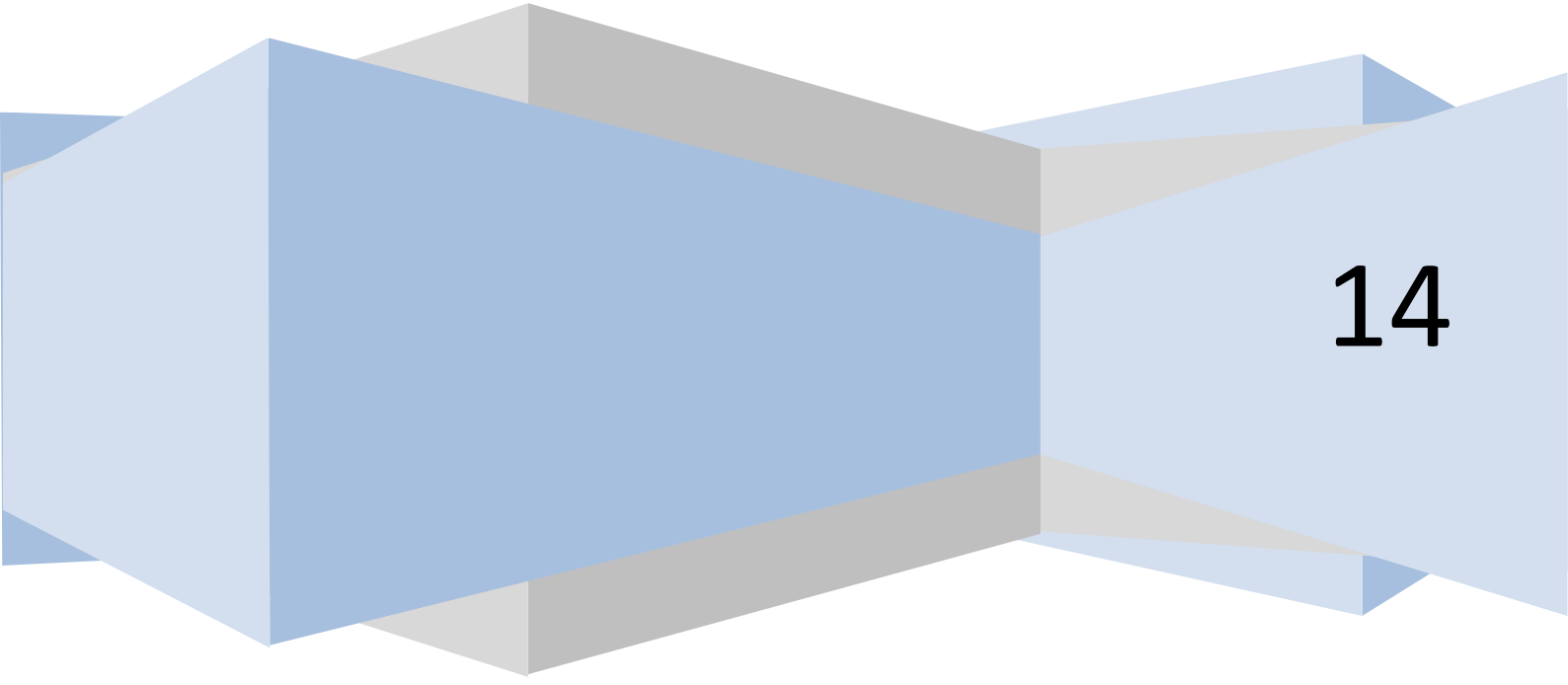
EFREI

TD Unix pour l'utilisateur

Partie 1

tpunix@efrei.fr



14

Les réponses aux exercices devront être écrites dans un fichier texte - pas de document Open Office ou de document Word, un simple fichier texte - dont le nom sera composé de :

- La lettre désignant votre groupe
- Les noms des membres de votre binôme
- La date

Voici un exemple de nom de fichier :
`B_edward_alphonse_19052012.txt`
N'hésitez pas à illustrer votre compte rendu avec des copiés/collés venant directement de votre terminal (pas des images, seulement le texte).
N'oubliez pas, pour chaque question, de préciser son numéro.

A la fin du TP, il conviendra de m'envoyer ce fichier par mail à l'adresse *tpunix@efrei.fr*, en précisant le nom du fichier dans le champ 'Sujet' du mail.

Table des matières

1 Présentation d'Unix

1.1 Un peu d'histoire

1.1.1 Les origines

En 1969, dans le Bell Laboratories (AT&T), Ken Thompson crée un système d'exploitation entièrement écrit en assembleur : **Unics**. Cet OS se veut être un remplaçant maison de **Multics** (MULTIplexed Information and Computing Service), abandonné en cours de route par AT&T, faute de moyen.

Rejoint en cours de route par Dennis Ritchie - qui est également très connu pour son livre co-écrit avec Brian Kernighan : *The C Programming Language* - , Ken Thompson, conscient des difficultés de maintenance que pose un système d'exploitation entièrement écrit en assembleur, décide, en 1971, de réécrire son produit, déjà appelé **Unix** à l'époque. Après quelques tergiversations, le langage **C**, mis au point par Ritchie, est retenu.

Ken Thompson et Dennis Ritchie présentent le premier article sur **Unix** au Symposium *Operating Systems Principles* à l'Université de Purdue en 1973. Un professeur de l'UCB (*University of California Berkeley*) très intéressé par le concept le fait déployer dans son université à des fins pédagogiques, par un étudiant du nom de Bill Joy.

Contraint par un décret de 1956 de ne vendre que du matériel téléphonique, AT&T accepte qu'**Unix** soit distribué presque librement avec ses sources.

Suite au passage à l'UCB de Ken Thompson en tant que professeur, le développement d'Unix est repris dans cette université pour aboutir en 1977 à la première **BSD** (*Berkeley Software Distribution*).

1.1.2 La diffusion

A partir de la fin des années 1970, il coexiste trois équipes de développeur travaillant sur **Unix** :

- Le laboratoire de recherche de Bell
- La branche commerciale de Bell qui développa **System III**, puis quatre éditions de **System V**
- L'UCB qui développa la **BSD** jusqu'en 1994

En 1979, la DARPA (*Defense Advanced Research Projects Agency*), intéressée par le système BSD, forme un *sterring committee*, et finance le développement des **BSD** jusqu'en 1983. Ce financement a permis le développement de BSD jusqu'à la version 4.2.

A ce moment là, **BSD** est la version d'**Unix** la plus populaire, notamment car elle implémente une couche TCP/IP, développée par Bill Joy, et un système de fichiers efficace, le FFS (*Berkeley Fast Filesystem*). Bill Joy deviendra par la suite un des fondateurs de Sun Microsystems, chez qui il occupera la position de directeur de la recherche jusqu'en 2003.

Le MIT (*Massachusetts Institute of Technology*) propose en 1984 la norme **X Window**, qui permet de définir une IHM graphique en surcouche à **Unix.**

1.1.3 Le début de l'Open Source

Jusqu'à **4.3BSD**, les sources étaient toujours distribuées aux utilisateurs finaux, qui pouvaient modifier le code à leur convenance. Malheureusement, AT&T décide à ce moment de faire payer des licences pour le code leur appartenant encore. **BSD** est alors nettoyé du code AT&T et est librement distribué, et en Juin 1989 est distribué la première **BSD** entièrement libre, alors nommée **Net/1**. La licence permet de modifier, de redistribuer et de vendre le code tant que les notices de copyright sont laissées intactes dans le code. Ce sont les prémisses de ce que deviendra le monde du Logiciel Libre que nous connaissons aujourd'hui. C'est suite à la distribution de **Net/2** qu'apparait la **386BSD**, dont découleront la **NetBSD** et la **FreeBSD**.

