sont les mêmes types de droits que pour les fichiers : `r`, `w` et `x` et même `t`
- La sémantique associée est toutefois différente
- Droits :
  - `r` : le répertoire est lisible, on peut lister son contenu
  - `w` : le répertoire est «écrivable», on peut y créer des fichiers ou des répertoires
  - `x` : le répertoire n'est pas exécutable, il est accessible : on peut aller dedans, ou le traverser pour accéder à ce qu'il contient (fichiers, sous-répertoires)
  - `t` : le *sticky bit*, valable pour un répertoire ouvert en `w` à tout le monde, indique que seul un propriétaire de fichier peut supprimer ce fichier.

Si un répertoire est interdit de passage (i.e. le droit `x` est ôté), on ne peut pas s'y déplacer, on ne peut pas non plus descendre dans ses sous-répertoires, même si ceux-ci sont autorisés. Un droit `x` ôté est comme un obstacle infranchissable.

Si un répertoire à le droit `w` pour les utilisateurs non propriétaires, n'importe qui peut y créer un fichier, mais n'importe qui peut aussi supprimer ce fichier. Ce serait le cas pour les répertoires contenant les fichiers temporaires (`/tmp`, `/var/tmp`) si ceux-ci n'étaient pas munis du droit `t`.

## Les fichiers et répertoires et les utilisateurs 22/62




Chacun chez soi... Et chacun maître chez soi. Si un utilisateur veut ouvrir son répertoire à tout le monde il en a le droit. Il a aussi le droit de fermer son répertoire à tous (même à lui

même, ce qui est embêtant pour lui sur le moment, mais il peut modifier à nouveau les droits pour se ré-autoriser...).

Les répertoires ouverts en lecture sont visitables pour autant qu'ils aient le droit `x` positionné. Les répertoires `/tmp`, `/var/tmp`, `/usr/tmp` sont ouverts à tous et tous peuvent y créer des fichiers et les supprimer. Généralement ces répertoires sont munis du *sticky bit* afin de restreindre le droit d'effacement aux seuls propriétaires des fichiers.

Il est donc possible de «se promener» dans la plus grande partie du système de fichiers. Seuls certains répertoires et fichiers sensibles sont interdits.

## Les fichiers spéciaux 23/62

- Référencent des périphériques
  - Permettent les échanges (lectures/écritures) avec les pilotes des périphériques
  - Permettent le contrôle de ceux-ci
- Deux types
  - Les périphériques en mode bloc
    - les échanges se font par bloc d'octets (par «pages»)
  - Les périphériques en mode caractère appelé encore mode transparent (on dit plutôt mode *raw*)
    - les échanges se font octet par octet
  - Les disques sont plutôt en mode bloc, les terminaux en mode *raw*

## Les fichiers spéciaux - le répertoire /dev 24/62

- Exemple d'entrée dans /dev

```
[bash]$ cd /dev; ls -l hda1
brw-rw---- 1 root  disk   3,  1  mar 24 2001  /dev/hda1
```

**Indique le mode**
b : bloc
c : caractère

**Nombre majeur**
indique le numéro du pilote (driver) dans le noyau

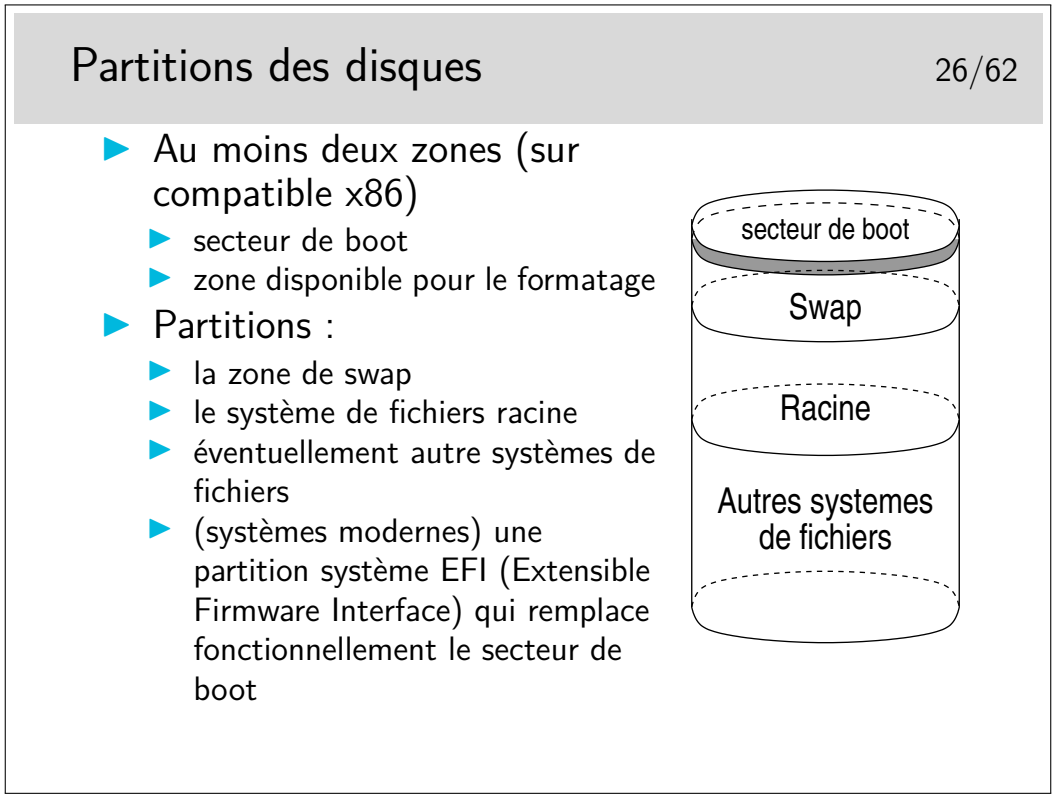
**Nombre mineur**
indique le numéro du périphérique piloté par le driver

Ces fichiers sont créés par la commande `mknod`, ils peuvent être créés n'importe où mais en

général on les trouve dans `/dev`.

Cette manière de voir les périphériques permet de considérer ceux-ci comme des fichiers (certes spéciaux, mais fichiers quand même), ainsi les échanges entre les applications et les périphériques se font essentiellement par des écritures et des lectures, comme pour des fichiers normaux.

## 2.2 Organisation sur disques




bootstrap : en français «amorçage», la séquence d'opérations à suivre pour démarrer le SE (OS).

zone de swap : en français «zone d'échange», permet de «décharger» la mémoire centrale si celle-ci est saturée. Les processus endormis (en attente d'un événement quelconque) sont transférés dans cette zone. Sa taille peut influencer le fonctionnement d'applications «gourmandes» en mémoire. Il est recommandé qu'elle soit au moins égale à la taille de la mémoire centrale.

On peut avoir plusieurs systèmes de fichiers de natures différentes : FAT, NT ou linux (ext2, ext3, etc...). Attention danger pour l'intégrité des partitions NT si autorisation en écriture.

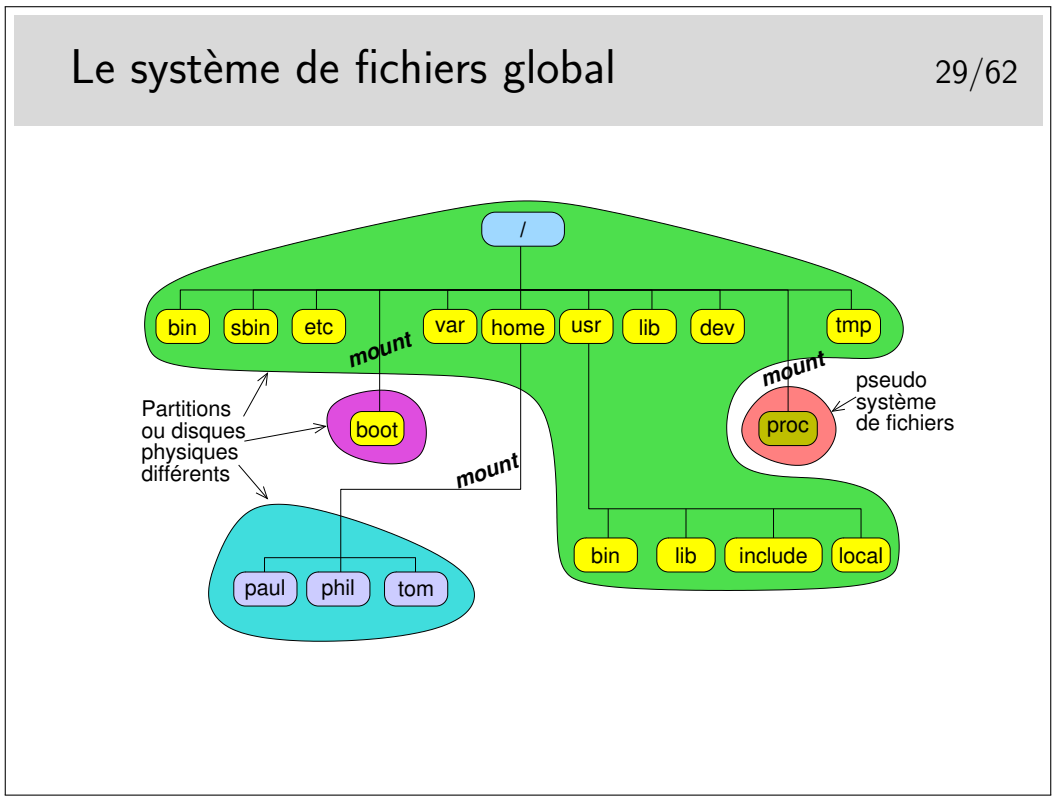
EFI : une evolution du BIOS (Basic input/output system), qui lui rajoute des fonctionnalites

## Création d'un système de fichiers 27/62

- Commande `mke2fs` (*make extended 2 File System*)
  - Issue de la commande standard Unix `mkfs`
  - Crée un système de fichiers sur une partition de type Linux (type fixé au formatage du disque via `fdisk` par exemple)
  - Le système créé est de type `ext2`, ou `ext3` (s'il est journalisé), ou `ext4` (s'il est gros, jusqu'à 1024 péta-octets)
  - Permet de fixer la taille des blocs (par défaut 4Ko), le nombre d'inodes, etc.
    Tous ces paramètres peuvent être fournis par l'administrateur sinon leur valeur est estimée en fonction de la taille de la partition (nombre d'inodes par exemple)

## Contrôle d'un système de fichiers 28/62

- Commande `fsck` ou similaire (`e2fsck`, `fsck.ext2`)
  - Vérifie la cohérence d'un système de fichiers Linux
    - Vérifie les super blocks, les inodes, les fichiers perdus (inode affecté mais pas de référence du fichier dans un répertoire). Ces fichiers sont recréés dans un répertoire appelle `lost+found` avec un nouveau nom autogénéré comportant le numéro de l'inode
  - Commande lancée automatiquement au boot. Elle vérifie d'abord si la partition a été correctement démontée lors de l'arrêt précédent, si oui, elle se termine, sauf si le nombre de montage maximum est atteint (voir transparent suivant)
  - Utilisable à tout instant, «à la main», sur un système de fichiers non monté de préférence ou monté en mode lecture seule (read only)




Le système de fichier global peut être constitué de plusieurs systèmes de fichiers de natures identiques ou différentes.

Dans le schéma ci-dessus on a choisi de créer 3 partitions formatées en système de fichier Linux (`ext2` ou `ext3`) :

- la partition racine avec `/bin`, `/etc`, `/var`, `/usr`, `/lib`, `/dev`, `/tmp`, ...
- une partition pour `/boot` (fait par défaut sous RedHat ou Debian par exemple)
- une partition pour les utilisateurs
- proc est un système virtuel de fichiers, utilisé par le noyau Linux. il n'existe pas sur disque mais est créé en mémoire vive.
- système de fichiers `/sys` avec le noyau 2.6, donne accès aux informations sur le système (les périphériques (devices) et leurs drivers).

L'utilitaire permettant de raccrocher un système de fichiers à l'arborescence globale est la commande `mount`.

Lorsqu'un utilisateur insère une clé USB, le gestionnaire de fichier utilisé (de façon cachée) `mount`. Le resultat : le volume apparaît alors comme par magie sur votre bureau.

# 3 Utilisation courante

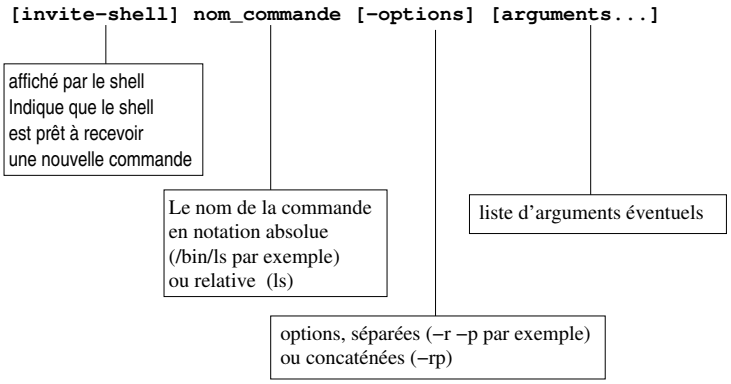
## 3.1 Les commandes et leur syntaxe

### Comment lancer une commande 31/62

- Manière moderne
  - Via l'interface graphique
- Manière moins moderne... mais ô combien efficace !
  - Via un émulateur de terminal (xterm, gnome-terminal, ...)
  - Dans un terminal virtuel de console (accessible via `Ctrl-Alt-F1` à `F6` depuis l'interface graphique)
  - Ces méthodes lancent un processus interpréteur de commandes associé au terminal, appelé *Shell*
  - Le *Shell* est une interface entre le clavier et le système, il offre des facilités pour entrer les commandes mais aussi un vrai langage de programmation

### Les commandes 32/62

- Syntaxe générale



## Les commandes - La variable PATH 33/62

- Le PATH est une variable d'environnement
- Contient la liste des répertoires où se trouvent les commandes qu'on peut appeler par leur nom relatif le plus court : `ls` au lieu de `/bin/ls` par exemple
- Si le système vous dit : «*command not found*», il se peut que votre variable PATH soit mal configurée...
- Configuration dans le shell directement ou dans `$HOME/.bashrc` :
  - `export PATH=$PATH:/usr/local/bin` (par exemple)

Quand vous appelez une commande, le SE doit savoir ou la chercher. Pour cela, il va regarder la variable PATH.

Si on modifie la variable `PATH` directement dans le Shell par la commande `export` ci-dessus, la modification est prise en compte dans CE shell seulement... Pas dans le shell de la fenêtre terminal d'à côté.

Pour qu'une modification d'environnement soit permanente il suffit qu'elle soit écrite dans un fichier de configuration pris en compte systématiquement lors du lancement du Shell. Le fichier personnel `.bashrc` est le plus approprié pour le shell `bash`; et le fichier `.tcshrc` (ou `.tcsh.PERSO`) pour le shell `tcsh`.

## Les commandes - Quelques raffinements 34/62

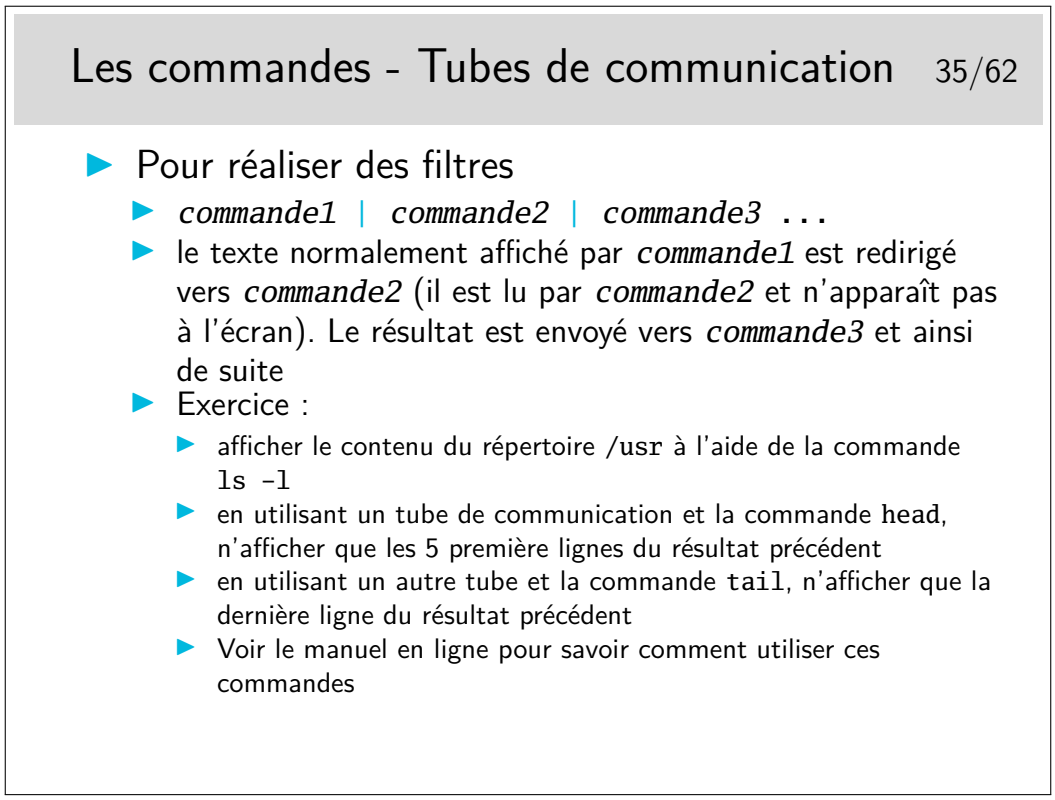
- Les commandes en arrière-plan (*background*)
  - la ligne de commande se termine par le caractère &
  - le shell lance la commande et redonne la main aussitôt sans attendre qu'elle se termine. La commande poursuit son exécution en arrière-plan
- Les redirections
  - redirection du fichier standard de sortie
    - *commande* > *fichier*
  - Redirection du fichier standard d'erreur
    - *commande* 2> *fichier*
  - redirection du fichier standard de sortie et d'erreur
    - *commande* > *fichier* 2>&1

To combine stderr and stdout into the stdout stream, we append this to a command :
2>&1

File descriptor 1 is the standard output (stdout). File descriptor 2 is the standard error (stderr).