1.1.4 Linux

En 1985, un professeur américain habitant les Pays-Bas, Andrew S. Tanenbaum, publie un système d'exploitation minimal, nommé **Minix**, à des fins pédagogiques. En 1991, un étudiant finlandais, Linus

Torvalds, décide de se lancer dans la même aventure, à titre de hobby, pour offrir un système expérimental pour processeur x86.
Voici son post sur `comp.os.minix` :

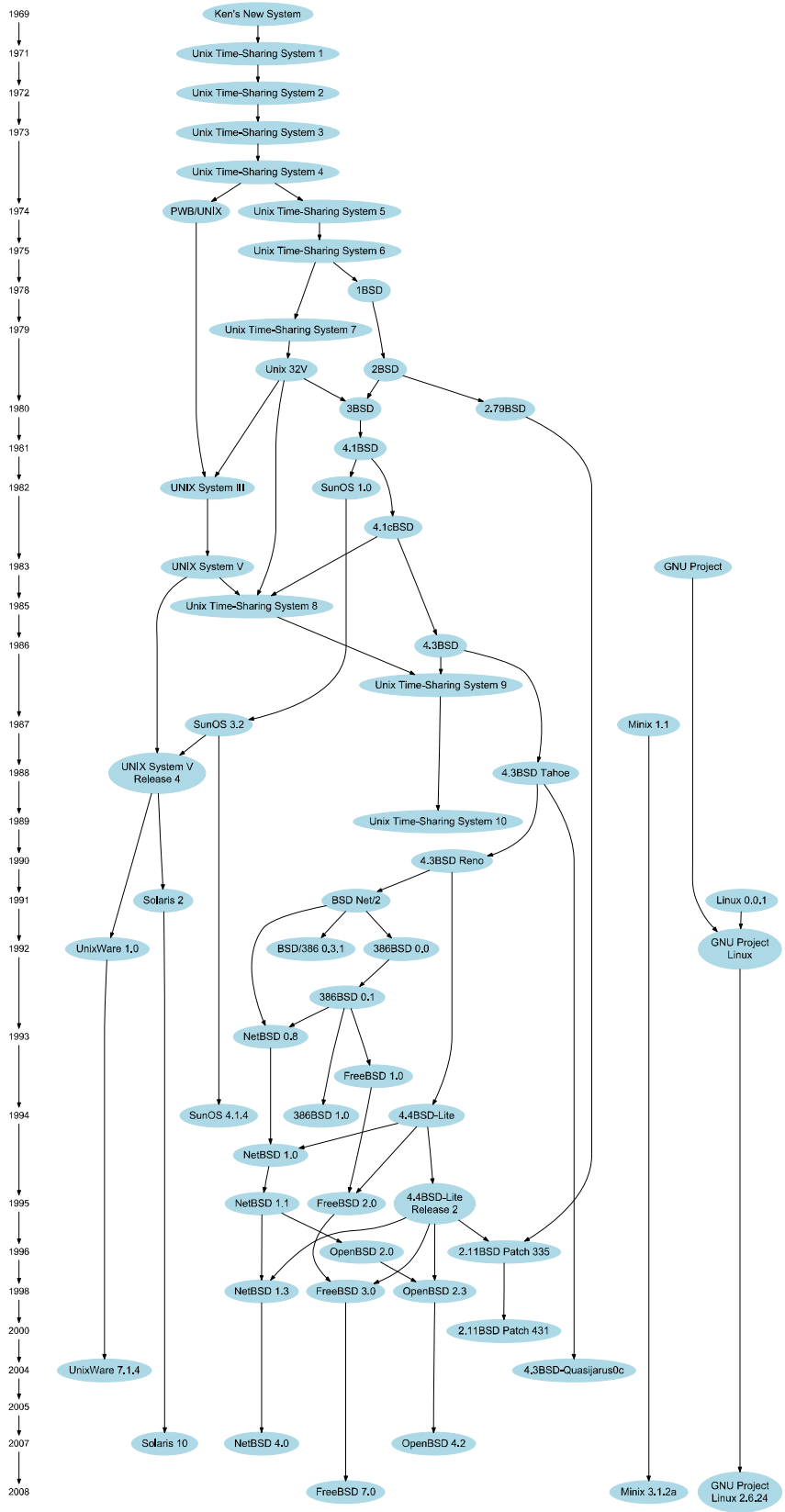
« Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. »

Linux n'est qu'un noyau, et a besoin d'utilitaires pour être un système d'exploitation complet et utilisables. Ces utilitaires proviennent en grande partie du projet GNU (*Gnu's Not Unix*).

1.1.5 GNU

Le projet GNU a été lancé en 1985 par Richard Stallman alors chercheur au laboratoire du MIT. Son objectif était, d'après ses propres mots, de « ramener l'esprit de coopération qui prévalait dans la communauté informatique dans les jours anciens ». Stallman avait pour but de créer un système d'exploitation libre, complet et gratuit. Le noyau du projet GNU, **Hurd**, est toujours en cours de développement, et c'est pourquoi les utilisateurs GNU utilisent le noyau **Linux**.

1.1.6 Un arbre généalogique des systèmes Unix



1.2 Qu'est ce qu'un système Unix

Le système Unix est multiutilisateur et multitâche, il permet donc à un ordinateur mono ou multi-processeurs d'exécuter apparemment simultanément plusieurs programmes dans des zones protégées appartenant chacune à un utilisateur.

Un système Unix repose sur quatre concepts : celui de fichier, de processus, d'IPC et de contrôle d'accès.

1.2.1 Fichiers

Dans un système Unix, tout est représenté sous forme de fichiers. C'est bien entendu vrai pour tout ce qui est binaires exécutables, documents, textes, configuration et autres, mais c'est également vrai pour les périphériques, qu'ils soient physiques (disque dur, port série, souris) ou logiques (partition d'un disque, pseudo terminal, ramdisk, socket…). Un exemple frappant de cette versatilité du fichier sous Unix est la possibilité « d'écouter son kernel » avec la commande `cat /boot/kernel > /dev/dsp` qui envoie le contenu du fichier */boot/kernel* sur le périphérique caractère qu'est la carte son.
A ce sujet, il existe d'ailleurs deux types de périphériques Unix : les périphériques blocs (disques durs, lecteurs de bandes magnétiques) et les périphériques caractères (port série, port USB, horloge RTC…).
Tous les fichiers d'un système Unix occupe une référence dans une arborescence ordonnée, dont la racine, notée / et nommée *root,* est le point de départ de tous les chemins vers tous les fichiers. Les périphériques de masse sont donc accessibles via cette arborescence, à l'opposé d'un système **Microsoft Windows** où les périphériques de masses sont la ou les racines du système de fichier.
Voici un exemple de système de fichiers **Linux**. Il s'agit de la machine **pommard.etudiants.efrei.fr** :

```
/
|-- bin
|-- boot
|-- cdrom
|-- dev
|   |-- bus
|   |-- disk
|   |-- fd -> /proc/self/fd
|   |-- input
|   |-- loop
|   |-- mapper
|   |-- net
|   |-- pts
|   |-- shm
|   `-- snd
|-- etc
|-- floppy
|-- home
|-- home_win
|-- initrd
|-- lib
|   |-- alsa
|   |-- discover
|   |-- firmware
|   |-- i386-linux-gnu -> i486-linux-gnu
|   |-- i486-linux-gnu
|   |-- init
|   |-- iptables
|   |-- linux-sound-base
|   |-- lsb
|   |-- modules
|   |-- oss-compat
|   |-- security
|   |-- terminfo
|   |-- tls
|   `-- udev
|-- media
|-- mnt
|-- opt
|   |-- eda
|   |-- local
|   |-- oracle
|   `-- test
|-- proc
|-- sbin
|-- sys
|   |-- block
|   |-- bus
|   |-- class
|   |-- devices
```

```
|   |-- firmware
|   |-- fs
|   |-- kernel
|   |-- module
|   `-- power
|-- tmp
|-- usbdrive
|-- users
|   |-- batb
|   |-- es
|   |-- guest
|   |-- p2010
|   |-- p2011
|   |-- promo-2007
|   |-- promo-2008
|   |-- promo-2009
|   |-- promo-2010
|   |-- promo-2011
|   `-- promo-2012
|-- usr
|   |-- X11R6
|   |-- bin
|   |-- doc
|   |-- games
|   |-- include
|   |-- lib
|   |-- lib64
|   |-- local
|   |-- sbin
|   |-- share
|   |-- src
|   `-- tmp
`-- var
    |-- autofs
    |-- backups
    |-- cache
    |-- db
    |-- games
    |-- lib
    |-- local
    |-- lock
    |-- log
    |-- mail
    |-- opt
    |-- run
    |-- spool
    |-- tmp
    `-- www
```