At first, 2>1 may look like a good way to redirect stderr to stdout. However, it will actually be interpreted as "redirect stderr to a file named 1".

& indicates that what follows and precedes is a file descriptor, and not a filename. Thus, we use 2>&1. Consider >& to be a redirect merger operator.



Les commandes - Tubes de communication 35/62

- Pour réaliser des filtres
  - *commande1* | *commande2* | *commande3* ...
  - le texte normalement affiché par *commande1* est redirigé vers *commande2* (il est lu par *commande2* et n'apparaît pas à l'écran). Le résultat est envoyé vers *commande3* et ainsi de suite
  - Exercice :
    - afficher le contenu du répertoire `/usr` à l'aide de la commande `ls -l`
    - en utilisant un tube de communication et la commande `head`, n'afficher que les 5 première lignes du résultat précédent
    - en utilisant un autre tube et la commande `tail`, n'afficher que la dernière ligne du résultat précédent
    - Voir le manuel en ligne pour savoir comment utiliser ces commandes

C'est en fait la philosophie de base de l'utilisation d'Unix : une pléiade de petits utilitaires que l'on assemble au moyen de tubes pour réaliser une grande tâche. Le contraire des approches monolithiques.

Note : lors de l'exécution de `commande1 | commande2`, les deux programmes sont exécutés en parallèle. Sur d'autres systèmes mono-tâche comme MS-DOS, il y a d'abord l'exécution du premier programme en sauvegardant sa sortie dans un fichier temporaire, puis ensuite l'exécution du second programme en lisant le fichier temporaire...

## Les commandes – Les jokers (*wildcards*) 36/62

- Un caractère spécial qui en remplace plusieurs...
  - Pour les noms de fichiers
  - Un genre d'expression rationnelle (mais non POSIX)
- Exemple : `rm *` (efface les fichiers du répertoire)
- Reconnus par le shell :
  - `?` : n'importe quel caractère
  - `*` : zéro ou plusieurs caractères
  - `[A1x]` : le caractère `A` ou `1` ou `x`
  - `[a-z]` : les caractère de `a` à `z` (code ASCII)
  - `{ab,ac}` : la chaîne `ab` ou `ac`

## Les commandes Unix classiques 37/62

(liste non exhaustive...)

- Créer, naviguer parmi les fichiers et répertoires
  - `ls cd pwd cp mv rm mkdir`
- Afficher — éditer des fichiers
  - `more less` — `nano`
- Filtres texte
  - `echo cat grep sort uniq sed tail tee head cut tr split paste printf`
- Comparaison de fichiers
  - `comm cmp diff patch`

## Les commandes Unix classiques 38/62

...

- Administration basique (niveau utilisateur)
  - `chmod chown ps su w who`
- Communication
  - `mail telnet ftp finger ssh`
- Shells
  - `sh csh ksh zsh bash tcsh fish`

Ce ne sont quelques commandes classiques que tout utilisateur d'Unix finit par connaître par coeur au bout de quelque temps de pratique...

Même si il est évidemment difficile de les connaître toutes dans le détail, il est bon de savoir qu'elles existent et de savoir retrouver leur documentation en temps utile.

Ces commandes sont rangées typiquement dans `/bin` et `/usr/bin`

`http://en.wikipedia.org/wiki/List_of_Unix_programs`

## 3.2 La documentation

## Les commandes - La documentation 40/62

- La commande `man`... Savoir lire la syntaxe de man...
  - Par exemple : `man rm`

```
RM(1)        Manuel de l'utilisateur Linux    RM(1)
NOM
       rm - Effacer des fichiers.
SYNOPSIS
       rm  [-dfirvR] nom...
```

Nom de la commande

Liste des options
Entre [] signifie que ces opttions ne sont pas obligatoires.
(Si on les indique cependant, on ne mettra pas les crochets)

Le ou les arguments.
Ici les caractères ... indiquent que nom peut être répété plusieurs fois

Les pages du «manuel» Unix, en ligne (Sun fournissait un gros classeur du man imprimé...).

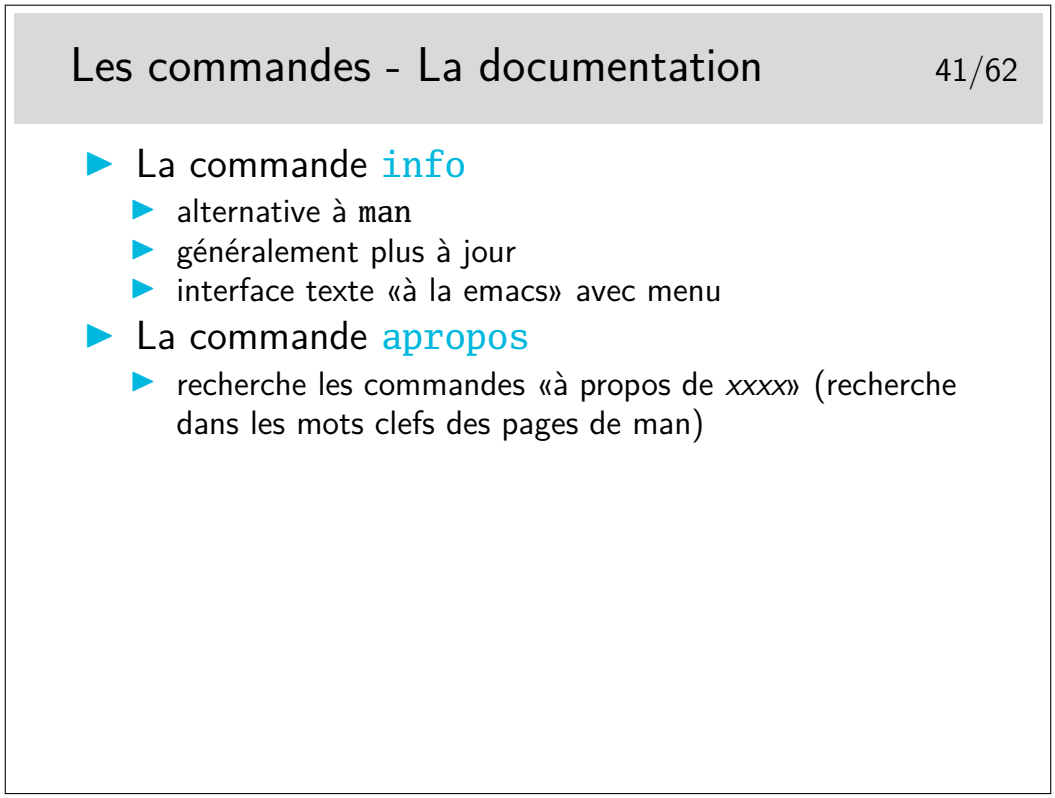
La source d'information de référence incontournable. Ne pas poser une question dans les forums de discussion dont la réponse est dans le man... sous peine d'être bien mal accueilli :-)

Attention : les pages du man ne sont pas des cours ni des tutoriels, mais du condensé d'information. Chaque mot est important. Il est parfois difficile de se plonger dedans, mais les pages de man sont toutes rédigées selon la même manière. relativement facile de s'y retrouver rapidement.

Le man est organisé en sections :

1. Commandes utilisateur
2. Appels système
3. Fonctions de bibliothèque
4. Fichiers spéciaux
5. Formats de fichier
6. Jeux
7. Divers
8. Administration système
9. Interface du noyau Linux

Chaque section possède une page d'introduction qui présente la section, disponible en tapant `man <num_section> intro`.




Note : les commandes *internes* au shell (p.ex. `cd pwd history` etc.) n'ont pas de page de `man` car ce ne sont pas des programmes, mais juste des mots clefs reconnus et interprétés directement par le shell. Ces commandes sont donc expliquées dans la documentation du shell lui-même. Par exemple, si vous cherchez de la documentation sur la commande `cd` et que vous avez *bash* comme shell, regardez dans le `man bash`. Une autre façon de faire est de taper `help cd` (En effet, `help` est une autre commande interne de *bash* qui affiche de la documentation sur les commandes internes de *bash*.)

### Les outils de recherche de fichiers 42/62

- La commande `locate`
  - recherche les occurrences de la chaîne de caractères qui lui est passée en arguments dans une base de données mise à jour via la commande `updatedb`
- La commande `find`
  - Recherche n'importe quoi n'importe où EN TEMPS REEL (lent)
- La commande `whereis`
  - Recherche le nom de commande passé en argument dans un certain nombre de répertoires standards
- La commande `which`
  - Recherche dans le `PATH` où se trouve la commande indiquée en argument

# 4 Les processus

## 4.1 Environnement, cycle de vie

### Les processus 44/62

- Le concept de processus
  - Un processus est un programme en cours d'exécution, dans un environnement donné, dans la mémoire centrale.
- Notion d'environnement
  - Ensemble d'informations complémentaires au programme en exécution qui viennent paramétrer l'exécution
  - Essentiellement trois types d'information d'environnement
    - Les variables d'environnement
    - L'identité de l'utilisateur au nom duquel s'exécute le processus
    - Les fichiers ouverts (notamment ceux d'entrée/sortie standard)

## Création d'un processus 45/62

- Un processus est toujours créé par le noyau, à la demande d'un autre processus
- Le processus qui demande la création est appelé le père, le processus créé est appelé le fils
- Le processus fils est créé en mémoire centrale dans une zone mémoire distincte du processus père
- Le processus fils est la copie intégrale du processus père. Mais un détail technique (PID : Process IDentifier) permet au développeur de différencier les deux processus.

Le fils est un clone du père mais il y a une mutation génétique... Le père demande la création du fils via une fonction de bas niveau (un appel système). Cette fonction rend 0 dans le processus fils et une valeur strictement positive dans le processus père. Cette valeur dans le père est en fait le numéro de processus du fils. Le programmeur profite de cette différence pour différentier le code exécuté par les deux processus. Sinon les deux exécuteraient strictement la même chose.

## Les variables d'environnement 46/62

- Chaînes de caractères, en majuscules par convention
  - Syntaxe `NOM=valeur`
  - Regroupées dans un espace mémoire appelé «tableau des variables d'environnement»
  - Ce tableau est hérité par les processus fils et résiste à l'exécution d'une commande (voir plus loin)
- Quelques variables standard
  - `PATH`, `HOME`, `USER`, `LOGNAME`, `DISPLAY`, ...

## Identité utilisateur associé au processus 47/62

- Deux identités d'utilisateur !
  - Utilisateur réel (qui a lancé le processus), identifié par son numéro d'utilisateur dans `/etc/passwd` (*ruid : real user ID*)
  - Utilisateur effectif :
    - Dans la majorité des cas il s'agit de l'utilisateur réel
    - Si le fichier exécuté (qui a donné naissance au processus) a le bit *S_UID* positionné (une lettre s apparaît à la place du x des droits du propriétaire du fichier) alors l'utilisateur effectif est le propriétaire du fichier exécuté (*euid : effective user ID*)
    - Attention : trou de sécurité potentiel, surtout si le fichier appartient à root
- Deux identités de groupe
  - Concept identique à ci-dessus : groupe réel, groupe effectif (bit *S_GID*)

Pendant l'exécution d'un fichier ayant le bit *S_UID* positionné on a les droits du propriétaire du fichier, les droits d'un autre utilisateur, sans lui demander... Mais si le fichier exécuté possède ce droit c'est que son propriétaire l'y a mis, car lui seul peut le faire, ou l'administrateur, ou une application malicieuse...

Tout utilisateur peut avoir chez lui de tels fichiers pour des raisons qui lui sont propres et pour que des applications qui lui sont personnelles puissent fonctionner pour tout le monde. L'administrateur peut toutefois demander aux utilisateurs de justifier la présence de tels fichiers, c'est son droit et peut être même son devoir.

De tels fichiers existent et appartiennent à root, l'administrateur. S'ils sont vulnérables à des attaques de type débordement de tampon (buffer overflow) alors la sécurité du système entier est gravement compromise.

## Les fichiers standards d'entrée-sortie 48/62

- Trois fichiers standards
  - Le fichier standard d'entrée (descripteur 0, FILE Pointer stdin)
  - Le fichier standard de sortie (descripteur 1, FILE Pointer stdout)
  - Le fichier standard de sortie d'erreur (descripteur 2, FILE Pointer stderr)
- Ouverts par défaut lors du lancement d'un exécutable
- Associés virtuellement au clavier pour l'entrée standard et à l'écran pour les deux autres
- Ils peuvent être redirigés vers des fichiers réels ou des tubes de communication

## Contrôle sur les processus 49/62

- Lister les processus
  - La commande `ps`
    - Nombreuses options : `ax`, `axl`, `axf`, `-ef`, etc.
  - La commande `top`
    - Comme `ps axu`, avec réaffichage régulier, plus des informations sur la charge et l'occupation mémoire
- Arrêter un processus
  - Si on a le contrôle (processus en premier plan dans un terminal)
    - Sans le tuer (arrêt momentané) : `<Ctrl-Z>` (`fg` pour resumer)
    - En le supprimant : `<Ctrl-C>`
  - La commande `kill` (voir pages suivantes)

Sur certains systèmes les paramétrages des terminaux ou des émulateurs de terminaux sont tels que les associations de touches `<Ctrl-Z>` ou `<Ctrl-C>` ne fonctionnent pas. On peut le vérifier avec la commande `stty -a` qui affiche le paramétrage du terminal.

Exemple :

```
[bash]$ stty -a
speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
```

```
lnext = ^V; flush = ^O; min = 1; time = 0;
...
```

Remarquer le paramètre `intr = ^C`, l'accent circonflexe indique la touche `<Ctrl>`. En fait, `intr` est le paramètre permettant de tuer rapidement un processus. Voir aussi `susp`. Et faire `man stty`.

On peut changer le paramétrage `intr` avec : `[bash]$ stty intr ^F`

## Contrôle sur les processus : les signaux 50/62

- Un signal est une interruption logicielle envoyée à un processus par le noyau après un événement
- L'événement peut être :
  - Une faute logicielle (ex : division par 0, manipulation d'une adresse mémoire interdite, erreur d'alignement de donnée)
  - Terminaison d'un processus fils : par défaut (mais paramétrable) le père est prévenu
  - Intervention de l'utilisateur via le shell ou l'interface graphique pour tuer le processus ou le stopper ou autre (modification de la taille d'une fenêtre par exemple)
- Dans la plupart des cas le signal est fatal au processus

## Les principaux signaux 51/62

| Signal | Numéro | Fonction |
|---|---|---|
| HUP | 1 | Signal envoyé au processus en premier plan associé à un terminal lorsque celui-ci est fermé |
| INT | 2 | Envoyé depuis le clavier avec la combinaison de touches <CTRL-C> (par défaut) |
| QUIT | 3 | Envoyé depuis le clavier avec la combinaison de touches <CTRL-\|> (par défaut) |
| KILL | 9 | Ne peut être intercepté, envoyé depuis le clavier via la commande `kill` (`kill -9` ou `kill -KILL`) |
| TERM | 15 | Envoyé via le clavier par la commande `kill` simple |
| SEGV | | Erreur de segmentation, accès à une zone mémoire interdite |
| CLD | | Terminaison d'un fils |
| WINCH | | Modification de la taille de la fenêtre associée à l'application |
| STOP | | Arrêt du processus sans le terminer. Envoyé via la combinaison de touches <CTRL-Z> |
| URG | | Une données urgente a été reçue via le protocole TCP et est en attente de lecture (voir le cours sur la programmation réseau) |
| IO | | Des données réseau sont arrivées et sont en attente de lecture. (voir le cours sur la programmation réseau) |
| USR1 | | Nom de signal utilisable par le développeur, à son gré |
| USR2 | | Nom de signal utilisable par le développeur, à son gré |

Les raccourcis claviers (p.ex. `<CTRL-C>`) pour envoyer les signaux au processus en cours dans le shell sont paramétrés au niveau du terminal : `stty -a`.

## La commande `kill` 52/62

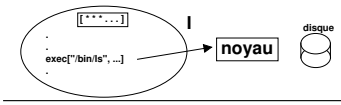
- `kill` *`[numéro_ou_nom_de_signal] numéro_processus | numéro_job`*
- le numéro ou le nom de signal sera en général omis sauf si le résultat est négatif, auquel cas on pourra essayer le signal `KILL` (-9) qui ne peut pas être intercepté par le processus
- Le numéro de processus sera obtenu par `ps`
- Le numéro de job n'est valable que pour les processus en arrière plan (background) ou les processus stoppés. On peut le connaître avec la commande `jobs`
- Le programmeur de l'application peut gérer l'arrivée des signaux (sauf le signal 9) et les ignorer ou les traiter, pour que le processus se termine proprement ou ne se termine pas

## Les processus zombies 53/62

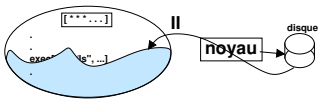
- Lorsqu'un processus fils se termine, son père doit acquitter la terminaison. C'est un problème de programmeur, pas d'utilisateur. Si l'acquittement n'est pas fait, le processus terminé reste dans la liste des processus (état `Z`), il est appelé «zombie»
- Un processus zombie est un processus fils pour lequel son père n'a pas acquitté la terminaison
- Le processus zombie est vidé de sa substance mais reste dans la liste des processus de la machine et peut être listé par `ps`
  - On ne peut plus le supprimer, il faut supprimer le père pour que le zombie disparaisse
  - Il est généralement dû à une erreur de programmation

En Shell (`sh`, `bash`, `ksh`) on peut récupérer le code de retour du exit du fils dans la variable `$?`.

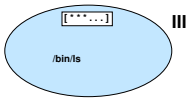
## Exécution d'un fichier par un processus 54/62




Étape I : le processus appelle exec en indiquant un fichier exécutable en paramètre

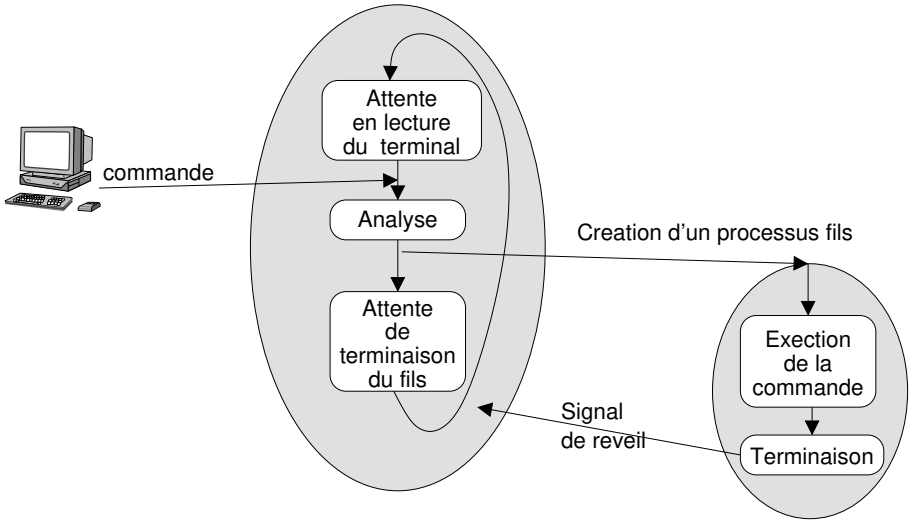



Étape II : le noyau va chercher le fichier sur le disque et le recopie à l'endroit où réside le processus. Le code original de celui-ci est écrasé et remplacé par le code du fichier




Étape III : le processus commence l'exécution de son nouveau code. Il ne peut y avoir retour à l'ancien code
On remarquera qu'un élément a résisté à l'envahisseur... Il s'agit du tableau des variables d'environnement qui n'a pas été altéré. Il est néanmoins possible de remplacer ce tableau par un nouveau.

## Le principe du Shell 55/62




— Remplacer les méta caractères (*, ?, etc) par ce qu'il sont censés représenter (donc construire proprement la commande à exécuter) ;

— Vérifier si la commande est un alias, si oui, «désaliasser» ;

— Vérifier si la commande est une fonction interne du Shell (commande interne).

Si la commande correspond à un fichier exécutable, alors un processus fils est créé pour l'exécution. Le Shell ne reste en attente que si la ligne de commande entrée au terminal ne se termine pas par un caractère «&», auquel cas le shell revient sans attendre et renvoie l'invite à l'écran.

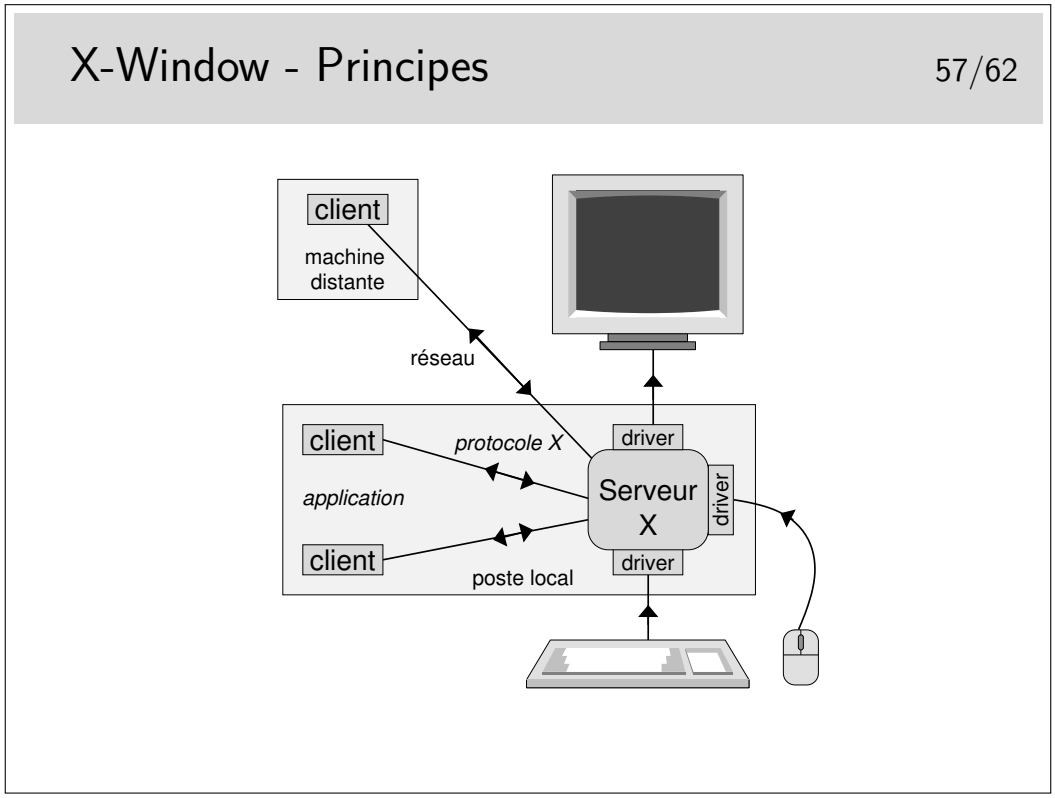
Dans ce dernier cas, il est néanmoins capable de gérer proprement la terminaison de son fils afin que ce dernier ne reste pas zombie.

# 5 L'interface graphique X-Window

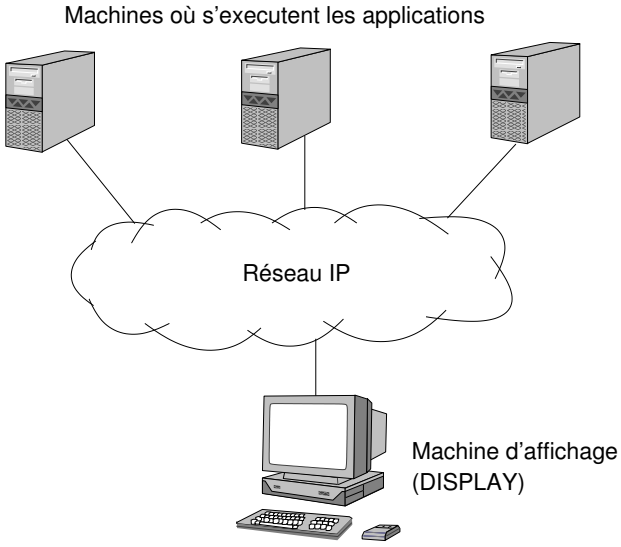
Le principe d'une *interface graphique* émerge à partir des années 80, du fait de l'évolution du matériel, avec différents concepts et architectures logicielles (client-serveur, fenêtres, multi-application, etc.). (Voir `https://en.wikipedia.org/wiki/History_of_the_graphical_user_interface`)

Dans le monde Unix, on utilise un système graphique client-serveur avec une couche d'adaptation X-Window.

## 5.1 Client-serveur, authentification, bureau



## X-Window - Intrinsèquement Réseau 58/62




## X-Window - Applications en réseau 59/62

- Toute application X peut s'exécuter sur une machine et s'afficher sur une autre
- La machine d'affichage est indiquée dans une option (`-display`) ou une variable d'environnement (`DISPLAY`)
  - Exemples :
  - `$ xterm -display lune:0.0`
  - `$ export DISPLAY=lune:0.0`
  - `$ xterm`
  - format du `display` : *adresseOuNomMachine*:*numServ*.*numEcran*

## X-Window - Réseau et sécurité 60/62

- Par défaut il n'est pas possible d'afficher des fenêtres sur un «display» utilisé par un autre utilisateur si celui-ci n'a pas donné l'autorisation
- Les autorisations sont possibles avec la commande `xhost`
  - `xhost +terre`
  - autorise les «connexion graphiques» depuis la machine `terre`
  - ⊕ authentification par adresse IP
- Autorisations avec la commande `xauth`
  - plus complexe mais plus *sûr*
  - fichier `.Xauthority`
  - ⊕ autentifie un utilisateur (qui doit posséder le bon *cookie* de 128 bits)

## X-Window - Le serveur et les applications 61/62

- Le serveur X sait gérer l'affichage mais il ne sait pas quoi ni comment afficher !
  - ce sont les applications qui l'informent via le protocole X
  - des librairies applicatives pour dessiner boutons, menu, assenceurs, etc. (*graphic toolkits*)
- Le serveur X reçoit tous les événements clavier et souris
  - il informe alors les applications de l'événement si celles-ci ont demandé qu'il leur soit envoyé
  - les applications traitent l'événement et décident de ce qu'il faut faire, elles demandent éventuellement des modifications d'affichage au serveur

Une documentation (particulièrement pédagogique et ludique) sur le fonctionnement interne de X-Windows : `https://magcius.github.io/xplain/article/`

## X-Window - Une application très particulière 62/62

- Le gestionnaire de fenêtre ou *Window Manager*
  - le serveur X **ne sait pas gérer** les fenêtres
  - application spécifique nécessaire : **Window Manager**
    - permet de déplacer, iconifier, restaurer, supprimer les fenêtres et aussi modifier leur taille
    - offre des menus de fond d'écran
  - Les Window Managers existent en grand nombre
  - Les applications sont normalement compatibles avec tous les Window Managers (vœu pieux !)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# TAF DCL

TP 1 - Unix/Linux

Automne 2023

## 1 Découverte de l'interface graphique

La distribution de Linux utilisée est une Ubuntu. L'interface graphique choisie est basée sur le standard Gnome, plus précisément il s'agit de la mouture *Gnome-Shell*[1] Cette interface graphique est hautement configurable, par les utilisateurs, mais également par les éditeurs de distribution Linux. Mais si l'aspect visuel peut changer d'une version de Linux à l'autre, le fonctionnement général restera le même (certains spécialistes avancent le terme d'interface *intuitive*...). [2]

Une fois que vous avez ouvert votre session graphique, vous avez [3] :
— une barre en haut de l'écran : avec à gauche quelques menus, à droite quelques icônes, au centre l'heure ;
— une barre sur la gauche de l'écran : sur le haut quelques raccourcis et applications favorites, sur le bas une icône qui ouvre un (court) catalogue d'applications ;
— un "coin magique" en haut à gauche qui vous permet de gérer vos bureaux virtuels (appelés également "espace de travail").

Entre ces deux barres, au centre de l'écran, vous trouverez votre *bureau*, là où s'afficheront les fenêtres de vos applications.

### 1.1 Votre espace disque dans le système complet

On vous demande d'ouvrir votre *Dossier Personnel* (c.à.d. tous les fichiers de votre espace disque) : soit cliquez directement sur l'icône sur votre bureau s'il est présent, soit passez par le menu *Emplacement* dans la barre du haut.

Vous devez voir apparaître une nouvelle fenêtre : l'explorateur de fichiers Nautilus [4] Cet explorateur vous permet de visualiser *vos fichiers* et *vos dossiers* (que l'on appelle des *répertoires* en informatique scientifique).

La barre du haut de Nautilus comporte :
— au centre : l'emplacement, là où se trouvent les fichiers listés dans la fenêtre principale ;
— à gauche de l'emplacement : deux flèches pour évoluer dans la hiérarchie des emplacements ;
— à droite de l'emplacement : trois boutons pour paramétrer le mode d'affichage (en liste, en damier, ou d'autres options) ;
— et dans le coin tout à droite : l'accès au menu de Nautilus.

Créons un premier fichier.

Dans la fenêtre principale, visez une zone où il n'y a rien, et cliquez avec le bouton droit. Un menu contextuel apparaît. Faites `Nouveau document`. [5] Par exemple, appelez votre nouveau document `essai1`. (Vous pouvez également passer par le menu principal de Nautilus, en cliquant sur le bouton en haut à droite.) Faites `Nouveau document > Document vide`. Constatez le résultat dans l'explorateur.

Sélectionnez le fichier dans l'explorateur par un clic de souris avec le bouton droit. Sélectionnez `Propriétés`. Répondez aux questions suivantes :
— Qui est le propriétaire de ce fichier ?
— Ce fichier appartient aussi à un groupe d'utilisateurs, quel est ce groupe ?
— Quelles sont les permissions (on dit aussi *droits*) sur ce fichier ?
— Quelle est l'application par défaut que Nautilus va lancer pour `Ouvrir` ce fichier ? (essayez, vous verrez bien)

1. Une autre mouture de Gnome est l'interface *Mate*. Mais il existe bien d'autres interfaces : `https://en.wikipedia.org/wiki/Desktop_environment`
2. N'hésitez pas à chercher des tutoriels sur Internet.
3. C'est tout du moins comme cela que cela apparaît à l'heure où j'écris ces lignes...
4. *Nautilus* est parfois appelé tout simplement *"Fichiers"* dans certains menus...
5. Notez qu'avec la version actuelle de Nautilus, ce menu n'apparaît pas si le répertoire `~/Modèles/` est vide...

— Regardez les `Propriétés` de ce fichier. Est-ce que l'on vous parle de *type* ?

En fait, la notion de *type de fichier* n'existe pas au niveau du système de fichier Unix (à la différence de Mac OS) ni au niveau du noyau et encore moins au niveau de l'extension du fichier (comme sous Windows) mais uniquement au niveau de l'application bureau Gnome : Gnome essaie de deviner quelle application doit être lancée pour chaque fichier (un mécanisme dérivé de MIME [6]), mais la notion de *type* s'arrête bien là.

**Permissions Unix :** Sous Unix, chaque fichier est déclaré comme appartenant à un utilisateur *propriétaire*, et à un *groupe* d'utilisateurs. Il y a une dernière catégorie : les *autres* utilisateurs, c'est-à-dire qui ne sont ni le propriétaire du fichier, ni membre du groupe.