En plus de la localisation du fichier (aussi nommée chemin du fichier), le système de fichiers contient un certain nombre d'informations relatives au fichier comme :

- Propriétaire
- Groupe
- Taille
- Date de dernière modification
- Date du dernier accès
- …

En fonction du type de système Unix, et de sa configuration, cette liste peut varier.
Voici un exemple de listage de ces informations :

```
vinzelle@pommard:~$ ls -l
total 61
-rw-r--r-- 1 vinzelle vacat  5186 2008-05-03 18:19 arborescence.txt
-rw------- 1 vinzelle vacat 53980 2008-04-08 14:38 cv.pdf
drwx------ 2 vinzelle vacat   512 2008-04-08 14:38 Mail
drwx------ 2 vinzelle vacat   512 1995-08-30 15:45 News
```

/bin	les fichiers exécutables (en binaire) (initialisation du système + commandes "essentielles")
/boot	le noyau vmlinuz et les fichiers de démarrage
/dev	répertoire de fichiers spéciaux, qui servent de canaux de communication avec les périphériques (disques, adaptateur réseau, cartes son etc...)
/etc	les fichiers de configuration du système et les principaux scripts de paramétrage
/etc/rc.d	scripts de démarrage du système
/etc/X11	scripts de configuration du serveur X
/etc/sysconfig	configuration des périphériques
/etc/cron	description des tâches périodiques à effectuer
/etc/skel	fichiers recopiés dans le rép. personnel d'un nouvel utilisateur
/home	la racine des répertoires personnels des utilisateurs
/lib	les bibliothèques et les modules du noyau
/mnt	la racine des points de montage des systèmes de fichiers périphériques ou extérieurs (cd, disquette, nfs ..).
/opt	lieu d'installation d'applications supplémentaires (comme starOffice, java ..)
/root	répertoire personnel du super-utilisateur root
/sbin	les fichiers exécutables pour l'administration du système
/tmp	stockage des fichiers temporaires
/usr	programmes accessibles à tout utilisateur; sa structure reproduit celle de la racine /
/var	données variables liées à la machine (fichiers d'impression, traces de connexions http, smb .. dans /var/log)
/proc	ce pseudo-répertoire contient une "image" du système (/proc/kcore est l'image de la RAM

1.2.2 Processus

Un processus est l'image en mémoire d'un programme lors de son exécution.
Chaque processus dispose de son espace mémoire, et peut éventuellement contenir un ou plusieurs threads (processus léger).
Le propre d'un système Unix est de pouvoir faire tourner un ou plusieurs processus simultanément, soit en répartissant les processus sur les différents processeurs de la machine, si plusieurs processeurs physiques sont disponibles, soit en temps partagé, et ce de manière transparente pour l'utilisateur.
L'ordonnanceur d'un système Unix, qui s'occupe d'allouer du temps processeur aux processus, est préemptif (par opposition à collaboratif), c'est-à-dire qu'il peut choisir d'arrêter ou de ralentir certains processus au profit d'autres processus plus prioritaires. L'ordonnanceur tente également de conserver des temps d'entrée/sortie acceptables tout en distribuant des ressources aux processus.

Voici un exemple d'une liste de processus tournant sur notre machine **pommard.etudiants.efrei.fr** :

```
vinzelle@pommard:~$ ps auxw
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1944   656 ?        Ss   Apr17   0:08 init [2]
root         2  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   Apr17   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   Apr17   0:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Apr17   0:26 [events/0]
root         6  0.0  0.0      0     0 ?        S<   Apr17   0:00 [khelper]
root        60  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kblockd/0]
root        63  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kacpid]
root        64  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kacpi_notify]
root       135  0.0  0.0      0     0 ?        S<   Apr17   2:56 [ata/0]
root       136  0.0  0.0      0     0 ?        S<   Apr17   0:00 [ata_aux]
root       138  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kseriod]
root       168  0.0  0.0      0     0 ?        S    Apr17   0:00 [pdflush]
root       169  0.0  0.0      0     0 ?        S    Apr17   0:07 [pdflush]
root       170  0.0  0.0      0     0 ?        S<   Apr17   0:13 [kswapd0]
root       171  0.0  0.0      0     0 ?        S<   Apr17   0:00 [aio/0]
root       172  0.0  0.0      0     0 ?        S<   Apr17   0:00 [cifsoplockd]
root       173  0.0  0.0      0     0 ?        S<   Apr17   0:00 [cifsdnotifyd]
root       177  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfslogd/0]
root       178  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfsdatad/0]
root       179  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfs_mru_cache]
root       850  0.0  0.0      0     0 ?        S<   Apr17   0:00 [scsi_eh_0]
root       852  0.0  0.0      0     0 ?        S<   Apr17   4:55 [scsi_eh_1]
root       882  0.0  0.0      0     0 ?        S<   Apr17   0:00 [kpsmoused]
root       886  0.0  0.0      0     0 ?        S<   Apr17   0:00 [rpciod/0]
root       887  0.0  0.0      0     0 ?        S<   Apr17   0:01 [xfsbufd]
root       888  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfssyncd]
root       986  0.0  0.2   2708  1160 ?        S<s  Apr17   0:00 udevd --daemon
root      1536  0.0  0.0      0     0 ?        S<   Apr17   0:00 [ksuspend_usbd]
root      1538  0.0  0.0      0     0 ?        S<   Apr17   0:00 [khubd]
root      2238  0.0  0.0      0     0 ?        S<   Apr17   0:01 [xfsbufd]
root      2239  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfssyncd]
root      2240  0.0  0.0      0     0 ?        S<   Apr17   0:01 [xfsbufd]
root      2241  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfssyncd]
root      2242  0.0  0.0      0     0 ?        S<   Apr17   0:01 [xfsbufd]
root      2243  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfssyncd]
root      2244  0.0  0.0      0     0 ?        S<   Apr17   0:01 [xfsbufd]
root      2248  0.0  0.0      0     0 ?        S<   Apr17   0:00 [xfssyncd]
daemon    2481  0.0  0.0   1684   476 ?        Ss   Apr17   0:00 /sbin/portmap
root      2485  0.0  0.0      0     0 ?        S    Apr17   0:00 [lockd]
root      2803  0.0  0.1   2556   936 ?        Ss   Apr17   0:06 /sbin/syslogd
root      2811  0.0  0.0   1576   376 ?        Ss   Apr17   0:00 /sbin/klogd -x
104       2830  0.0  0.3   7068  1736 ?        Ss   Apr17   0:00 /usr/bin/dbus-daemon --system
109       2839  0.0  1.7  10676  9120 ?        Ss   Apr17   0:02 /usr/sbin/hald
root      2840  0.0  0.2   2888  1040 ?        S    Apr17   0:00 hald-runner
avahi     2886  0.0  0.2   7356  1092 ?        Ss   Apr17   0:00 avahi-daemon: chroot helper
root      2926  0.0  0.5  13704  2896 ?        Ssl  Apr17   0:10 /usr/sbin/nscd
root      2939  0.0  0.1   2892   996 ?        Ss   Apr17   0:00 /usr/sbin/inetd
root      2999  0.0  0.3   5180  1756 ?        Ss   Apr17   0:05 /usr/lib/postfix/master
root      3012  0.0  0.2   5016  1092 ?        Ss   Apr17   0:00 /usr/sbin/sshd
postfix   3021  0.0  0.2   5160  1500 ?        S    Apr17   0:00 qmgr -l -t fifo -u -c
root      3023  0.0  0.5   4608  2732 ?        Ss   Apr17   0:00 /usr/bin/xfs -daemon
root      3195  0.0  0.3  12848  1728 ?        Ss   Apr17   0:00 /usr/sbin/gdm
statd     3207  0.0  0.2   2688  1060 ?        Ss   Apr17   0:00 /sbin/rpc.statd
root      3215  0.0  0.1   3944   676 ?        Ss   Apr17   0:05 /usr/sbin/rpc.idmapd
daemon    3250  0.0  0.0   2044   384 ?        Ss   Apr17   0:00 /usr/sbin/atd
root      3260  0.0  0.1   2196   716 ?        Ss   Apr17   0:01 /usr/sbin/cron
root      3316  0.0  0.0   1572   500 tty1     Ss+  Apr17   0:00 /sbin/getty 38400 tty1
root      3317  0.0  0.0   1572   496 tty2     Ss+  Apr17   0:00 /sbin/getty 38400 tty2
root      3318  0.0  0.0   1572   496 tty3     Ss+  Apr17   0:00 /sbin/getty 38400 tty3
root      3319  0.0  0.0   1572   496 tty4     Ss+  Apr17   0:00 /sbin/getty 38400 tty4
root      3320  0.0  0.0   1572   500 tty5     Ss+  Apr17   0:00 /sbin/getty 38400 tty5
root      3321  0.0  0.0   1572   496 tty6     Ss+  Apr17   0:00 /sbin/getty 38400 tty6
root      4568  0.0  0.5  14200  2960 ?        S    Apr17   0:00 /usr/sbin/gdm
gdm       6378  0.0  2.9  23952 15108 ?        Ss   Apr18   0:31 /usr/lib/gdm/gdmgreeter
lp       18513  0.0  0.0   2820   496 ?        Ss   Apr27   0:00 /usr/sbin/lpd -s
root     28842  0.0  0.4   9016  2516 ?        Ss   18:25   0:00 sshd: vinzelle [priv]
vinzelle 28848  0.0  0.3   9148  1784 ?        S    18:25   0:00 sshd: vinzelle@pts/0
vinzelle 28851  0.0  0.3   5344  1892 pts/0    Ss   18:25   0:00 -bash
postfix  28884  0.0  0.2   5124  1464 ?        S    18:53   0:00 pickup -l -t fifo -u -c
vinzelle 28911  0.0  0.1   4276   820 pts/0    R+   19:26   0:00 ps auxw
```

1.2.3 Communication interprocessus (IPC)

La communication interprocessus sert, d'une part à échanger des données entre les processus, et d'autre part à synchroniser les processus.
Dans le cadre des échanges de données interprocessus, citons : les fichiers, et les segments de mémoire partagée.
Le problème qui se pose bien souvent lors de la mise en œuvre de ce type de mécanisme est que l'accès d'un processus à une ressource a pour effet de provoquer le verrouillage de cette dernière, rendant tout nouvel accès impossible.
En ce qui concerne la synchronisation des processus, trois outils sont à notre disposition : les verrous, déjà évoqués, qui ne s'appliquent qu'aux fichiers, les sémaphores qui sont des systèmes de verrous plus généralistes, et enfin les signaux. Les signaux permettent « d'avertir » un processus que son arrêt est demandé, et peuvent permettre des opérations de « blocage/déblocage » pour éviter les accès simultanées à une ressource.
Il existe enfin d'autres méthodes qui regroupent les fonctions d'échange de données et la synchronisation, il s'agit des files d'attente de message, des sockets Unix (locaux) ou Internet (distants) et enfin des tubes que nous allons exploiter dans ce cours.

1.2.4 Contrôle d'accès

Une ressource Unix, et notamment un fichier, possède plusieurs attributs qui rendent possible sa manipulation. Il s'agit de :

- Un identificateur (un nom de fichier, par exemple)
- Un propriétaire
- Un ensemble de droits d'accès.

Sur la plupart des systèmes Unix, et ce en fonction du système de fichiers, les droits d'accès (lecture, écriture et exécution) sont appliqués à trois entités :

- Le propriétaire du fichier
- Le groupe auquel appartient le fichier
- Les autres utilisateurs

Voici un exemple, toujours sur notre **pommard.etudiants.efrei.fr** :

```
-rw-r--r-- 1 vinzelle vacat  5186 2008-05-03 18:19 arborescence.txt
```

Le fichier identifié par le nom *arborescence.txt* a pour propriétaire l'utilisateur *vinzelle*, et comme groupe *vacat*.
Le propriétaire a le droit de lecteur/écriture (*rw*), le groupe a le droit de lecture seule (*r*) et les autres utilisateurs ont le droit de lecture seule (*r*).
Certains systèmes de fichiers plus évolués comme le **XFS** supportent les **ACL**, c'est-à-dire des droits d'accès étendus qui permettent d'avoir plusieurs propriétaires pour un fichier, et de gérer un héritage des droits dans l'arborescence, comme ce qui existe dans le monde **Microsoft Windows**.
A noter que l'utilisateur *root*, aussi appelé *super-utilisateur*, a accès à tous les fichiers du système en lecture, écriture et exécution.

2 Préambule : le système d'aide en ligne

La commande `man` - pour *manual* - permet de rechercher des informations sur la plupart des commandes et programmes installés sur le système.
Pour l'appeler, il suffit de taper man suivi du nom de la commande pour laquelle on souhaite obtenir l'aide.

```
vinzelle@pommard:~$ man vim
```

L'ensemble de la documentation du système, appelée *manpages*, est accessible via cette commande.
Cette documentation est découpée en plusieurs sections :

1. **Commandes Unix.** Elles sont appelées soit par les utilisateurs, soit directement par les procédures ;
2. **Appel système.** Cette section décrit les points d'entrée du noyau, notamment son interface en C ; cette section n'est pas installée sur vos machines de l'Efrei.
3. **Sous-programmes et bibliothèques.** Cette section décrit les procédures systèmes qui ne font pas appel aux primitives du système ;
4. **Formats des fichiers.** Cette section décrit l'ensemble des formats de fichiers auxquels peut être confronté l'utilisateur ; cette section n'est pas installée sur vos machines de l'Efrei.
5. **Divers**. Contient souvent des données propres à la distribution installée;
6. **Jeux ;** cette section n'est pas installée sur vos machines de l'Efrei.
7. **Fichiers spéciaux**. sur vos machines, cette section contient de la documentation spécialisée développement Debian.
8. **Commandes spécialisées Debian**. Sur vos machines, cette section contient des commandes systèmes qui sont spécifiques au système Linux Debian.

La commande `man` peut également être utilisée pour rechercher un terme dans l'ensemble des pages de manuel, grâce à l'option *-K.*

3 La gestion des utilisateurs

3.1 Se loger et se déloger

A l'invite du système, il suffit d'entrer son login puis son mot de passe. Les comportements sont multiples en fonction des distributions. L'utilisateur a en général le droit à trois essais avant un blocage temporaire de son terminal.

Que le programme de login que vous utilisiez soit graphique (*gdm*, *kdm* et autres) ou texte (*login*) les mécanismes mis en œuvre pour l'authentification sont les mêmes.

Pour se déconnecter, il suffit de taper la commande `exit`, `logout` ou la combinaison de touche ctrl+d quand on est connecté sur un terminal.