Chacune de ces catégories d'utilisateurs dispose de droits sur ce fichier spécifiant les actions qu'ils sont autorisés ou non à faire dessus. Les actions possibles sont : *la lecture*, *l'écriture*, *l'exécution.*

Ceci est souvent représenté de manière textuelle par les fameux trois triplets : `rwx rwx rwx`
(Qui signifient : `Read Write eXecute` pour *le propriétaire*, *le groupe*, *les autres.*)

Seul le propriétaire peut changer ces permissions. Le propriétaire peut changer le groupe parmi l'un des groupes auquel il appartient. Seul l'administrateur peut changer le propriétaire d'un fichier, ajouter ou supprimer des utilisateurs à un groupe.

Notons que les administrateurs Unix ont souvent l'habitude de créer autant de groupes que d'utilisateurs : pour chaque nouvel utilisateur, il existe un groupe qui porte le même nom, avec un seul membre (l'utilisateur lui-même), et qui est le groupe par défaut de l'utilisateur en question.

## 1.2 Fichiers et répertoires, les vôtres et les autres

Explorons l'arborescence du système de fichiers...

Dans la fenêtre principale de Nautilus, faites `<Ctrl-L>` (ou via le menu principal de Nautilus, puis `Saisir un emplacement`). Dans la barre du haut, le champ `Emplacement` vous offre la possibilité de changer d'emplacement en tapant au clavier et non plus à la souris (ouf !).

— Quel était le nom du répertoire qui s'est affiché naturellement lorsque vous avez lancé l'explorateur ? Ce répertoire est votre répertoire d'accueil (*home directory*).
— Regardez les `Propriétés` d'un fichier ou d'un répertoire ; quel est le symbole séparateur dans les noms des répertoires affichés dans l'item `Emplacement` ?

Placez-vous à la racine du système de fichiers et tentez de créer un nouveau répertoire (clic droit).
— Quel est le problème ?
— Y a-t-il une solution ?

Promenez-vous dans l'arborescence et retrouvez votre répertoire d'accueil. Dans la barre d'`Emplacement`, tapez le nom d'un autre répertoire (par exemple `/usr/bin`).
— Que vient-il de se passer ?

## 1.3 Les fichiers cachés

Retournez rapidement dans votre répertoire d'accueil avec l'icône `Dossier personnel`. Modifiez les paramètres d'affichage de Nautilus (à droite, dans la barre du haut, le bouton qui représente une flèche qui descend), et cochez `Afficher les fichiers cachés`.

Vous venez de faire apparaître de nouveaux fichiers dans votre répertoire d'accueil. Tous ces fichiers ont un nom qui commence par un caractère *point*. Ce sont les *fichiers cachés* sous Unix. On voit qu'ils ne sont pas très bien cachés, c'est en fait une simple convention alors que sous Windows c'est un attribut système du fichier. Vous pouvez constater qu'il n'y a pas que des fichiers, il y a aussi des répertoires cachés.

Les fichiers et répertoires cachés contiennent généralement des informations d'environnement. Les répertoires dont le nom commence par `.config` contiennent par exemple des informations sur votre environnement graphique, ils pourront contenir la mémorisation de votre écran en cours lorsque vous vous déconnecterez.

Les fichiers et répertoires cachés sont souvent un peu équivalents à ce qu'on appelle la *base de registres* sous Windows. Beaucoup d'applications mémorisent leurs paramétrages et les préférences de l'utilisateur dans ce type de fichiers. Ils constituent en quelque sorte votre base de registres personnelle.

6. Cela peut vous sembler magique ? Ça l'est un peu puisque l'information se cache derrière `man magic` !

## 1.4 Les fonctions copier/couper/coller

Comme sous Windows il est facile de sélectionner du texte à l'aide de la souris et ensuite de le recopier ailleurs ou de le couper. On sélectionne à l'aide du **bouton gauche** de la souris de la manière suivante [7] :

— Lancez un navigateur web (soit en cliquant sur l'icône Firefox, ou via le menu Applications). Chargez une page web quelconque (la page par défaut peut faire l'affaire).
— Dans cette page, pointer le début du texte à sélectionner, appuyer sur le bouton gauche et *tirer* la sélection jusqu'à la fin de la zone.
On peut aussi cliquer deux fois sur un mot pour le sélectionner ou trois fois pour sélectionner la ligne.
— Avec Nautilus, double-cliquez sur votre fichier `essai1`. Nautilus lance l'application Gedit pour l'éditer.
— La fonction *coller* s'obtient en se déplaçant à la destination (ici, la fenêtre Gedit) et en cliquant sur le **bouton milieu** de la souris.

Si vous avez déjà écrit des pages web, vous savez que les données que vous avez sélectionnées à la souris ne se limitent pas à ce simple texte mais qu'il est décoré de nombreuses balises HTML. En fait, lors de cette opération de copier-coller les deux applications négocient un format commun, en l'occurrence l'éditeur texte Gedit demande du texte ASCII simple. Ceci explique que certaines applications peuvent proposer du `Collage sans mise en forme` et autre `Collage spécial`...

Selon les environnements il est aussi possible d'utiliser les combinaisons de touches classiques suivantes (attention, pas toujours effectifs) :

— copier : `<Ctrl-C>`
— coller : `<Ctrl-V>`
— couper : `<Ctrl-X>`

Généralement ces fonctions sont aussi accessibles via le menu `Édition` lorsque celui-ci est visible dans la barre de menu des applications.

## 1.5 Utilisation de la corbeille

Dans le navigateur Nautilus revenez dans votre répertoire d'accueil si vous n'y êtes pas déjà. Pointez sur le fichier `essai1` et appuyez sur le bouton droit de la souris. Sélectionnez `Mettre à la corbeille`.

Vous pouvez aussi déplacer *physiquement* le fichier dans la corbeille en le *tirant* à l'aide de la souris en appuyant cette fois sur le bouton gauche et en le laissant tomber sur l'icône représentant la corbeille sur le bureau (opération de *drag-and-drop* ou *glisser-lâcher*).

Toujours dans l'explorateur Nautilus, avec l'option `Afficher les fichiers cachés`, déplacez-vous maintenant dans le répertoire `.local/share/Trash`. Fouillez dans les sous-répertoires. Que constatez-vous ?

Comment supprimer réellement ce fichier ?

Notez que ce répertoire `Trash` sera créé la première fois que vous placerez un fichier à la poubelle, vous ne le verrez probablement pas avant.

**Problème de quotas.** Même si vous ne voyez pas les disques physiques, ceux-ci existent bel et bien sur une machine serveur dans le réseau. Leur taille n'est pas extensible à l'infini. Tout le monde doit pouvoir travailler. Vous êtes donc limité en espace disque par des quotas. Nous verrons plus loin comment visualiser quels sont les vôtres. Pour l'instant nous constatons que les fichiers qui semblent effacés par déplacement dans la corbeille ne sont pas physiquement effacés de votre espace disque. Si vous n'effacez pas physiquement vos fichiers, c'est-à-dire avec un *Vider la Corbeille*, votre corbeille va remplir petit à petit vos quotas et vous ne pourrez plus travailler. [8] Vous ne pourrez même plus vous connecter, car l'ouverture d'une session (graphique) nécessite justement de créer quelques fichiers !

## 1.6 Les sauvegardes de `.snapshot`

La *corbeille* offre un service rassurant pour les étourdis, celui de pouvoir *récupérer* un fichier qui a été mis à la corbeille. Mais comprenez bien que cette corbeille est une application, un bout de programme qui s'exécute dans votre session graphique (et donc, pas dans une session texte), qui offre ce service en déplaçant les fichiers dans un répertoire caché. Ce n'est pas du tout une fonctionnalité du système d'exploitation Unix : lui, ce qu'il permet, c'est de supprimer un fichier ; et quand c'est supprimé, c'est supprimé !... sauf si l'on a un système de sauvegarde...

À l'école, un système de sauvegarde à plusieurs niveaux a été mis en place. Les fichiers utilisateurs sont dans une baie de stockage qui utilise des disques RAID (pour pallier aux défaillances matérielles). Le système

7. Ou le bouton de droite si vous avez préféré configurer votre souris en mode gaucher via l'icône de paramétrage en haut à droite.
8. Vous pouvez consulter l'état de vos quotas dans le menu `Applications > Accessoires > Synthèse des quotas`

de gestion de fichiers utilisé est *ReiserFS* qui offre un service de journalisation des données. Ces données sont sauvegardées sur bande régulièrement (le soir, typiquement).

Bref, si vous avez perdu vos données, il devrait être possible de les récupérer, mais ça peut être plus ou moins compliqué suivant les cas. Et la manière la plus simple, c'est le `.snapshot` de *ReiserFS*.

Avec Nautilus revenez dans votre répertoire d'accueil. Repérez le répertoire caché `.snapshot`. Il est possible que ce répertoire ne soit pas présent pour le moment ; il y a un mécanisme d'*auto-montage* qui ne va le charger que si l'on a besoin d'y accéder. Dans ce cas, tapez son nom dans la barre d'`Emplacement` de Nautilus. Vous y trouvez divers sous-répertoires, typiquement : `hourly.0/ hourly.1/` ... `nightly.0/ nightly.1/`

Promenez-vous là-dedans. Chacun de ces sous-répertoires est une *vue* de vos fichiers tels qu'ils étaient il y a une heure, deux heures, la nuit dernière, etc. Bref, si vous avez fait une petite bêtise et supprimé un fichier par mégarde, le plus simple est d'aller le rechercher ici (plutôt que d'aller trouver les administrateurs système de l'école pour leur demander de ressortir les bandes de sauvegarde et de lancer une restauration...)

### 1.7 Le répertoire `Bureau` (`Desktop`)

— Avec Nautilus, allez dans le sous-répertoire `Bureau` de votre répertoire de travail. Maintenant, cliquez avec le bouton droit sur votre fond d'écran et faites `Nouveau document > Document vide`. Que constatez-vous ?
— Mettez ce fichier à la corbeille. Que constatez-vous ?

### 1.8 Quitter sa session

Lorsque l'on souhaite quitter sa session, il est préférable d'adopter l'attitude suivante :
— Allez à l'imprimante ramasser tout ce que vous avez imprimé et que vous n'êtes pas encore allé chercher...
— Faites le tour de vos applications et fermez-les proprement (sauvegarder vos fichiers), sans quoi votre environnement va les fermer brutalement sans forcément faire ce que vous auriez aimé qu'il fasse. Pensez également à vérifier chacun de vos bureaux virtuels (soit via le *coin magique* en haut à gauche, soit via l'icône en haut à droite qui vous propose la liste de vos *Espaces de travail* ; ce TP n'a pas parlé de ces choses-là vous êtes libre de les découvrir par vous-même).
— Cliquez sur le bouton en haut à droite de l'écran, cliquez sur votre nom (en fait le nom de votre session), et choisissez *Fermer la session*.

  Attention : si vous ne faites pas *Fermer la session* mais que vous cliquez sur l'icône en forme de verrou, vous ne quittez pas votre session, vous la verrouillez... Ce n'est pas tout à fait la même chose...
— Éteignez l'écran, cela fera quelques économies d'énergie.

### 1.9 Bilan de l'environnement graphique

Nous venons de réaliser un petit nombre de manipulations avec l'environnement Gnome et l'outil Nautilus. Le fonctionnement que vous avez pu expérimenter est globalement similaire à ce que vous pouvez connaître avec d'autres environnements graphiques sous Unix/Linux (ou même Windows pour certains aspects). N'oubliez pas que ce n'est qu'un environnement possible parmi plusieurs dizaines, qui sont eux-mêmes configurables à l'infini... Bien évidemment, à l'usage vous aurez vos petites habitudes, vos préférences, et vous vous sentirez peut-être plus confortable avec certains environnements qu'avec d'autres. Même si vous trouverez l'un meilleur que l'autre, votre souci en tant que futur ingénieur devra être de savoir s'adapter à n'importe quel environnement...

## 2 L'interface en ligne de commande

Le travail à l'aide des interfaces graphiques est souvent facile et intuitif, même si elles sont finalement assez mouvantes (le nombre de boutons à cliquer, leur nom, le rôle, peut varier d'une version à l'autre...). Cependant sous Unix il existe un très grand nombre d'outils accessibles de préférence en *ligne de commande* dans un terminal ou un émulateur de terminal. Ce sont *les commandes Unix*. Il est souvent préférable de les utiliser pour un travail rapide et efficace, mais leur manipulation nécessite un apprentissage.

### 2.1 Notion de Shell et de commande

Ouvrez un émulateur de terminal : dans la barre de menus du haut, choisissez `Applications`. Là vous trouverez l'application `Terminal` soit dans l'onglet `Favoris`, soit dans l'onglet `Utilitaires` [9]

Une fenêtre apparaît à l'écran. Elle affiche un petit texte abscons appelé l'*invite* ou *prompt* qui comme son nom l'indique vous invite à saisir une commande. En fait, vous avez deux choses à ce niveau : vous avez (1) une

9. Cela peut changer d'une version à l'autre de l'environnement graphique. Parfois c'est `Accessoires` ou `Système`.

application graphique qui sait afficher une fenêtre, gérer un petit ascenseur de défilement, afficher des caractères ASCII et sous différents encodages, et vous avez (2) un interpréteur de commande qui porte le nom de Shell.

Le Shell associé à votre fenêtre vous demande d'entrer une commande au clavier. Lorsque vous allez le faire, il va analyser cette commande puis lancer son exécution. Cette commande peut être complexe, elle correspond généralement à un fichier exécutable, mais on peut paramétrer son exécution de différentes manières. L'analyse préalable à l'exécution consiste à prendre en compte ce paramétrage.

## 2.2 Les commandes Unix et le Shell

Pour ce qui suit, vous pouvez vous aider des *"Ressources documentaires"* proposées sur Moodle.

### 2.2.1 Les répertoires `.` et `..`

Dans la fenêtre d'émulation de terminal tapez la commande `ls -al`. Vous devez voir apparaître la liste du contenu du répertoire dans lequel vous êtes, incluant les fichiers et répertoires cachés. Par rapport à la liste affichée dans le navigateur Nautilus vous avez en plus les répertoires `.` et `..`

Exécutez les commandes suivantes :

— `pwd` (print working directory)
— `cd ..` (change directory)
— `pwd`
— `cd .`
— `pwd`

Qu'en déduisez-vous sur le rôle de ces répertoires :

— le répertoire `..`
— le répertoire `.`

Petit quizz freudien : peut-on avoir `.=..` ?

### 2.2.2 La racine

Remontez au niveau le plus haut de l'arborescence (soit avec de multiples `cd ..`, soit avec `cd /`), puis utilisez la commande `ls -ld`.

— Comment s'appelle la racine ?
— Qui est le propriétaire de la racine de l'arborescence ?
— Quels sont vos droits sur ce répertoire ?

### 2.2.3 Sauter de branche en branche

Placez-vous dans le répertoire `/usr/bin`. (Commande `cd`)
Sautez dans le répertoire `/usr/lib` (puis revenez) en utilisant chacune des deux méthodes : le *nommage absolu* et le *nommage relatif*.
Amusez à refaire la même chose en sautant de `/usr/share/man/man1` à `/usr/share/man/man2` et réciproquement.

— Dans quel cas peut-on préférer le *nommage absolu* et dans quel cas peut-on préférer le *nommage relatif* ?

### 2.2.4 Vos fichiers et ceux des autres

— Revenez à votre répertoire de travail : `cd`
— Remontez au répertoire parent : `cd ..`
— Vérifiez tout ça : `pwd` puis `ls`
  En principe, vous devriez voir un sous-répertoire, celui à votre nom, votre répertoire de travail.
— Essayez d'aller dans le répertoire d'un collègue : `cd` *`login_du_collègue`*
— Vérifiez tout ça : `pwd` puis `ls`
— Remontez au répertoire parent : `cd ..`
— Vérifiez tout ça : `pwd` puis `ls`
  C'était comme ça avant ? Que s'est-il passé ?

Vos fichiers ne sont pas présents physiquement sur chacun des PC d'école. Ils sont localisés dans une baie de stockage. Chacun des PC peut y accéder en réseau. Lorsqu'un PC en a besoin, il procède à un montage, puis, après un certain temps d'inutilisation les fichiers sont démontés. C'est automatique et transparent. Typiquement, lorsque l'on se connecte à un PC, il a besoin de nos fichiers, et procède au montage. Mais si l'on demande à accéder au répertoire du collègue, il procède au montage également. (Ensuite, le fait que le collègue ait positionné des permissions Unix sur ses fichiers est une autre histoire.) C'est un bon moyen pour partager simplement des fichiers (pas besoin de clef usb ou de transfert réseau, c'est déjà en réseau).