3.2 Connaître son UID/GID

Un utilisateur est décrit par son UID (*Unique Identifier*) et son GID (*Group IDentifier*).

L'UID lui est propre, alors qu'il partage son GID avec les autres membres du groupe auquel il appartient.

La commande `id` permet de connaitre ces informations.

3.3 Retrouver l'UID/GID sur le système

Si les comptes utilisateurs sont stockés sur le système, la liste des utilisateurs est stockée dans :

```
/etc/passwd
```

Et celle des groupes est stockée dans :

```
/etc/group
```

Les systèmes Unix avec lesquels nous travaillons à l'Efrei authentifient leurs utilisateurs grâce à un annuaire LDAP via le mécanisme PAM (*Plugable Authentication Module*) de Linux, ce qui explique que nous ne trouvons aucune trace des comptes utilisateurs dans le fichier `/etc/passwd`.

En revanche, l'utilisateur `root` local est bien visible.

3.4 Changer son mot de passe

Pour changer son mot de passe Unix, il suffit d'utiliser la commande `passwd`.

```
skyce@samvimes ~ $ passwd
Changing password for skyce
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

En revanche, sur les systèmes de l'Efrei, où le référentiel n'est pas le traditionnel fichier plat, mais un annuaire LDAP, la commande est un peu différente, il s'agit de `kpasswd`, le k signifiant Kerberos.

```
vinzelle@pommard:~$ kpasswd
Password for vinzelle@ETUDIANTS.EFREI.FR:
Enter new password:
Enter it again:
```

Attention, le mot de passe étant partagé entre les systèmes Linux et Windows, il est nécessaire qu'il respecte certaines contraintes :

- 10 caractères minimum ;
- il faut au moins 3 de ces éléments : majuscules, minuscules, chiffres et caractères spéciaux.

On évite les caractères spéciaux au début et à la fin du mot de passe pour des problèmes de compatiblité.

3.5 Qui est là ?

`who` donne la liste des utilisateurs connectés au système avec indication du nom du terminal, de la machine utilisée et de l'heure d'entrée en session.

Un exemple sur **choam.efrei.fr** :

```
vinzelle@choam:~$ who
litchink pts/0        2008-05-06 19:23 (amontsouris-159-1-35-92.w90-46.abo.wanadoo.fr)
maupard  pts/1        2008-05-06 19:41 (kf.couak.net)
aribaud  pts/2        2008-05-06 19:48 (aut75-8-88-171-185-245.fbx.proxad.net)
jutteau  pts/3        2008-05-06 20:16 (corbak.couak.net)
vinzelle pts/4        2008-05-06 20:53 (gate.skyce.net)
```

La commande `finger` propose des informations semblables.

Pour connaitre les utilisateurs qui sont connectés à aux machines du réseau local, il suffit d'employer la commande `rusers`.

```
vinzelle@pommard:~$ rusers
belsambar.etudiants. root
oz.etudiants.efrei.f mahieuxa mahieuxa mahieuxa
stonard.etudiants.ef boudet
astranaar.etudiants. mbangomb
pommard.etudiants.ef vinzelle
emilion.etudiants.ef purdessy purdessy
corton.etudiants.efr parmelan
chambertin.etudiants nicoled nicoled
joyce.etudiants.efre verbeke verbeke
undercity.etudiants. maupard maupard maupard maupard maupard maupard
dafalgan.etudiants.e root
xander.etudiants.efr menardg
angel.etudiants.efre cravero
```

3.6 Où suis-je ?

Pour connaitre le nom de la machine sur laquelle on est connecté, il suffit d'utiliser la commande `uname`.

3.7 Exercices

3.7.1 Exercice

Affichez le *man* de la commande `man` ;

Afficher le *man* de la commande utilisateur *printf* ;

Indiquez la ligne de commande nécessaire pour chacune de ces opérations. Inutile de recopier le contenu de la *manpage* – voir chapitre 2.

3.7.2 Exercice

A l'aide de la commande `id`, retrouvez l'UID et le GID de votre utilisateur.

Même question pour l'UID et le GID d'un autre utilisateur.

3.7.3 Exercice

A l'aide de la commande `cat`, *et depuis votre dossier d'utilisateur,* affichez la ligne qui correspond à l'utilisateur root dans le fichier /etc/passwd.

3.7.4 Exercice

Quelle est la commande qui permet de changer son mot de passe ?

3.7.5 Exercice

A l'aide de la commande `who` uniquement et des paramètres adéquats, n'affichez que le nom de votre utilisateur actuellement connecté, dans le terminal courant. Votre nom ne doit apparaitre qu'une seule fois.

3.7.6 Exercice

A l'aide de la commande `uname`, déterminez le nom de l'OS que vous utilisez actuellement, et indiquez la version du kernel utilisé.

4 Quelques commandes et raccourcis claviers courants

4.1 Raccourcis claviers

1. **Déplacement**
   Ctrl + a : aller au début de la ligne
   Ctrl + e : aller à la fin de la ligne
   Alt + b : se déplacer mot par mot dans la ligne de commande en arrière
   Alt + f : se déplacer mot par mot dans la ligne de commande en avant
   Ctrl + xx : positionner le curseur au début de la ligne ou à la fin, alternativement
   Shift+PageUp : afficher l'écran précédent
   Shift+PageDown : afficher l'écran suivant
2. **Couper / Coller**
   Ctrl + k : couper la chaîne du curseur jusqu'à la fin de la ligne
   Ctrl + u : couper la chaîne du curseur jusqu'au début de la ligne
   Ctrl + w : couper le mot avant le curseur
   Ctrl + y : coller une chaîne
3. **Modification**
   Ctrl + t : inverser la position des deux caractères avant le curseur
   Alt + t : inverser la position des deux mots avant le curseur
   Alt + c : mettre une lettre en majuscule
   Alt + l : mettre un mot en en minuscule
   Alt + u : mettre un mot en majuscule
   Alt + . : réécrire le paramètre de la dernière commande
4. **Divers**
   Tab : complète la commande en cours de frappe
   Ctrl + l : effacer le contenu de l'écran (équivalent à `clear`)
   Ctrl + r : rechercher une commande déjà tapée
   Ctrl + _ : annuler la dernière modification
   Ctrl + c : arrêter la commande en court
   Ctrl + d : envoie un caractère *EOF*, et permet également de se déconnecter du *terminal* courant (équivalent à `exit` ou `logout`)
   ↑ & ↓ : parcourt de l'historique des commandes saisies

4.2 Manipulation de l'historique

La commande `history` donne la liste numérotée des *n* dernières commandes.
Un certain nombre de combinaisons de touches permettent de manipuler des commandes mémorisées. Ces combinaisons sont les suivantes :

`!!`	Relance la dernière commande exécutée
`!10`	Relance la commande numéro 10
`!ls`	Relance la dernière commande commençant par `ls`
`!-n`	Relance la n^{ième} commande avant la dernière, sachant que la dernière est celle que l'on saisit
`!?toto ?`	Renouvelle la dernière commande qui contient la chaîne de caractères *toto*

5 Commandes relatives à la gestion de fichiers et répertoires

5.1 Wildcards, notations et noms de fichiers

5.1.1 Wildcards

Le caractère '*' représente un ensemble de caractères quelconques.
Le caractère '?' représente un caractère donné.
La chaîne de caractères **[a-z]** représente l'ensemble de caractères compris entre les crochets.
La chaîne de caractères **[^a-z]** correspond à tous les caractères sauf ceux compris entre les crochets.

5.1.2 Notations

Le répertoire courant se note « . ».
Le répertoire parent du répertoire courant se note « . . ».

5.1.3 Noms de fichiers

A noter que les caractères :
`\ > < | $ ? & [ ] * ! «  `( ) @ ~ <espace>`
sont à bannir absolument des noms de fichiers et de répertoires.

5.2 Manipulation des répertoires

5.2.1 pwd

Print Working Directory
Permet d'afficher le chemin d'accès du répertoire courant.

```
vinzelle@pommard:~$ pwd
/users/guest/vinzelle
```

5.2.2 cd

Change Directory
Permet de changer de répertoire de travail

```
vinzelle@pommard:~$ pwd
/users/guest/vinzelle
vinzelle@pommard:~$ cd plop
vinzelle@pommard:~/plop$ pwd
/users/guest/vinzelle/plop
```

Pour revenir chez soi, dans le « home » de son utilisateur, on peut utiliser `cd` sans argument.

5.2.3 mkdir

MaKe DIRectory
Crée un nouveau répertoire

```
vinzelle@pommard:~$ mkdir test
vinzelle@pommard:~$ cd test
vinzelle@pommard:~/test$ pwd
/users/guest/vinzelle/test
```

5.2.4 rmdir

ReMove DIRectory

Supprime un répertoire, s'il est vide.

```
vinzelle@pommard:~$ rmdir test/
```

5.2.5 df

Disk usage by Filesystem

Permet d'afficher l'espace disque restant en bloc de 1Ko sur chaque système de fichiers monté sur le système.

```
vinzelle@pommard:~$ df
Sys. de fich.        1K-blocs       Occupé Disponible Capacité Monté sur
/dev/sda1               991168    106008    885160  11% /
tmpfs                   256892         0    256892   0% /lib/init/rw
udev                     10240        64     10176   1% /dev
tmpfs                   256892       588    256304   1% /dev/shm
/dev/sda6              7990080   6348956   1641124  80% /usr
/dev/sda7              3989888      4516   3985372   1% /tmp
/dev/sda8              3989888    503476   3486412  13% /var
/dev/sda9              1548032      4384   1543648   1% /home
```

5.2.6 du

Disk Usage

Permet d'afficher l'occupation disque en bloc de 1Ko des sous répertoires du répertoire spécifié ou, si aucun répertoire n'est spécifié, du répertoire courant.

```
vinzelle@pommard:~$ du
125     ./Mail
1       ./News
1       ./.gnupg
1       ./.muttGn65QD
4       ./.ssh
166     ./Unix
378     .
```

5.2.7 find

Permet de rechercher un fichier à partir du répertoire donné.

Cette commande est assez complexe, et permet de faire des recherches à partir de nombreux critères (date de création, de dernière modification, taille, *inode*…). En plus d'afficher la liste des résultats trouvés la commande `find` permet également de faire des traitements sur la liste des fichiers trouvés.

- Recherche par nom

```
vinzelle@pommard:~$ ls
arborescence.txt  cv.pdf  Mail  News  Unix
vinzelle@pommard:~$ find ./ -name cv.pdf
./cv.pdf
```

- Recherche par taille

```
vinzelle@pommard:~$ find ./ -type f -size +20000c
./Mail/received
./cv.pdf
./Unix/CoursDétailléEnCoursDeRédaction.doc
./Unix/Unix_history-simple.svg
./Unix/CoursDétailléEnCoursDeRédaction.docx
```

- Recherche par nom et suppression

```
vinzelle@pommard:~$ ls
arborescence.txt  cv.pdf  Mail  News  plop  Unix
vinzelle@pommard:~$ find . -name plop -exec rm {} \;
vinzelle@pommard:~$ ls
arborescence.txt  cv.pdf  Mail  News  Unix
```

5.3 Manipulation des fichiers

5.3.1 ls

LiSt files

Permet d'obtenir la liste et les caractéristiques des fichiers contenus dans un répertoire, ou à défaut de paramètre, dans le répertoire courant. L'affichage par défaut se fait par ordre alphabétique.

```
vinzelle@pommard:~$ ls -l
total 9
-rw-r--r-- 1 vinzelle vacat 5186 2008-05-03 18:19 arborescence.txt
drwx------ 2 vinzelle vacat  512 2008-05-07 17:32 Mail
drwx------ 2 vinzelle vacat  512 1995-08-30 15:45 News
drwxr-xr-x 2 vinzelle vacat  512 2008-05-04 18:08 Unix
```

5.3.2 cat

conCATenate

La commande `cat` est multi-usage, car elle permet d'afficher, de créer, de copier et de concaténer des fichiers à l'aide de quelques redirections que nous explorerons plus tard (**section 6**).

- Afficher

```
vinzelle@pommard:~$ cat /etc/hosts
127.0.0.1       localhost
192.168.1.96    pommard.etudiants.efrei.fr      pommard
```

- Créer

```
vinzelle@pommard:~$ cat > test
Ceci est un test
TD Unix Efrei 2008
Ctrl + d^F
vinzelle@pommard:~$ cat test
Ceci est un test
TD Unix Efrei 2008
```

- Copier

```
vinzelle@pommard:~$ cat test > test2
vinzelle@pommard:~$ cat test2
Ceci est un test
TD Unix Efrei 2008
```

- Concaténer

```
vinzelle@pommard:~$ cat test2 >> test
vinzelle@pommard:~$ cat test
Ceci est un test
TD Unix Efrei 2008
Ceci est un test
TD Unix Efrei 2008
```

5.3.3 cp

CoPy

Permet de copier les fichiers. La commande `cp` s'utilise sous plusieurs formes : copie d'un fichier source dans un fichier de destination ou copie d'un fichier dans un répertoire par exemple.

- Copie d'un fichier source dans un fichier de destination

```
vinzelle@pommard:~$ cp test test3
vinzelle@pommard:~$ cat test3
Ceci est un test
TD Unix Efrei 2008
Ceci est un test
TD Unix Efrei 2008
```

- Copie d'un fichier dans un répertoire

```
vinzelle@pommard:~$ mkdir dirtest
vinzelle@pommard:~$ cp test3 ./dirtest/
vinzelle@pommard:~$ ls ./dirtest/
test3
```

Dans le cas où l'on veuille copier un fichier dans le répertoire parent du répertoire courant :

```
vinzelle@pommard:~/dirtest$ ls ../
dirtest  Mail  News  test  test2  Unix
vinzelle@pommard:~/dirtest$ ls
test3
vinzelle@pommard:~/dirtest$ cp test3 ../
vinzelle@pommard:~/dirtest$ ls ../
dirtest  Mail  News  test  test2  test3  Unix
vinzelle@pommard:~/dirtest$ rm ../test3
```

5.3.4 mv

MoVe

Déplace un fichier ou un répertoire. Fonctionne comme une copie suivie d'une suppression.

- Déplacement d'un fichier

```
vinzelle@pommard:~/dirtest$ ls
test3
vinzelle@pommard:~/dirtest$ mv test3 ../
vinzelle@pommard:~/dirtest$ ls ../
dirtest  Mail  News  test  test2  test3  Unix
```

- Déplacement d'un répertoire

```
vinzelle@pommard:~/testdir$ ls
testdir2
vinzelle@pommard:~/testdir$ mv testdir2/ ../
vinzelle@pommard:~/testdir$ ls ../
testdir2
```

5.3.5 rm

ReMove

Supprime un ou plusieurs fichiers d'un répertoire.