### 2.2.5 Les méta-caractères

Lancez la commande suivante : `ls -l /bin/mk*`
— Que constatez-vous ?
— En déduire le rôle du caractère `*`
— Lancez maintenant la commande `ls -ld /usr/bin/zi?` et comparez avec `ls -ld /usr/bin/zi*`
— En déduire le rôle du caractère `?`
— Comparez par exemple `ls -ald .*` et `ls -ald .????*`

### 2.2.6 La complétion

Apprenez à taper plus vite que votre ombre !

**Note :** l'utilisation de la complétion dépend fortement du type de shell que vous avez sur votre système (tapez `echo $SHELL` pour savoir).
— Entrez la commande `ls -l /bin/mk` suivie immédiatement par la touche <Tab>. La touche tabulation est à gauche du clavier, et souvent représentée ⇆
Que constatez-vous ?
— Faites à nouveau <Tab> et voyez le résultat. Quelle aide est alors apportée ?
— Complétez la commande avec quelques caractères pour lever toute ambiguïté et réutilisez la touche <Tab> pour afficher finalement automatiquement le nom `/bin/mkdir`

Notez que la complétion est une fonctionnalité de confort offerte par le shell que vous utilisez (`bash`, `tcsh`, etc.). Avec des shells plus anciens comme `sh`, cela ne marche pas... Avec les plus modernes on peut faire de la complétion très subtile, sur des noms d'options de commande ou des arguments (par exemple après `man` on n'aura de la complétion que sur des noms de commande dont on a le manuel en ligne) ou encore de la correction orthographique.

### 2.2.7 Les redirections

— Retournez dans votre répertoire de travail en tapant simplement : `cd`
— Demandez l'heure : `date`
Que constatez-vous ?
— Redirigez maintenant la sortie de la commande précédente dans le fichier `maintenant.txt` dans votre répertoire d'accueil : `date > maintenant.txt`
Que constatez-vous ?
— Listez le contenu de votre répertoire de travail : `ls -l`
Alors ?
— Affichez le contenu de ce fichier : `cat maintenant.txt`
Comparez avec la date maintenant : `date`

### 2.2.8 Les tubes de communication

Combien de fois avez-vous tapé la commande `cd` depuis tout à l'heure ?
Non, ne cherchez pas dans votre tête, l'ordinateur a meilleure mémoire que vous.
— Consultez l'historique de votre interpréteur de commande : `history`
C'est à la fois le contenu du fichier caché `~/.bash_history` et le contenu de ce qu'il a en mémoire depuis que vous avez ouvert votre terminal. Lorsque vous fermez votre terminal, ce fichier `.bash_history` est mis à jour.
— Tout s'affiche trop vite à l'écran. Paginez via la commande `more` ou la commande `less`[10] de la manière suivante : `history | less`
Vous pouvez paginer avec les touches classiques <Page Up> <Page Down>. De plus, pour diverses raisons historiques (les terminaux n'ayant pas toujours eu des claviers si sophistiqués...), vous pouvez utiliser la touche <Espace> pour afficher page par page ou avec la touche <Entrée> pour afficher ligne par ligne. Pour sortir vous tapez <Q> (Quit).
— Bon, reprenons, nous n'allons pas faire le travail à la main, mais le faire faire par l'ordinateur. C'est pour cela qu'il est là. Faisons-le chercher les lignes qui contiennent `cd` :
`history | grep cd`
— Il ne reste plus qu'à lui faire compter les lignes :
`history | grep cd | wc -l`

10. Le monde d'Unix est un monde d'humour, en particulier au niveau des noms de commande. Un humour un peu élitiste parfois...

#### 2.2.9 Les commandes en premier et arrière plan (*foreground*, *background*)

— Lancez la commande `xeyes`. Dans le même terminal, lancez maintenant une commande quelconque (`pwd` par exemple). Que constatez-vous ?
— Faites <`Ctrl-C`> pour tuer la commande `xeyes` et relancez-là en arrière plan (cherchez comment faire soit dans le cours, soit sur la page web indiquée précédemment). Que constatez-vous comme différence ? Que pouvez-vous faire, alors, qui vous était impossible lorsque la commande était lancée en premier plan ?

### 2.3 La documentation des commandes Unix

Ayez le réflexe RTFM : *Read That Fine Manual*. Chaque commande Unix est documentée dans le manuel Unix en ligne et accessible via la commande `man`.

Dès que vous avez un problème avec une commande, faites : `man` *`nom_de_la_commande`*

Vous pouvez chercher un terme dans la page de manuel en tapant `/mot_a_rechercher` qui indique la première occurrence du terme. Pour visualiser les occurrences suivantes, utilisez la touche `n` (*next*). Pour quitter la page de manuel, utilisez `q` (*quit*). (En fait ces touches sont gérées par votre *pageur*, typiquement la commande `more` ou `less`, et dont vous pouvez lire le man...).

Chaque documentation est structurée de la même manière. Vous trouverez essentiellement :
— le nom de la commande et sa description courte ;
— un synopsis, résumé de la manière d'utiliser la commande ;
— une description plus complète du fonctionnement de la commande, avec souvent des exemples concrets d'utilisation ;
— la liste des options acceptées, avec leur signification ;
— à la fin de la page de manuel, la section SEE ALSO peut vous guider vers d'autres commandes connexes.

Dans la suite de ce TP (et de manière générale), prenez le temps et la peine de consulter ces pages de manuel. (J'insiste car c'est comme cela que l'on apprend beaucoup !)

À l'aide du manuel de référence, répondez aux questions suivantes :
— Comment faire une copie récursive de fichiers et de répertoires (commande `cp`) ?
— Comment afficher le contenu d'un répertoire dans l'ordre alphanumérique inverse (commande `ls`) ?
— Quelles sont les options de la commande `man`, et plus particulièrement, que fait l'option `-k` ?
— Quelles sont les pages du manuel traitant du mot *listing* ?
— Quelles sont les pages du manuel traitant du mot *image* ?

Notez également que l'interpréteur de commande (le shell) comprend également quelques commandes internes (`cd pwd ..`). C'est donc dans le `man` de votre shell que vous trouverez des explications sur ces commandes. De plus certains shell modernes peuvent vous fournir de l'aide en ligne sur leurs commandes internes. Par exemple sous `bash` on peut faire `help cd`.

### 2.4 Paramétrage de votre environnement de travail en mode terminal

Votre environnement est déjà configuré pour vous permettre d'utiliser la plupart des commandes classiques. Cependant, si l'on veut utiliser une commande non classique (p.ex. pour faire un exercice de TP particulier, où parce que c'est une commande que l'on a installée soi-même), il faut paramétrer un certain nombre de choses dans notre environnement.

La chose la plus importante est de savoir où se trouve cette commande. Ainsi on va positionner une variable de votre environnement de façon à ce que le shell trouve cette commande sans problème. La variable en question est `PATH` (en majuscules) et contient la liste des répertoires dans lesquels se trouvent les commandes que l'on peut appeler par leur nom simple. Pour consulter son contenu, utilisez la commande `"echo $PATH"`.

Notez que cette variable `PATH` peut être vide, par exemple si votre shell a été installé (et compilé) avec un chemin par défaut comme `/bin:/usr/bin`

#### 2.4.1 La commande `SETUP`

La modification environnementale à effectuer est complexe pour le néophyte, aussi, à l'école, nos administrateurs système ont développé une commande spéciale appelée `SETUP` (en majuscules) qui permet de faciliter les choses.
— Exécutez cette commande pour lister les logiciels accessibles
— Affichez le *chemin* connu par votre environnement pour rechercher les commandes utilisateur : `echo $PATH`
— Faites la commande `java -version`, et repérez la version par défaut de votre java.
— À l'aide de la commande `SETUP`, demandez à changer de version (p.ex. `SETUP JAVA17` ou `SETUP JAVA18`)

— Vérifiez que désormais vous travaillez avec une autre version de java : `java -version`
— Vérifiez les changements dans votre configuration : `echo $PATH`

**A** ttention, la commande `SETUP` ajoute mais n'enlève pas... Par exemple, si dans un même terminal vous faites `SETUP JAVA17` puis `SETUP JAVA18`, vous vous retrouvez avec un environnement configuré pour rechercher *d'abord* la version 1.7 de java, et ensuite la version 1.8... et qui donc va toujours trouver d'abord java 1.7 ! (Pour vous en convaincre, faites des `java -version` ou `echo $PATH`.)

Ceci dit, rien ne vous empêche de faire `SETUP JAVA17` dans un terminal, et `SETUP JAVA18` dans un autre...

### 2.4.2 Le fichier `.profile.PERSO`

Ouvrez une nouvelle fenêtre terminal et retentez d'exécuter la commande `java -version`. Vous devez à nouveau constater le problème. La modification apportée par la commande `SETUP` a été prise en compte dans le shell associé à la fenêtre précédente mais pas dans celui associé à la nouvelle. Il faut donc recommencer, c'est fastidieux, donc le contraire de l'informatique. Pour éviter ce genre d'ennui il faut que le paramétrage soit pris en compte chaque fois qu'on lance un shell (donc chaque fois qu'on ouvre un terminal). Pour cela il faut écrire la commande de paramétrage dans le fichier `.profile.PERSO` (fichier propre à l'environnement Unix à l'école).

Mais pourquoi donc ce fichier-là ? Lors de la question sur les tubes vous avez notamment pu afficher le nom de votre shell de login. Le shell par défaut à l'école est `bash`. (On peut le remplacer par l'un des shells déclarés dans le fichier `/etc/shells` à l'aide de la commande `chsh` ou `ypchsh`, lire le `man` avant.) Si vous lisez attentivement le `man bash` vous comprendrez qu'au démarrage de votre session shell (juste après votre login) il exécute entre autres le fichier `.profile`. Ce fichier est déjà configuré par les administrateurs de l'école, et a priori on ne le modifie pas. Mais si vous le lisez, vous constaterez que la dernière instruction consiste à exécuter les commandes qui se trouvent dans un autre fichier, le fichier `.profile.PERSO`. Ceci explique cela... Sachez donc ce que vous faites avant de modifier votre shell par défaut ou votre fichier `.profile`... En particulier testez dans de nouvelles fenêtres avant de vous déconnecter car il est facile de scier la branche sur laquelle repose votre environnement et vous par la même occasion.

Donc, dans l'état actuel des choses, le fichier `.profile.PERSO` est le bon endroit pour des paramétrages personnels. Modifiez ce fichier pour y incorporer la bonne commande `SETUP` indiquée à la question précédente. Ouvrez une nouvelle fenêtre terminal (qui va lancer un nouveau shell qui va exécuter `.profile.PERSO`), et tentez de lancer de nouveau la commande `java -version`. Cela devrait marcher cette fois et les suivantes... (Notez que vous n'aurez peut-être pas besoin de l'application `jcvs` tout de suite.)

# 3 L'impression de documents

L'impression est un problème important pour tout établissement, c'est un centre de coût non négligeable. Il en va de même pour une école, il faut que nous en soyons tous bien conscients (élèves et personnels), d'ailleurs on se voit imposer des quotas...

L'impression pose aussi des problèmes d'utilisation car il existe plusieurs outils pour la mettre en œuvre. Les documents créés via des outils de traitement de texte sont imprimables à l'aide des outils eux-mêmes (typiquement menu `Fichier > Imprimer`). C'est le cas pour OpenOffice (outil similaire aux outils Office de Microsoft) ou pour Mozilla ou Firefox (navigateurs Web).

Les imprimantes étant généralement accessibles en réseau, il convient de connaître le nom *réseau* de celle que l'on souhaite utiliser. Aussi, les administrateurs ont généralement la bonne idée de coller une petite étiquette sur chaque imprimante indiquant son nom. Les imprimantes disponibles depuis les salles Unix/Linux portent les noms suivants : `IMP-DF-200`, `IMP-DF-300`, etc. Imprimez donc préférentiellement sur l'imprimante qui est dans votre salle.

Notez également que les administrateurs de l'école ont configuré votre environnement de telle manière que l'imprimante *par défaut* de votre session s'appelle `POUBELLE`, et qui est une imprimante virtuelle qui n'imprime rien. Cela évite que les étourdis cliquent sur *Imprimer* sans se poser de question.

## 3.1 La commande a2ps

La plupart des applications permettant de manipuler des documents offrent des fonctions natives pour imprimer ces documents. Pour de simples fichiers textes, tels que des listings de programmes vous pouvez utiliser la commande **a2ps** (*Ascii to Postscript*) qui a le bon goût d'imprimer en deux pages par feuilles et d'embellir le contenu imprimé en fonction de son type (par exemple si c'est un source de programme, mettre en gras les mots clés, numéroter le lignes...) et être hautement paramétrable.

Syntaxe : `a2ps -P`*`imprimante nom_du_fichier_listing`*

Le mot *imprimante* est bien entendu à remplacer par le nom réel de l'imprimante destinatrice. En principe, chaque poste est configuré de telle sorte que l'imprimante par défaut soit l'imprimante la plus proche. Dans ce cas on peut se passer de l'option `-P`*imprimante*.

La commande `a2ps` a de nombreuses options (voir le man). Par exemple pour une impression avec numérotation des lignes il faut rajouter l'option `-line-numbers=`$x$. Si $x$ vaut 1, toutes les lignes seront numérotées, s'il vaut 5, les numéros figureront toutes les 5 lignes...

Questions :

— Dans quelle salle vous trouvez-vous et quelle est l'imprimante la plus proche ?
La meilleure méthode pour le savoir est d'aller voir. D'ailleurs, il faudra bien faire le chemin pour aller récupérer ses impressions, alors autant repérer l'itinéraire maintenant.

— Imprimez votre fichier `.profile.PERSO` ou tout autre fichier contenant quelques lignes de code.
Non, je rigole, pas la peine de gaspiller du papier si vous n'en n'avez pas besoin. Mais faites semblant.

Note : `a2ps` ne sait malheureusement pas gérer nativement l'encodage *UTF-8* utilisé pourtant de plus en plus pour gérer les caractères accentués ou autres caractères non purement *ASCII*... Aussi, les commandes `iconv` ou `recode` peuvent vous aider à convertir votre document texte encodé en *UTF-8* dans l'encodage *latin1* voir *latin9*, avant d'utiliser `a2ps`...

## 3.2 Autres commandes

— Pour connaître les imprimantes accessibles depuis son poste : `lpstat -v`
— Pour savoir si l'une de ces imprimantes a été désignée comme imprimante par défaut : `lpstat -d`
— Pour envoyer directement à l'imprimante des fichiers dans un langage qu'elle comprend (ASCII, PostScript) : `lp -d` *imprimante fichier*
— Pour lister les travaux en cours : `lpstat -t`
— Pour annuler un travail (possible pour l'utilisateur qui a lancé le travail et depuis le même poste) : `cancel` *numéro_travail* (Le numéro du travail est obtenu avec `lpstat -t`)
Note : la commande `lprm` est équivalente à la commande `cancel`

**Plus de papier !** Vous observerez qu'à proximité de chaque imprimante une petite affichette vous indique où vous procurer du papier s'il en manque. On évitera donc d'aller piller le bac à feuilles d'une autre imprimante, et on ira chercher du papier là où il se doit, et sans rouspéter «c'est toujours moi qui y vais»... Bien évidemment, les personnes qui peuvent vous procurer des ramettes de papier ont des horaires. Il faut donc anticiper. Si vous découvrez à 19h30 qu'il n'y a plus de papier pour imprimer les trucs super importants, on ne pourra pas vous aider. Vous observerez également qu'il y a des périodes de l'année comme ça où tout le monde a besoin d'imprimer son rapport ou son CV le même jour (soir)... Anticipez.

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# FIP SIT151

TP 2 - Unix/Linux

Automne 2023

**Résumé**

Cette seconde série d'exercices est à faire en autonomie. Des salles de TP Linux sont accessibles en libre service. La grosse majorité des questions est également faisable sur un PC Linux quelconque ; vous pouvez préférer utiliser votre propre machine si vous le souhaitez.

Comptez environ 3h pour faire ces exercices correctement. Certaines personnes déjà accoutumées à l'environnement Linux prendrons moins de temps (il ne s'agit pas de bâcler non plus). D'autres personnes moins accoutumées à l'outil informatique prendrons plus de temps. Avancez à votre rythme.

Il est impératif d'avoir fait ces exercices **avant** la seconde séance de TP programmée à votre emploie du temps. Cette courte séance sera l'occasion de débriefer ces exercices tous ensemble, avec un enseignant.