```
vinzelle@pommard:~$ ls
test
vinzelle@pommard:~$ rm test
vinzelle@pommard:~$ ls
```

5.3.6 ln

LiNk

Permet de désigner un fichier par plusieurs noms différents, de créer des « liens » entre les fichiers.
Les liens peuvent être de deux types : des « *hard links* » ou des « *soft links* ».
Créer un « *hard link* » revient à créer deux noms pour le même fichier. Si l'un des deux fichiers est modifié, alors le fichier qui lui est lié est également modifié.
Dans le cas du « *soft link* », on crée un petit fichier spécial qui contient un chemin vers un fichier. Le « *soft link* » fonctionne même entre deux partitions différentes, ce qui n'est pas le cas du « *hard link* ».
Exemple de « *hard link* »:

```
vinzelle@pommard:~$ cat test
Ceci est un test
TD Unix Efrei 2009
vinzelle@pommard:~$ ln test test2
vinzelle@pommard:~$ echo "by skyce" >> test2
vinzelle@pommard:~$ cat test
Ceci est un test
TD Unix Efrei 2009
by skyce
vinzelle@pommard:~$ ls -l
total 2
-rw-r--r-- 2 vinzelle vacat  45 2008-05-09 14:06 test
-rw-r--r-- 2 vinzelle vacat  45 2008-05-09 14:06 test2
```

Exemple de « *soft link* »:

```
vinzelle@pommard:~$ ln -s test2 test3
vinzelle@pommard:~$ echo "Linux rulez" >> test3
vinzelle@pommard:~$ cat test2
Ceci est un test
TD Unix Efrei 2009
by skyce
Linux rulez
vinzelle@pommard:~$ ls -l
total 3
-rw-r--r-- 2 vinzelle vacat  57 2008-05-09 14:09 test
-rw-r--r-- 2 vinzelle vacat  57 2008-05-09 14:09 test2
lrwxrwxrwx 1 vinzelle vacat   5 2008-05-09 14:08 test3 -> test2
```

5.3.7 touch

Permet (entre autres) de créer un fichier vide.

```
vinzelle@pommard:~$ touch test4
```

```
vinzelle@pommard:~$ ls
test4
```

5.3.8 echo

Affiche à l'écran le texte qui suit la commande echo.

```
vinzelle@pommard:~$ echo Bonjour le Monde
Bonjour le Monde
```

5.4 Manipulation des droits d'accès

5.4.1 Droit d'accès aux fichiers et répertoires

A chaque fichier est associé un ensemble d'indicateurs précisant les droits d'accès au fichier.
Pour chaque fichier, il existe trois types d'utilisateurs :

- le propriétaire du fichier,
- les membres du groupe du propriétaire du fichier,
- les autres utilisateurs du système

Pour chaque fichier et par type d'utilisateur, il existe trois modes principaux :

- autorisation d'écriture (w)
- autorisation de lecture (r)
- autorisation d'exécution (x)

En plus de ces neuf bits (rwx rwx rwx), Unix définit trois autres bits de permission : SUID, SGID et t, qui seront présentés plus loin.
Unix gère plusieurs types de fichiers :

- fichier ordinaire (-)
- répertoire (d)
- fichier spécial : périphérique accédé en mode caractère (c)
- fichier spécial : périphérique accédé en mode bloc (b)
- tube nommé (named pipe) (p)
- lien symbolique (l)
- socket (s)

Dans le cas des répertoires, l'interprétation des droits est légèrement différente de celle concernant les fichiers.
L'interprétation des protections pour les répertoires est la suivante :

- **r** : autorise la lecture du contenu du répertoire comme dans le cas des fichiers ; permet donc de voir la liste des fichiers qui sont dans le répertoire
- **x** : autorise l'accès au répertoire (à l'aide de la commande cd).
- **w** : autorise la création, la suppression et le changement du nom d'un élément du répertoire. Cette permission est indépendante de l'accès aux fichiers dans le répertoire.

Attention, si un utilisateur n'a pas l'autorisation **w** sur un répertoire, il ne pourra pas supprimer un fichier lui appartenant dans ce répertoire mais il pourra seulement le modifier.
La possibilité de supprimer un fichier n'est donc pas fixée dans les permissions de ce fichier, mais dans les permissions du répertoire qui le contient.

```
vinzelle@pommard:~/testdir$ ls
vinzelle@pommard:~/testdir$ touch truc
vinzelle@pommard:~/testdir$ ls -l
total 1
-rw-r--r-- 1 vinzelle vacat 6 2008-05-09 19:37 truc
vinzelle@pommard:~/testdir$ chmod u-w ../testdir/
vinzelle@pommard:~/testdir$ touch truc2
touch: ne peut faire un touch sur `truc2': Permission non accordée
vinzelle@pommard:~/testdir$ echo "texte" > truc
vinzelle@pommard:~/testdir$ cat truc
texte
vinzelle@pommard:~/testdir$ rm truc
rm: ne peut enlever `truc': Permission non accordée
vinzelle@pommard:~/testdir$ ls -l ../testdir/
total 1
```

```
-rw-r--r-- 1 vinzelle vacat 6 2008-05-09 19:37 truc
vinzelle@pommard:~/testdir$ ls -l ../
total 1
dr-xr-xr-x 2 vinzelle vacat  512 2008-05-09 19:37 testdir
```

5.4.2 Modification des droits d'accès aux fichiers - chmod

CHange MOD

Il existe deux méthodes pour utiliser cette commande, soit en utilisant la description des protections par un nombre octal, soit en utilisant le mode symbolique.

5.4.2.1 Mode octal

Dans le mode octal, un bit activé correspond à un 1, un bit non activé correspond à un 0.
Ainsi, le groupe de permissions rwx rw- r-x se lit 111 110 101, soit en décimal 765.

```
vinzelle@pommard:~$ ls -l
total 1
-rw-r--r-- 2 vinzelle vacat   57 2008-05-09 14:09 test
vinzelle@pommard:~$ chmod 765 test
vinzelle@pommard:~$ ls -l
total 1
-rwxrw-r-x 2 vinzelle vacat   57 2008-05-09 14:09 test
```

5.4.2.2 Mode symbolique

Le mode symbolique permet une description absolue ou relative des droits d'accès, comme suit :

```
chmod [who]op[expression] fichier
```

où :

- `who` est une combinaison de lettres **u** (user=propriétaire), **g** (group=groupe), **o** (other=autre) ou **a** (all=tous) ce qui équivaut à **ugo** ;
- `op` est un opérateur, **+** qui permet d'ajouter un droit d'accès, **-** qui permet de supprimer un droit d'accès ou **=** qui permet d'affecter de manière absolue (tous les autres bits sont remis à zéro)
- `permission` est soit **r** (read=lecture), **w** (write=écriture) ou **x** (execution=exécution)

Ainsi le mode octal 765 peut s'écrire **u=rwx,g=rw,o=rx**.

```
vinzelle@pommard:~$ ls -l
total 1
-rw-r--r-- 2 vinzelle vacat   57 2008-05-09 14:09 test
vinzelle@pommard:~$ chmod u=rwx,g=rw,o=rx test
vinzelle@pommard:~$ ls -l
total 1
-rwxrw-r-x 2 vinzelle vacat   57 2008-05-09 14:09 test
```

5.5 Modification du propriétaire et du groupe

5.5.1 Modification du propriétaire - chown

CHange OWNer

La commande `chown` permet de changer le propriétaire d'un fichier. Pour des raisons de sécurité, seul l'administrateur peut modifier le propriétaire d'un fichier ou d'un répertoire. Il s'agit de l'utilisateur *root*.

```
root@samvimes skyce # ls
root@samvimes skyce # touch test
root@samvimes skyce # ls -l
total 1
-rw-r--r-- 1 root  root    0 May  9 19:52 test
root@samvimes skyce # chown skyce:users test
root@samvimes skyce # ls -l
total 1
-rw-r--r-- 1 skyce users   0 May  9 19:52 test
```

On peut voir dans cet exemple que la commande `chown` permet de changer le propriétaire et le groupe auxquels appartient un fichier simultanément.

5.5.2 Modification du groupe - chgrp

CHange GRouP

Il n'est possible de changer le groupe auquel appartient un fichier seulement si l'utilisateur qui fait cette modification appartient au groupe cible.

```
skyce@samvimes ~ $ id skyce
uid=1000(skyce)gid=100(users)
groups=100(users),5(tty),10(wheel),18(audio),19(cdrom),85(usb),5008(samba)
skyce@samvimes ~ $ chgrp samba test
skyce@samvimes ~ $ ls -l
total 1
-rw-r--r-- 1 skyce samba   0 May  9 19:52 test
skyce@samvimes ~ $ chgrp root test
chgrp: changing group of `test': Operation not permitted
```

5.6 Manipulation du contenu des fichiers textes

5.6.1 more

Permet d'afficher à l'écran un fichier page par page.

5.6.2 less

Permet également d'afficher un fichier page par page, mais offre des fonctionnalités supplémentaires notamment le retour en arrière lors du défilement du fichier.

5.6.3 file

Permet d'identifier le type du fichier passé en paramètre.

```
vinzelle@pommard:~$ file arborescence.txt
arborescence.txt: ASCII C++ program text, with CRLF line terminators
vinzelle@pommard:~$ file backupUnixLecture.sh
backupUnixLecture.sh: Bourne-Again shell script text executable
vinzelle@pommard:~$ file TD_Unix_Efrei_052008.pdf
TD_Unix_Efrei_052008.pdf: PDF document, version 1.4
vinzelle@pommard:~$ file Unix_history-simple.svg
Unix_history-simple.svg: XML document text
```

5.6.4 head

Affiche les 10 premières lignes du fichier passé en paramètre.

5.6.5 tail

Affiche les 10 dernières lignes du fichier passé en paramètre.

5.6.6 grep

Cette commande est un filtre qui permet de rechercher, dans un ou plusieurs fichiers, toutes les lignes qui contiennent une chaîne de caractère donnée.
La chaîne de caractère recherchée peut être décrite par une expression régulière réduite.
Les caractères `$ * [ ^ ( )` et `\` ont une signification particulière pour le shell. Pour pouvoir les utiliser dans une expression régulière, il faut mettre l'expression entre quotes.
Affichage de toutes les lignes du fichier `/etc/passwd` contenant le mot root

```
vinzelle@pommard:~$ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

5.6.7 wc

Word Count

Permet de compter les lignes, mots ou caractères d'un fichier texte. Un mot est déterminé par des espaces, des tabulations ou une nouvelle ligne.

```
vinzelle@pommard:~$ cat test3
Ceci est un test
TD Unix Efrei 2008
Ceci est un test
TD Unix Efrei 2008
vinzelle@pommard:~$ wc -l test3
4 test3
vinzelle@pommard:~$ wc -w test3
16 test3
vinzelle@pommard:~$ wc -c test3
72 test3
```

5.7 Exercices

5.7.1 Exercice

Dans le cas d'un système de fichier, qu'est-ce qu'un chemin absolu ? Qu'est-ce qu'un chemin relatif ?

5.7.2 Exercice

Dans votre dossier d'utilisateur, créez l'arborescence suivante, en n'utilisant que des chemins relatifs et sans utiliser la commande `cd` :

```
repA
|-- fich11
|-- fich12
|-- repB
|   |-- fich21
|   `-- fich22
`-- repC
    |-- fich31
    `-- fich32
        `-- repc
```

5.7.3 Exercice

Comment déplacer toute l'arborescence repC dans le répertoire repB ? Supprimez tout sauf repA, fich11 et fich12.

5.7.4 Exercice

Renommez le répertoire repA en rep1 et le fichier fich11 en fichAA.

5.7.5 Exercice

Toujours dans votre répertoire d'accueil, créez en une seule commande l'arborescence repA/repB/repC. Effacez ensuite cette arborescence en une seule commande sans qu'il vous soit demandé de confirmation.

5.7.6 Exercice

La commande `ls` est très utilisée sous Unix. Que permet la commande *ls -lhiatr* ? Que représentent "." et ".." ?

5.7.7 Exercice

En utilisant les commandes `mkdir`, `echo` et `cat`, créez dans un nouveau répertoire de nom `dirtest` le fichier `bienvenue.sh` contenant la ligne de commande :

```
echo Bienvenue dans le monde Unix
```

Exécutez ce fichier.

5.7.8 Exercice

En utilisant la commande `echo`, créez un fichier que vous pouvez lire, modifier et supprimer. Faites en la démonstration avec les commandes `cat`, `echo` et `rm`.

5.7.9 Exercice

En utilisant les commandes `chmod`, `mkdir` et `echo` créez un fichier que vous pouvez lire, et modifier mais que vous ne pouvez pas supprimer. Faites en la démonstration avec les commandes `cat`, `echo` et `rm`.

5.7.10 Exercice

En utilisant les commandes `chmod` et `echo`, créez un fichier que vous pouvez lire, mais que vous ne pouvez ni modifier, ni supprimer. Faites en la démonstration avec les commandes `cat`, `echo` et `rm`.

5.7.11 Exercice

Explicitez l'usage de la commande `umask`.

Dans quel cas les permissions d'un fichier à sa création sont-elles différentes des permissions fixées par `umask` ?

5.7.12 Exercice

Cet exercice doit se faire en binôme. Pour cela, ouvrez un deuxième terminal sur lequel votre binôme s'authentifiera grâce à la commande `su`. Vous pouvez utiliser `/tmp` pour créer vos fichiers et répertoire.

Modifiez les permissions du fichier créé à l'exercice 5.7.8 ci-dessus de telle façon que votre binôme puisse le lire et l'exécuter, mais ne puisse pas le modifier ni le supprimer.

Pouvez-vous modifier les permissions de ce fichier de telle sorte que votre binôme puisse le lire, le modifier et l'exécuter alors que vous-même ne pouvez pas le modifier ?

5.7.13 Exercice

En utilisant la commande `find`, trouvez et listez les noms de :

1. tous les fichiers sous le répertoire /etc dont les noms commencent par rc
2. tous les fichiers réguliers vous appartenant
3. tous les sous répertoires de /etc
4. tous les fichiers réguliers se trouvant sous votre répertoire d'accueil et qui n'ont pas été modifié dans les 10 derniers jours.

5.7.14 Exercice

Créez un fichier nommé `-i` et supprimez le.

5.7.15 Exercice

Créez dans le répertoire `rep1` les fichiers suivants : `fich1`, `fich2`, `fich11`, `fich12`, `fich1a`, `fich33`, `.fich1`, `.fich2`, `toto`, `afich`.

Listez les fichiers :

1. dont les noms commencent par fich,

2. dont les noms commencent par fich suivi d'un seul caractère,
3. dont les noms commencent par fich suivi d'un chiffre,
4. dont les noms commencent par « . »,
5. dont les noms ne commencent pas par f.
6. dont les noms contiennent fich.

5.7.16 Exercice

Créez un fichier texte et un « hard link » sur ce fichier dans le même répertoire.

Vérifiez que les deux noms correspondent au même *inode*.

Changez les permissions de l'un et vérifiez que les permissions de l'autre ont suivi.

Modifiez le contenu de l'un et relisez le contenu de l'autre.

Supprimez l'un, que devient l'autre ? Essayer de créer un nouveau lien entre un de ces noms et un nouveau nom dans /tmp. Expliquez.

5.7.17 Exercice

Créez un fichier texte et un lien symbolique sur ce fichier texte dans le même répertoire. Quelles sont les permissions du lien symbolique ?

Avec la commande `more` ou la commande `less`, affichez le contenu du lien symbolique.

Effacez le fichier texte. Que devient le lien symbolique ?

5.7.18 Exercice

Affichez l'espace disque restant sur chacun des disques de votre système. Prenez garde à utiliser une unité de mesure compréhensible !

Dans votre répertoire utilisateur, affichez la taille de chaque élément du premier niveau de votre arborescence. Utilisez également une unité de mesure « lisible » par un être humain.

5.7.19 Exercice

Visualisez le type de fichier de /usr/bin/who et de /etc/passwd à l'aide de la commande `file`. Que vous apprend `file` ?

5.7.20 Exercice

Affichez les onze premières lignes de /etc/services et les onze dernières lignes /etc/passwd.