## 1 Travail sur les répertoires et fichiers

Vous êtes connecté(e). Lancez une fenêtre émulant un terminal. Dans cette fenêtre une chaîne de caractères (le prompt) vous indique que vous pouvez saisir une ligne de commande. Dans le tableau qui suit, vous trouverez en colonne de gauche sur chaque ligne un travail à réaliser. En colonne de droite, vous indiquerez la ou les commandes utilisées ainsi que leurs options et arguments, vous indiquerez aussi leur résultat.

| | Travail | Aide | Réponse |
|---|---|---|---|
| 1.1 | Affichez le nom de votre répertoire de travail (c'est aussi votre répertoire d'accueil si vous venez de vous connecter). | | - |
| 1.2 | Listez le contenu de votre répertoire d'accueil (listing long : option `-l`). Listez les fichiers cachés (idem). | `ls` | -<br>- |
| 1.3 | Déplacez-vous dans le répertoire `/etc` Vérifiez que vous y êtes bien. | `cd` | -<br>- |
| 1.4 | Revenez dans votre répertoire de travail en utilisant la commande la plus simple possible (sans aucun argument). Vérifiez que le changement est correct. | | -<br>- |
| 1.5 | Créez le répertoire Unix et déplacez-vous dedans. Vérifiez le déplacement. | `mkdir` | -<br>-<br>- |
| 1.6 | Copiez dans ce répertoire le fichier `/etc/passwd`. Vérifiez que la copie est correcte (avec arguments donnant au moins la taille des fichiers). | `cp` | -<br>- |
| 1.7 | Renommez cette copie en lui donnant le nom `test`. Vérifiez... | `mv` | -<br>- |
| 1.8 | Déplacez-vous dans le répertoire `/usr`. Vérifiez que vous y êtes. | | -<br>- |
| 1.9 | Depuis cet endroit, listez le contenu de votre répertoire Unix en utilisant le raccourci donné par le caractère `~` (sorte de constante contenant le nom du répertoire d'accueil [1]) | `ls` | - |

1. Le caractère `~` désigne mon répertoire à moi, et `~dupont` celui de l'utilisateur *dupont*.

| | | | |
|---|---|---|---|
| 1.10 | Vous êtes toujours dans `/usr`, tentez d'y copier le fichier `test` que vous avez créé précédemment (réutilisez le raccourci `~` pour nommer votre fichier). Vous devez normalement rencontrer un problème... Lequel et pourquoi ? Pouvons-nous remédier à ce problème ? | `cp, ls`, *droits* | -<br><br><br>Commentaires : |
| 1.11 | Retournez dans votre répertoire Unix. Faites une copie du fichier `test` que vous nommerez `test1`. Vérifiez le résultat. | | -<br>-<br>- |
| 1.12 | Supprimez le fichier `test`. Vérifiez... | `rm, ls` | -<br>- |
| 1.13 | Retirez le droit de lecture à tout autre que vous sur le fichier `test1` en utilisant la notation symbolique. Vérifiez... | `chmod` | -<br>- |
| 1.14 | Rajoutez le droit d'écriture pour le groupe et ceci en notation octale (les droits devront donc être : `rw-rw--`). Vérifiez | *idem* | -<br>- |
| 1.15[2] | Quelle est la valeur de votre `umask` ? Quels droits auront vos fichiers par défaut (les fichiers non exécutables d'abord, les fichiers issus d'une compilation/édition de liens ensuite) ? Créant des fichiers, des répertoires et vérifiez. | `umask`<br>`umask -S`<br>`man 2 umask`<br>`ls -l`<br>`stat` | |
| 1.16[2] | Quels sont vos quotas ? | `quota -vm`[3] | |
| 1.17[2] | Quel est l'espace disque accessible depuis votre machine ? | `df` | |
| 1.18[2] | Quel est l'espace disque occupé par tous vos fichiers ? (Attention, n'oubliez pas les fichiers cachés). | `du` | (Attention à l'unité d'affichage : option `-k`) |
| 1.19[2] | Si vous avez déjà au moins une fois lancé `mozilla` ou `firefox` vous devez avoir un répertoire `.cache/mozilla`. Sinon lancez-le, ce répertoire sera créé. Déplacez-vous dedans et affichez la taille du répertoire cache qui s'y trouve. À l'aide du menu préférences d'un de ces navigateurs, videz votre cache | `du` | (Attention à l'unité d'affichage) |

**Les commandes `chmod` et `umask`** manipulent traditionnellement les permissions Unix sous leur représentation en octal. L'octal, c'est la base 8, lorsque l'on compte de 0 à 7. Les nombres de 0 à 7 se codent sur 3 bits. Les permissions Unix `rwx` sont des drapeaux sur 3 bits. Voilà pourquoi on représente souvent les permissions Unix en octal. Cependant, pour plus de confort, les commandes `chmod` et `umask` modernes comprennent également la notation symbolique. Par exemple, pour ajouter les droits d'écriture au groupe pour un fichier, on peut faire `chmod g+w` *`fichier`*.

3. Vous pouvez également appeler le script école `/opt/bin/campux-quota.sh`, ou via le menu `Applications > Accessoires > Synthèse des quotas`

2. Question optionnelle : sachez que cette commande existe, mais on ne vous demande pas de la connaître par coeur. Vous ne serez pas évalué dessus.

| Notation symbolique | Notation octale |
|---|---|
| `-x -w- r-` | 124 |
| `rwx r-x r-x` | 755 |
| `rw- -- --` | 600 |

Notez que l'on positionne ici les 3 triplets `rwx`. On a donc une notation à trois nombres en octal. Il est également possible de positionner en même temps les drapeaux *suid bit*, *sgid bit*, et *sticky bit*. On a alors un quatrième nombre en octal qui est placé en tête des trois autres.

Autre chose : comprenez bien que le `umask` décrit un masque, c'est-à-dire les droits que l'on retire. C'est la négation binaire des permissions que les fichiers obtiendront par défaut.

## 1.1 Question bonus : un espace de partage

Voici un petit exercice qui devrait vous permettre de vérifier que vous avez bien assimilé le principe des permissions Unix.

L'idée est de mettre en place un espace vous permettant de partager facilement des fichiers avec vos collègues. (En effet, puisque les comptes Unix de l'école sont déjà en réseau, ce n'est pas la peine d'envoyer ses fichiers dans les nuages pour que le voisin puisse les récupérer...)

### 1.1.1 Mise en place

Voici la marche à suivre pour mettre en place cet espace de partage :

— Dans votre répertoire de travail (votre *home directory*), créez un sous-répertoire appelé (par exemple) : `partage`
— Ajouter les permissions de *lecture* et de *traversée* à ce sous-répertoire `partage`, pour le *groupe* et les *autres*. (Quelle commande tapez-vous ?)
— Vérifiez que votre répertoire de travail dispose bien également de la permission de *traversée* pour tout le monde.
— Dans ce sous-répertoire `partage`, créez deux sous-sous-répertoire appelés (par exemple) : `public` et `privé`
— Ajouter les permissions de *lecture* et de *traversée* au répertoire `public`, pour tout le monde.
— Ajoutez uniquement la permission de *traversée* au répertoire `privé`, pour tout le monde.

Vous disposez maintenant d'un espace `~/partage/public` accessible en lecture à toute personne disposant d'un compte Unix école, ainsi que d'un espace `~/partage/privé` accessible à toute personne disposant d'un compte Unix école mais de manière restreinte/cachée...

### 1.1.2 Utilisation

— Espace public

  Déposez un fichier dans votre espace `~/partage/public`. Assurez-vous que vous avez positionné les droits en lecture. Demandez à votre voisin de parcourir votre espace `public` et de vérifier qu'il peut bien accéder à votre fichier.

  Note : il faut que votre voisin connaisse au moins le nom de votre répertoire de travail, qui est typiquement votre nom de login. Par exemple pour visiter l'espace de partage public de Christophe Lohr, on peut faire `cd ~clohr/partage/public/`

— Espace privé

  Déposez un fichier dans votre espace `~/partage/privé`. Assurez-vous que vous avez positionné des droits en lecture. Demandez à votre voisin de parcourir votre espace `privé`. Normalement, il ne peut pas voir le contenu de ce répertoire (c.à.d. qu'il ne peut pas faire `ls`).

  Cependant, s'il connaît le nom de votre fichier, il peut y accéder : le nom du fichier devient en quelque sorte le mot de passe secret. À vous de choisir un nom de fichier que ne puisse pas être deviné facilement. (Pour choisir un nom un peu aléatoire, vous pouvez vous aider de commandes telles que `echo $RANDOM`, `mcookie`, `uuidgen`, etc.) Concrètement, seules les personnes à qui vous aurez communiqué le nom de votre fichier privé pourront y accéder. Faites l'essai.

# 2 Exercices sur l'environnement

| | Travail | Aide | Réponse |
|---|---|---|---|
| 2.1 | Quel est le nom de votre Shell | Déjà vu plus haut... | |

| | | | |
|---|---|---|---|
| 2.2 | Affichez les variables d'environnement de votre Shell. | `env` | - |
| 2.3 | À l'aide de la commande `echo`, affichez les valeurs des variables suivantes : `TERM, MAIL, PATH, USER, LOGNAME, SHELL`. Rappel : pour accéder à la valeur d'une variable, faire précéder son nom par le caractère `$`. Exemple : `echo $var` | | `echo $NOM_VAR`<br>-<br>-<br>-<br>-<br>- |
| 2.4 | Créez une variable *utilisateur* de nom `VAR1` et de valeur `abc` et une variable d'*environnement* de nom `VAR2` et de valeur `xyz`. Vérifiez la portée des variables `VAR1` et `VAR2` avec la commande `env` (vous pouvez affiner l'affichage en utilisant un tube de communication : `env \| grep VAR`) | `=`<br>`export` | -<br>- |
| 2.5 | Ouvrez deux autres fenêtres terminal, l'une depuis votre shell actuel avec la commande `xterm` ou `gnome-terminal`, l'autre depuis le menu graphique. Vérifiez si les shells associés à ces fenêtres connaissent vos variables déclarées dans la première fenêtre.<br>Pour plus de rigueur, et si vous êtes courageux, vous pouvez repérer la filiation des processus shell concernés avec la commande `pstree -h`[2]. Expliquez... | | - |
| 2.6(2) | La commande `mount` existe sur le système. Elle est utilisable par tout le monde pour lister les montages de systèmes de fichiers (il faut normalement être `root` pour monter un système). Avec la commande `which`, vérifiez que cette commande est accessible Sinon, cherchez avec `locate`. Indiquez quel est son chemin. | `which` | - |
| 2.7(2) | Avec la commande `whereis` cherchez si `mount` existe dans un répertoire standard. Indiquez le résultat. Avec la commande `ls -l`, listez chaque élément du résultat précédent. Que constatez-vous ? | `whereis`<br>`whatis` | -<br>- |
| 2.8(2) | Quels sont les systèmes de fichiers montés sur votre machine ? Indiquez les systèmes locaux ainsi que les systèmes montés en réseau. Pour ces derniers vous indiquerez le nom des serveurs. Sur quel disque se trouve le système de fichiers racine (le `/`) ? | `mount` | - |
| 2.9(2) | Quelle est l'imprimante par défaut de la machine sur laquelle vous êtes connectés actuellement ? À quelles imprimantes avez-vous accès ? | `lpstat` | - |
| 2.10(2) | Quel est votre numéro d'utilisateur ? | `id`<br>`$UID` | - |

# 3 Exercices sur les processus

| | Travail | Aide | Réponse |
|---|---|---|---|
| 3.1 | Testez la commande `ps`, d'abord sans argument puis avec les options `-u $LOGNAME`, et enfin en rajoutant l'option `-f` | | Quelles différences dans les affichages ? |
| 3.2(2) | Testez la commande `ps -ef` | | Que voit-on ? |
| 3.3 | Lancez une commande qui ne se termine pas tout de suite, par exemple `xeyes`. Ouvrez une autre fenêtre terminal et dans celle-ci recherchez le numéro du processus correspondant à la commande `xeyes`. En restant dans cette seconde fenêtre, utilisez le résultat de `ps` pour supprimer le processus `xeyes`. | `ps -ax`, `kill` | fenêtre 1<br>-<br>fenêtre 2<br>-<br>- |
| 3.4 | Relancez la commande `xeyes`. Votre terminal ne répond plus, les commandes que vous y entrez ne sont pas prises en compte. Stoppez alors le processus `xeyes` (faites `Ctrl-Z`) et passez-le en background (`bg`). Tapez la commande `jobs`. Que vous répond t-elle ? [4] | | |
| 3.5(2) | Lancez un nouveau `xeyes` suivi cette fois du caractère `&`. Que se passe-t-il ? Faites `jobs`. Que voyez-vous ? À l'aide de la commande `kill` et du résultat de `jobs`, supprimez le second `xeyes`.[4] | `jobs`, `kill` | -<br>-<br>- |
| 3.6 | Repassez le premier `xeyes` en premier plan (`fg`) et supprimez-le (`Ctrl-C`). | | -<br>- |
| 3.7 | À l'aide de la commande `top` ou `htop` relevez les caractéristiques de votre machine : taille mémoire, taille du swap, charge moyenne, par CPU... | `top`, `htop` | - |
| 3.8 | Avec la commande `tty` relevez le nom du terminal virtuel associé à votre fenêtre terminal. Avec `ps -aux` retrouvez le numéro de processus associé à votre fenêtre. Trouvez le numéro du shell associé à la fenêtre. | `tty, ps` | -<br>- |
| 3.9 | En utilisant le mécanisme des tubes de communication affichez la 23ème personne à s'être connectée dernièrement sur votre machine. L'historique des dernières connexions est accessible avec la commande : `last` Vous utiliserez les commandes `head` et `tail` pour filtrer la sortie standard de `last` et obtenir le résultat souhaité. [5] | | `last \| ..........` |
| 3.10 | Rediriger la sortie standard de la commande `ls` dans un fichier. Vous listerez le contenu de `/usr`. | | |

4. Le contrôle des tâches : `help jobs bg fg disown`

5. Si personne ne s'est connecté dernièrement, ça ne sera pas rigolo. Dans ce cas faites l'exercice en regardant l'historique des dernières commandes que vous avez tapées (commande `history`).

# 4 Exercice sur les commandes en réseau

*Cette série de questions est également optionnelle : vous ne serez pas évalués dessus, et elles ne sont pas requises pour mener à bien vos enseignements. Cependant, l'expérience montre qu'il est bon d'avoir quelques compétences sur le sujet (si vous souhaitez travailler sur les PC de l'école depuis chez vous, transférer vos fichiers, etc.).*

Remarque : traditionnellement on utilisait sous Unix les commandes `telnet` et `rlogin` pour les connexions à distance. Ces commandes sont maintenant à éviter car les mots de passe sont véhiculés en clair sur le réseau. Sous Linux, le serveur pour `rlogin` n'est d'ailleurs plus installé de manière standard sur les distributions courantes. Il faut préférer `ssh` que nous allons voir maintenant.

| | Travail | Réponse |
|---|---|---|
| 4.1 | Repérez le nom d'une autre machine que la vôtre dans la salle. Connectez-vous dessus par `ssh`. Vérifiez que vous êtes bien sur la machine en question grâce à la commande `uname -n`. | -<br>- |
| 4.2 | Dans quel répertoire êtes-vous arrivé avec `ssh` ? Listez son contenu. Est-il semblable au contenu de votre répertoire d'accueil sur la machine physique sur laquelle vous travaillez ? Pourquoi ? | -<br>- |
| 4.3 | Déconnectez-vous en fermant votre session `ssh` par `exit` (ou `logout`). Un répertoire caché `.ssh` est créé ; que contient-il ? | - |
| 4.4 | Après avoir regardé sa page de `man`, lancez la commande `ssh-keygen`, et contentez-vous de faire `Enter` aux diverses questions posées (vous avez confiance ou vous avez lu le `man` ?). Que s'est-il passé, et que contient le répertoire `.ssh` ? | -<br>- |
| 4.5 | Ajoutez votre clé publique ainsi générée dans votre liste de clés autorisées (`cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`)[6].<br>Reconnectez-vous à la machine voisine. Que se passe-t-il ? Pourquoi ? | - |
| 4.6 | Exécutez la commande `ssh -X` *`machine_voisine`*. Dans la fenêtre de connexion sur la machine distante lancez la commande `xterm`. Que constatez-vous sur la machine locale ? (Ce résultat est obtenu grâce à l'option `-X`.[7]) | - |
| 4.7 | Si vous disposez d'un compte différent sur un autre machine Unix (p.ex. au RésEl ?), copiez chez vous un fichier depuis cette autre machine avec la commande `scp`. Si non, copiez le fichier `/etc/hostname` de la machine voisine (cela a moins d'intérêt vu que vous avez des comptes partagés en réseau, c'est juste pédagogique...). | - |

**Remarques** (à prendre en compte en dehors des séances de TP)

— Sous environnement MS-Windows il existe également des implémentations des commandes `ssh` et `scp` avec le logiciel *putty*. Ce logiciel est disponible librement à l'adresse suivante : `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

6. Le script `ssh-copy-id` permet de copier sa clef publique sur un compte distant.
7. Cela peut être une bonne idée d'utiliser l'option `-C` en complément.

— Si vous voulez obtenir directement l'affichage des fenêtres graphiques Unix sur votre écran MS-Windows c'est un peu plus compliqué... Pour cela il y a plusieurs niveaux de solutions suivant que le serveur graphique et/ou les applications graphiques se trouvent en local sur votre PC MS-Windows ou bien sur une machine Unix distante. Voici donc trois cas de figure distincts :
  — VNC : un système open-source de bureau à distance. Cette solution est disponible sur les machines MS-Windows de l'école. Un client sur votre machine MS-Windows se connecte à une machine Unix sur laquelle se trouvent des applications graphiques et un serveur graphique qui envoie au client VNC les graphismes des applications par le protocole VNC (un genre de flux jpeg). En retour le client VNC envoie au serveur graphique les événements clavier et souris (excepté parfois le caractère `~`...).
  — Une implémentation d'un serveur X-Window sous MS-Windows. Par exemple `Xming` `http://www.straightrunning.com/XmingNotes/`. Dans cette solution, seul le serveur X est présent sur le PC MS-Windows ; et il faut se connecter sur une machine Unix pour y exécuter des applications graphiques. Celles-ci, grâce au protocole X (un peu gourmand en bande passante mais qui se compresse très bien) dialogueront avec le serveur X sur la machine MS-Windows. Les communications graphiques pouvant se faire également au travers d'un tunnel ssh (un petit tutoriel au RéSel `http://resel.fr/configuration/xming/`).
  — *Cygwin* `http://www.cygwin.com/`. C'est une implémentation sous MS-Windows d'un environnement POSIX qui peut ainsi exécuter toutes les commandes et applications classiques Unix ; et notamment un serveur X-Window, des applications graphiques, un shell, etc. Il faut télécharger un programme de «setup», le lancer et chercher l'item *X11* puis les items *xorg*. Avec Cygwin vous pouvez aussi obtenir les mêmes commandes ssh et scp que sous Unix.

Ceci décrit comment faire une connexion graphique à distance, reste la question de *«comment passer le firewall ?»*

— Tapez la commande suivante depuis votre machine locale (Linux ou Windows, équipée des commandes `ssh`), et dites ce qu'elle fait :

```
ssh -L 5901:srv-disi-vnc-04.priv.enst-bretagne.fr:5950 \
                                srv-disi-vnc-04.priv.telecom-bretagne.eu
```

— Toujours sur votre machine locale (équipé d'un client VNC), et sans fermer la connexion précédente, tapez la commande suivante et dites ce qu'elle fait : `vncviewer :1`

# 5 Le Shell de Bourne, initiation à la programmation

Le shell est nous offre un langage de programmation avec des structures de contrôle classiques (*if*, *while*, *for*, etc.). Il permet l'utilisation de variables, il possède des commandes qui lui sont propres (commandes internes). Les commandes Unix deviennent de véritables fonctions du shell. Ainsi nous pouvons écrire des fichiers exécutables qui enchaînent des commandes ou des programmes, ces fichiers sont en quelque sorte des commandes de plus haut niveau.

Dans un certain nombre d'établissements, pour des raisons historiques (typiquement un héritage de Sun Solaris), le shell par défaut proposé aux utilisateurs est le shell `tcsh` (c'est encore un peu le cas à Télécom Bretagne). Pour connaître votre shell faites par exemple `echo $0`.)

Pour différentes raisons [8] nous ne nous attarderons pas sur l'apprentissage de ce shell. Aussi, si vous êtes sous un shell `tcsh`, il serait préférable que vous travailliez directement en Bourne Shell, sous `bash` par exemple. Vous disposez de deux façons d'obtenir `bash` :

— De manière ponctuelle. Vous pouvez vous contenter de lancer la commande `bash` dans votre shell courant. Le prompt de `bash` a été paramétré par les administrateurs système pour ressembler à celui de `tcsh`. C'est un peu troublant (vous pourrez le changer en modifiant la variable d'environnement `PS1`), mais vous avez bien un shell `bash`.

— De manière permanente (mais non nécessairement définitive, vous pourrez rechanger pour `tcsh` si le cœur vous en dit). La commande `ypchsh` (lisez le `man`) vous permet de changer votre shell par défaut par l'un de ceux listés dans le fichier `/etc/shell` (du serveur NIS, mais c'est globalement le même que celui de la machine locale). Notez qu'il y aura un délai le temps que se propagent les pages NIS depuis le serveur et que votre cache local se mette à jours (si vous êtes pressés, connectez-vous sur une autre machine qui ne vous a pas vu depuis longtemps).

## 5.1 Les variables

— Créez la variable `var` et donnez-lui la valeur `10`. Vérifiez avec `echo $var` que la variable existe et possède la valeur indiquée.

8. `http://www.grymoire.com/Unix/CshTop10.txt`

— Créez la variable `VAR` et fixez-lui la valeur `VAL1`.
— Ajoutez au contenu de `VAR` le contenu `VAL2` en séparant les deux valeurs par le caractère « `:` ». Cela revient à concaténer `VAL1` avec `:VAL2`. On doit obtenir le résultat `VAL1:VAL2` dans `VAR`.

## 5.2 Les paramètres positionnels

— À l'aide d'un éditeur, créez le fichier `exo52` et écrivez dedans les lignes suivantes :
```
#!/bin/sh
echo $1 $2 $4 $3 $0
echo "Nombre de paramètres: $x"
echo "Numéro du processus: $y"
```
Vous remplacerez les *`$x`* et *`$y`* ci-dessus par les expressions convenables.
— Sauvegardez et rendez le fichier exécutable avec la commande `chmod`.
— Exécutez ce script de la manière suivante : `./exo52 un deux trois quatre cinq`
— Que constatez-vous ? Expliquez.
— En fin de script, ajoutez les lignes qui suivent, ré-exécutez et expliquez la commande interne `shift` :
```
shift
echo $1 $2 $4 $3 $0
```
— En fin de script, ajoutez les lignes qui suivent, ré-exécutez et expliquez la commande interne `set` :
```
set assez dit la baleine
echo $1 $2 $4 $3 $0
```

## 5.3 Les paramètres positionnels (suite)

— Au prompt de votre Shell (`bash`!), tapez les commandes suivantes :
```
date
d=`date`
```
(attention, ce sont des accents graves, des apostrophes inversées ou backquotes)
```
set $d
echo $1 $3
echo $d
```
— Attendez quelques instants (au moins une minute) et refaites `echo $d`.
— En déduire le rôle des accents graves et le rôle de la commande `set`.

## 5.4 Les commandes UNIX sont des fonctions du Shell

Ce sont des fonctions au sens où elles rendent une valeur qui peut être utilisée et testée grâce aux structures de contrôle du Shell comme `if`. La valeur rendue est stockée dans la variable « `?` ». Elle est donc accessible par l'expression `$?`.

Essayez les commandes suivantes, et dites quelle est la valeur rendue :
— `grep root /etc/passwd`
`echo $?`
— `grep truc /etc/passwd`
`echo $?`
— `grep root /etc/pwd`
`echo $?`
— `touch truc` *(création d'un fichier vide de nom* *`truc`*`)`
`rm truc`
`echo $?`
— `rm truc`
`echo $?`
— `rm  clohr/.cshrc`
`echo $?`

Vérifiez pour chacune de ces commandes si les valeurs rendues sont conformes à ce qu'annonce le manuel de référence dans les sections `EXIT STATUS` (faites `man grep`, `man rm`).

## 5.5 Les commandes Unix sont des fonctions du Shell (suite)

— Écrivez ceci dans un fichier `exo55` :
```
#!/bin/sh
grep $1 /etc/passwd > /dev/null
if test $? -eq 0
```

```
then
        echo "l'utilisateur $1 existe sur cette machine"
else
        echo "$1 n'est pas un utilisateur enregistré sur cette machine"
fi
```

— Rendez ce fichier exécutable (`chmod`). Et exécutez-le avec divers arguments (`root`, `truc`,...)

Attention, les utilisateurs des machines du réseau de l'école sont gérés par un service centralisé appelé NIS (Network Information Service) ou encore *Yellow Pages*. Vous trouverez donc curieux que la commande ci-dessus vous indique que vous n'êtes pas enregistrés (si vous cherchez votre nom de login). Remplacez alors la première ligne utile (`grep`...) par `ypmatch $1 passwd`).

## 5.6 Structures de contrôle `if` et `case`

La commande `date` vous donne normalement la date en français. Écrivez un script (`exo56`) qui vous l'affiche en anglais. (Sans jouer avec la `locale ;-)`) Vous indiquerez en particulier l'heure au format suivant : `it's 5 past 15` ou `it's 20 to 16` selon que les minutes seront inférieures ou supérieures à 30.

Si votre environnement est tel que la commande date affiche son résultat en anglais alors faites un programme de traduction anglais vers français.

Les commandes à utiliser (éventuellement) sont : `date` pour obtenir la date, `set` pour fractionner la réponse de date en mots (récupérés dans `$1`, `$2`, etc.) [Note : on peut également utiliser `read` pour ça.] La variable `IFS` contient les séparateurs permettant à `set` de fractionner les chaînes de caractères qu'on lui passe en paramètre. Pour fractionner l'heure il faudra rajouter le caractère « `:` » dans `IFS` (ou positionner `IFS` à la valeur `":"`).

La principale structure de contrôle du Shell à utiliser est `case`.

## 5.7 Scripts interactifs

Créez un fichier `exo57` qui intègre une séquence interactive qui pourrait être la suivante :

```
$ exo57
 Veuillez entrer votre nom et votre prénom: nom prenom
 Bienvenue prenom nom
$ _
```

Ce qui est souligné ci-dessus est entré par l'utilisateur. Ce qui est en italique est le résultat émis par votre programme. Veuillez noter l'inversion de l'affichage, le nom est affiché après le prénom alors qu'ils ont été entrés dans l'ordre inverse. Cette inversion est bien entendu le fait du programme, ce n'est pas *magique*.

Commandes à utiliser : `read`, `set`, `echo`.

Un plus serait de vérifier que deux mots ont bien été rentrés par l'utilisateur et dans le cas contraire reprendre la séquence en son début (utilisation de `test` et de la structure de contrôle `while`).

Nota : si on veut supprimer l'écho de chaque caractère renvoyé par le terminal (par exemple lorsqu'il y a un mot de passe à entrer) il faut utiliser la commande `stty -echo`. La commande `stty echo` rétablira l'écho.

## 5.8 Programmez en shell au standard Unix

Nous avons vu aux 5.4 et 5.5 que les commandes Unix rendaient une valeur lors de leur terminaison et que cette valeur était stockée dans la variable « `?` » du Shell. Dans cet exercice nous allons voir comment écrire un script qui se comporte de la même façon.

Reprenez le fichier de la question 5.5 et complétez-le avec la commande `exit` à laquelle vous fournirez en paramètre une valeur correspondant au résultat, à savoir : *utilisateur trouvé* → résultat VRAI, valeur rendue `0`, *utilisateur non trouvé* → résulta FAUX, valeur rendue différente de `0`.

Testez différentes valeurs de retour. Quelle est la valeur maximum (autour de 256...) ?

## 5.9 Comprendre les mécanismes des processus et ce qu'est une commande interne du Shell

À l'aide d'un éditeur créez le fichier `exo59` et écrivez dedans ceci : `cd /usr; pwd`

— Rendez le fichier exécutable et exécutez-le.

— Faites alors "à la main" `pwd` et notez ce que cette commande affiche, vous constatez que vous n'êtes pas dans `/usr` alors que votre commande `exo59` semble vous y avoir conduit. Expliquez pourquoi vous n'êtes pas restés dans `/usr`.

— Exécutez ce fichier à la manière d'une commande interne du Shell (en `sh` ou `bash` vous faites `.<`*espace*`>nom_fichier`, en C-Shell vous faites `source nom_fichier`) et vérifiez son résultat avec `pwd`. Constat ?
— Qu'en déduisez-vous sur la nature de cd en tant que commande ?

# 6 Extra bonus

Bien vivre son shell au quotidien... Parcequ'on le vaut bien !

## 6.1 `ls`

1. Comment afficher tous les fichiers du répertoire `/etc` sachant que l'affichage doit se faire avec les contraintes suivantes :

   (a) on doit afficher des informations sur chaque fichier

   (b) les fichiers les plus récemment modifiés doivent être affichés à la fin

   (Pour cela, utilisez la commande Unix `man` qui vous permet d'obtenir la documentation sur n'importe quelle commande du système.)

2. Allez sous le répertoire racine du système. Exécutez la commande `ls -F`. Que fait-elle selon vous ? Observez l'affichage de `/lib`. Qu'est-ce que cela signifie ? Comment peut-on en savoir plus ?

3. Affichez tous les fichiers du répertoire `/usr/bin` ayant les lettres `l` ou `b` dans leur nom de même que les fichiers dont le nom se compose de 4 lettres, et tout ça en une seule commande.

## 6.2 `cat more less head tail`

1. Affichez le contenu du fichier `/var/log/dmesg` en une seule fois puis page-à-page. Affichez seulement les 5 dernières lignes. Affichez seulement les 5 premières lignes.

2. Lorsque vous êtes en affichage page-à-page, rechercher la chaîne de caractère `PCI`

### 6.3 `cp rm mv`

1. Créez les répertoires `JAVA, BIN, DIVERS`. À l'intérieur du répertoire `DIVERS`, créez les fichiers `youpla`, `youplala` et `on_est_les_champions`. En une seule commande, faites une copie du répertoire `DIVERS` et son contenu sous votre home-directory. Créez le fichier `ca_assure` dans le répertoire `JAVA`. Changez ensuite son nom en `ca_assure.un_max`. Effacez les répertoires `JAVA`, `BIN` et `DIVERS`.

2. Créez un fichier dont le nom est le caractère `-`. Essayez maintenant de le détruire. (Ça vous arrivera !!)

### 6.4 `find which locate whereis`

1. Donnez le chemin absolu du fichier `cat`. Donnez son chemin de façon relative à votre home-directory.

2. Recherchez dans l'arborescence des fichiers, à partir du répertoire `/etc` l'ensemble des fichiers qui ont été modifiés il y a 4 jours ; recherchez dans l'arborescence des fichiers, à partir du répertoire `/tmp` l'ensemble des fichiers dont les droits sont `rw-r-r-`.

### 6.5 `chmod`

1. Allez dans le répertoire `/usr/bin` et créez le fichier `beurk`. Quelle est la réaction du système ? Allez dans le répertoire `/tmp` et effectuez la même opération. Quelle est la réaction du système. Commentez.

2. Créez un répertoire `C` dans votre home-directory et interdisez à tous les utilisateurs (à part vous) d'entrer dans ce répertoire. Vérifiez en demandant à un autre utilisateur de tenter d'entrer dans votre répertoire. Enlevez le droit de lecture sur ce répertoire pour vous-même. Tentez à présent de pénétrer dans ce répertoire et d'en lire le contenu (vous effectuerez toutes ces opérations en positionnant les droits un-à-un, mais aussi en les positionnant de façon globale à l'aide de la notation octale).

3. Le mécanisme du *sticky bit*. Créez un répertoire `oula` dans votre compte. Donnez tous les droits à tout le monde sur ce répertoire. Créez ensuite dans ce répertoire un fichier `perso` sur lequel vous ne donnerez

aucun droit aux utilisateurs (à part vous bien sûr). Demandez à un autre utilisateur de détruire ce fichier. Y parvient-il ? Pourquoi ? Faites maintenant en sorte qu'il ne puisse plus détruire le fichier sans lui enlever le droit d'écrire dans le répertoire.

4. Question subsidiaire : le mécanisme du *suid bit*. La commande `passwd` vous permet de changer votre mot de passe (local). Le fichier où sont stockés les mots de passe en local est le fichier `/etc/shadow`. Quels sont les droits d'accès de ce fichier ? Comment se fait-il qu'un utilisateur puisse changer son mot de passe ?

## 6.6 *pipe* `grep wc tr cut`

1. Comptez le nombre de lignes (puis de mots, puis de caractères) du fichier `/etc/X11/xorg.conf`.
2. La commande `ypcat passwd` affiche les informations utilisateur provenant des pages jaunes en réseau NIS. Affichez, puis comptez, les d'utilisateurs dont le shell de login est `/bin/bash`.
3. N'affichez que le login de ces utilisateurs
4. Affichez leur login en majuscules

## 6.7 `& ; <Ctrl>-C <Ctrl>-Z jobs bg fg kill ps`

1. Quelle est la différence entre ces deux lignes de commande (et faites un `<Ctrl>-C` pour voir) : `xclock & xeyes` puis `xclock ; xeyes`

2. Réessayez `xclock ; xeyes` et faites un <`Ctrl`>-`Z`. Que se passe-t-il ?

3. Quels sont les `jobs` qui sont en activité et quels sont ceux qui sont stoppés ?

4. Réactivez le ou les jobs stoppés de manière à ce qu'ils s'exécutent en tâche de fond.

5. Vérifiez bien que tous vos jobs sont actifs. Choisissez-en un et placez-le en avant-plan. Puis tuez-le avec un <`Ctrl`>-`C`.

6. Repérez le numéro de processus de chacun des jobs encore actifs (attention, un numéro de job et un numéro de processus ce n'est pas la même chose).

7. Tuez les jobs encore actifs en indiquant soit le numéro de job, soit le numéro de processus.

## 6.8 `alias unalias`

1. Consulter la liste des alias qui sont d'ores et déjà définis pour votre compte. Détruisez l'alias `cp` et créez les alias suivants :
   (a) `lf` est un alias pour `ls -F`
   (b) `truc` est un alias qui permet d'afficher la date et la chaîne de caractères `"pffff"`
   (c) `motsdepasse` est un alias vers le fichier `/etc/passwd`

   Vérifiez que ces alias fonctionnent.

2. Ouvrez un autre terminal. Les alias définis précédemment sont-ils définis dans ce terminal ? Faites en sorte que l'alias `lf` soit défini dans tout nouveau shell créé.

## 6.9 Les variables d'environnement

1. Que fait la commande interne `env` ?
2. Quelle est la commande qui vous permet de connaître le chemin du programme `cat`. Quelle variable d'environnement utilise-t-elle ?
3. Positionnez la variable d'environnement `PATH` à `/usr/bin`. Essayez d'exécuter la commande `useradd`. Quelle est la réponse du système ? Cherchez où se trouve cette commande et modifiez la variable d'environnement `PATH` de telle façon que vous puissiez exécuter la commande `useradd`.
4. Quelle est la variable d'environnement nécessaire au bon fonctionnement de la commande `man` ?
5. Créez une variable `i` en lui affectant une valeur quelconque Affichez la valeur de cette variable.
6. À partir du shell dans lequel vous avez créé cette variable `i`, créez un autre terminal. La variable `i` est-elle connue dans ce terminal ? Comment faire pour qu'elle soit connue ?
7. Définissez une variable dont la valeur est une chaîne de caractères contenant le caractère `$`. Affichez ensuite sa valeur. Que constatez-vous ? Comment faire pour résoudre ce problème ?

8. Définissez 2 variables `i` et `j` en leur affectant des chaînes de caractères quelconques. Définissez une variable `k` dont la valeur est la concaténation de `i` et `j`.

9. Définissez les variables `m` et `n` de valeur `2` et `3`. Exécutez une commande qui affiche la somme de ces 2 variables (pensez à utiliser la commande Unix `expr` ou aux fonctionnalités internes du shell).

### Décompresser une archive

| Extension | Commande |
|---|---|
| .tar.gz | tar zxvf <nom_du_programme>.tar.gz |
| .tar.bz2 | tar jxvf <nom_du_programme>.tar.bz2 |
| .gz | gunzip <nom_du_programme>.gz |
| .bz2 | bunzip2 <nom_du_programme>.bz2 |
| .zip | unzip <nom_du_programme>.zip |

## Je n'ai plus de souris

| Raccourci | Fonction dans KDE |
|---|---|
| Alt + F12 | Démarre l'émulation du déplacement de la souris avec les touches du pavé numérique |
| 2, 4, 6, et 8, (ou Bas, Gauche, Haut et Droite) | Déplace le curseur de la souris dans les quatre directions |
| Entrée (ou Espace) | Clic gauche et sortie du mode émulation |

Autre possibilité, plus riche mais plus contraignante, lancer le programme `xorgcfg` avec Alt + F2.

| Touches du pavé numérique | Fonction dans xorgcfg |
|---|---|
| 1 2 3 4 6 7 8 9 | Déplace le curseur de la souris dans les huit directions |
| / (ou * ou -) | Sélection du bouton gauche (ou milieu ou droit) |
| 5 (ou +) | Clic (ou double clic) avec le bouton sélectionné |
| 0 ou . | Verouillage (ou relâchement) de l'appui du bouton sélectionné |

Par ailleurs dans une boîte de dialogue on peut se déplacer entre les différentes zones ou boutons avec Tab ou Maj + Tab, dans les zones ou listes avec Haut, Bas, Gauche ou Droite, et cliquer avec Espace. Les listes déroulantes s'ouvrent et se ferment souvent avec F4.

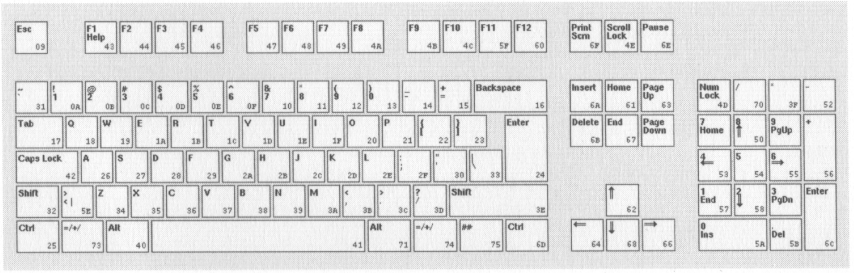
## Touches magiques

Si plus rien ne marche vous pouvez redémarrer votre machine avec les raccourcis suivants (attention il faut utiliser les trois raccourcis successivement).

| Raccourci | Fonction |
|---|---|
| Alt + ImprEcran + S | Vide le cache disque |
| Alt + ImprEcran + U | Remonte le(s) disque(s) en lecture seule |
| Alt + ImprEcran + B | Redémarre la machine |

Toutes les notions contenues dans cet aide-mémoire sont présentées de façon détaillée dans le livre, et sont donc décrites ici sommairement.

## Clavier américain






# Aide-mémoire du débutant sous Linux

## Raccourcis généraux

| Raccourci | Fonction |
|---|---|
| Ctrl + Alt + Plus (du pavé numérique) | Passe à la résolution d'écran supérieure |
| Ctrl + Alt + Moins (du pavé numérique) | Passe à la résolution d'écran inférieure |
| Ctrl + Alt + Retour | Redémarre le serveur X |
| Ctrl + Alt + Suppr | Redémarre l'ordinateur |
| Ctrl + Alt + F1 à F6 | Accès aux consoles texte |
| Ctrl + Alt + F7 (F8 etc.) | Accès aux consoles graphiques |
| Maj + Précédent (ou Suivant) | Remonte (ou redescend) dans l'affichage d'une console |

## KDE

| Raccourci | Fonction |
|---|---|
| F1 | Ouvre le centre d'aide de KDE |
| Ctrl + F1 à F12 | Accès direct à un bureau (s'il existe) |
| Ctrl (+ Maj) + Tab | Déplacement circulaire entre les bureaux (en arrière) |
| Alt (+ Maj) + Tab | Déplacement circulaire entre les fenêtres du bureau courant (en arrière) |
| Alt + F2 | Ouvre la mini-ligne de commande |
| Alt + F3 | Menu des opérations de fenêtre |
| Alt + F4 | Ferme une fenêtre |
| Alt + F5 ou clic milieu sur le fond du bureau | Affiche la liste des fenêtres |
| Ctrl + Alt + Echap | Permet de tuer une application avec un clic gauche (Échap pour revenir au curseur normal) |
| Ctrl (+ Maj) + A | Tout (dé)sélectionner |
| Ctrl + O (ou W ou S) | Ouvrir (ou fermer ou enregistrer) un fichier |
| F4 | Ouvre ou referme une liste déroulante |
| Haut ou Bas | Navigue dans une liste déroulante même non ouverte |
| Ctrl (ou Maj) pendant un glisser-déposer | Copie (ou déplace) le(s) fichier(s) sans demander confirmation |
| Ctrl + Maj pendant un glisser-déposer | Crée un lien sans demander confirmation |
| Ctrl + Alt + L | Efface l'écran et le verrouille |
| Alt + Bouton gauche (ou droit) de la souris | Déplace (ou redimensionne) la fenêtre |
| Ctrl + Molette de la souris | Zoom avant ou arrière dans le document ou la page web |
| Alt + Molette de la souris | Défilement horizontal des ascenceurs |
| Clic droit sur le fond du bureau | Menu contextuel de création/configuration du bureau/déconnexion |

## Édition

| Raccourci | Fonction |
|---|---|
| Ctrl + C | Copier |
| Ctrl + V | Coller |
| Ctrl + X | Couper |
| Ctrl + Droite (ou Gauche) | Se déplacer d'un mot vers la droite (ou vers la gauche) |
| Ctrl + Maj + Droite (ou Gauche) | Sélectionner un mot vers la droite (ou vers la gauche) |
| Ctrl + Supr (ou Efface) | Supprimer le mot de droite (ou de gauche) * |
| Ctrl (+ Maj) + Z | Annuler (rétablir) |
| Ctrl + F | Rechercher |
| (Maj +) F3 | Rechercher le suivant en avant (en arrière) |
| Ctrl + R | Remplacer |
| Mal + Début (ou Fin) | Sélectionne du début (ou de la fin) de la ligne jusqu'au curseur |
| Ctrl + Début (ou Fin) | Aller au début (ou à la fin) du document |

* Dans OpenOffice.org 1.1 cette combinaison de touches efface la phrase complète.

## Navigation

| Raccourci | Fonction |
|---|---|
| Alt + Gauche (ou Droite) | Affiche la page précédente (ou suivante) de l'historique de la navigation |
| F5 | Recharge la page |
| Maj + Clic gauche | Enregistre le lien sous... |
| Ctrl + Plus (ou Moins) du pavé numérique | Zoom avant (ou arrière) dans une page |

# Le Shell – Ergonomie

### Raccourcis clavier

| Raccourci | Fonction |
|---|---|
| Tab | Active la saisie assistée |
| Ctrl + A (ou E) | Déplace vers le début (ou la fin) de la ligne de commande |
| Ctrl (ou Alt) + B (ou F) | Déplace d'un caractère (ou d'un mot) en arrière (ou en avant) |
| Ctrl (ou Alt) + D | Efface le caractère (ou le mot) suivant le curseur |
| Ctrl + K (ou Y) | Efface la fin de la ligne (ou rapelle le dernier effacement) |
| Ctrl (ou Alt) + T | Inverse les caractères (ou les mots) précédant le curseur puis déplace le curseur après le dernier caractère (ou mot) inversé |
| Ctrl + C | Stoppe la commande en cours |
| Ctrl + D | Ferme le terminal en cours |
| Ctrl + L | Nettoie l'écran |

### Utilisation de l'historique

| Nom | Fonction |
|---|---|
| history \| grep `texte1` | Affiche toutes les commandes commençant par `texte1` et leur numéro d'historique |
| ! `nombre` | Exécute la commande ayant le numéro `nombre` dans l'historique |
| `texte`!! | Exécute une commande composée de `texte` concaténé avec la dernière commande tapée |
| `texte1`!`texte2` | Exécute une commande composée de `texte1` concaténé avec la dernière commande tapée commençant par `texte2` |

# Le Shell – Commandes

### Options et notations générales

| Nom | Fonction |
|---|---|
| . (un point) .. (deux points) ou ~ | Répertoire courant, répertoire parent ou répertoire utilisateur |
| Caractères jokers ? et * | ? remplace un caractère et * un nombre indéfini de caractères dans le nom d'un fichier (* seul signifie tous les fichiers) |
| commande - -help | Affiche une aide sommaire sur la commande |
| commande - -version (ou -v) | Affiche le numéro de version de la commande |
| commande & | Exécute la commande en arrière-plan et rend la main |

### Commandes

Les différentes options (précédées d'un tiret) peuvent être chainées. Ex : `ls -al`, `rm -rf`.
Un fichier peut être désigné par son nom seul s'il est dans le répertoire courant, sinon par son nom complet comprenant aussi le chemin d'accès relatif (à partir du répertoire courant) ou absolu (à partir de la racine /).
Exemple de chemin absolu : `/home/perrine/repertoire/fichier` (autre syntaxe : `~/repertoire/fichier`) et de chemin relatif à partir de `/home/perrine` : `/repertoire/fichier`. Le nom d'un répertoire commence toujours par /.
On peut chaîner deux commandes avec le caractère | (tube) de la façon suivante : `commande1 | commande2`. La sortie de la `commande1` est utilisée comme entrée de la `commande2`. Le caractère | s'obtient avec Alt Gr + 6 (du pavé alphabétique).
On peut envoyer la sortie d'une commande dans un fichier texte ainsi : `commande>fichier.txt` pour créer ou écraser le fichier, et `commande>>fichier.txt` pour compléter le fichier existant.



### Fichiers et répertoires

| Nom | Fonction |
|---|---|
| cd <repertoire><br>cd .. | Déplace vers le <repertoire><br>Déplace vers le répertoire parent |
| ls (<repertoire>) (\| more)<br>ls -a, ou ls -l | Liste les fichiers du répertoire courant (ou de <repertoire>) (page par page)<br>Liste aussi les fichiers cachés, ou de façon détaillée |
| cp <fichier1> <fichier2><br>cp <fichier> <repertoire><br>cp -R <repertoire1> <repertoire2> | Copie le <fichier1> sur le <fichier2><br>Copie le <fichier> dans le <repertoire><br>Copie <repertoire1> et tout ce qu'il contient dans <repertoire2> |
| mv <fichier1> <fichier2><br>mv <fichier> <repertoire> | Renomme <fichier1> en <fichier2><br>Déplace <fichier> du répertoire courant vers <repertoire> |
| rm (-f) <fichier><br>rm (-r) <repertoire> | Supprime <fichier> (sans demander confirmation)<br>Supprime <repertoire> (et tout ce qu'il contient) |
| mkdir <repertoire><br>rmdir <repertoire> | Crée le <repertoire><br>Supprime le <repertoire> (uniquement s'il est vide) |
| ln (-s) <cible> <lien> | Crée un lien (symbolique) vers <cible> nommé<lien> |
| chmod <droits> <fichier><br>(Lecture = 4, Écriture = 2, Exécution = 1) | Affecte de nouveaux droits au <fichier> |
| chown, chgrp <fichier> | Affecte un nouveau propriétaire, ou groupe au <fichier> |
| mount | Affiche les systèmes de fichiers montés |
| mount (ou umount) <peripherique><br>(ou <point_de_montage>) | Monte (ou démonte) le périphérique (de la forme `/dev/xxxx`) ou le point de montage (de la forme `/repertoire`) |

### Processus

| Nom | Fonction |
|---|---|
| ps -A (PS aux) (\| grep <nom>) | Affiche la liste des processus et leur PID (affichage plus complet) (uniquement le programme ou la commande <nom>) |
| kill (-9) PID | Tue proprement (ou sauvagement) le processus ayant le numéro PID |
| killall (-9) <nom> | Tue proprement (ou sauvagement) le(s) processus nommés <nom> |

### Recherche et information

| Nom | Fonction |
|---|---|
| find <repertoire> -critere<br>find <repertoire> -name "*chaine*" | Trouve un fichier correspondant au critère dans <repertoire><br>Trouve tous les fichiers de <repertoire> dont le nom contient chaine |
| grep <chaine> <*nom*> | Affiche les fichiers contenant <chaine> dont le nom contient <nom> |
| locate <fichier> | Affiche le(s) répertoire(s) contenant <fichier> |
| file <fichier> | Affiche le type de <fichier> |
| cat <fichier> (\| more) | Affiche le contenu de <fichier> (page par page) |
| which <commande> | Affiche le chemin du répertoire où se trouve <commande> |
| df (ou du) <repertoire> | Affiche le taux d'occupation des partitions montées et points de montage (ou de tous les sous-répertoires) de <repertoire> |
| pwd | Affiche le nom du répertoire courant |
| whoami | (Qui suis-je?) Affiche le nom de l'utilisateur courant |
| id <utilisateur> | Affiche les numéros de l'utilisateur et de ses groupes |
| free | Affiche l'occupation de la mémoire physique et du swap |
| top | Affiche la liste des tâches qui mobilisent le plus le processeur |
| tail (-25 ou -f) <fichier> | Affiche les 10 (ou 25, ou réactualise) dernières lignes de <fichier> |
| (/sbin/)ifconfig | Affiche les informations réseau (avec le préfixe si simple utilisateur) |
| uname | Affiche la version du noyau |
| halt, reboot | Stoppe ou redémarre la machine |