Systèmes d'exploitation
L'exemple de Linux

1. Généralités

Unix est un système d'exploitation réseau, multitâches et multi-utilisateurs.
Linux est une version universitaire gratuite dérivée d'Unix.
Linux est largement utilisé comme système d'exploitation de serveurs de communication dans le monde Internet.
Il peut assumer le rôle de serveur de fichiers (NFS, SMB), de FireWall, de serveur de Proxy, de serveur de noms (WINS, DNS), de serveur d'authentification (NIS, LDAP, Serveur de Domaine Microsoft), de serveur Web / PHP (Apache), de serveur de bases de données (MyQSL), etc.…

Il fournit en standard plusieurs langages de développement (C, C++, Pascal, …) et de script (bourne shell, python, …) dans l'environnement de type terminal texte ou fenêtres graphiques (QT creator).

On retrouve Linux dans les systèmes temps réels (RTLinux) et dans les systèmes embarqués de l'industrie. Il est présent également dans le monde des nano-ordinateurs (Raspberry).

Ces atouts en font une bonne plate-forme d'apprentissage tout en étant très en phase avec l'industrie.

Le système d'exploitation Linux se compose d'un noyau (**kernel**) chargé au démarrage, et qui gère l'ensemble de la machine pendant toute la durée de son utilisation. Les parties électroniques non gérées par le noyau (cartes récentes par ex.) sont prises en charge par l'adjonction de **modules** chargés au démarrage du système (les commandes `lsmod` et `modinfo` permettent de les visualiser).

Linux propose depuis toujours une interface utilisateur de type « interpréteur commandes en mode terminal texte ». Cette interface est appelée le **shell**. Il existe plusieurs variantes de shells (sh, csh, bash, …). Les commandes indiquées dans ce document sont utilisables uniquement à partir du shell.
Linux propose aussi une interface utilisateur graphique, basée sur son système X-Windows. Selon les distributions, cette interface s'appelle KDE, Gnome, Xfce, MATE, LXQt, …

Le paramétrage du système et des logiciels se fait par la modification (avec un éditeur de texte) de fichiers de configuration (stockés généralement dans le dossier `/etc`).

Des outils de configuration existent dans certaines distributions de Linux ; ils proposent la gestion du système via une interface utilisateur par menus graphiques ou même via une interface Web. Cependant les administrateurs de machines Linux préfèrent utiliser le mode « ligne de commande » qui permet des tâches d'administration automatique et répétitives grâce aux *scripts*.

Comme dans tout système d'exploitation complexe, on définit un **administrateur** (utilisateur *root*, ou sur certain système comme Raspberry, le préfixe *sudo*) dont le rôle est de veiller à ce que le système soit disponible aux utilisateurs non initiés. Ses tâches sont (liste non exhaustive) : Gérer l'espace disque, la sécurité et l'accès, les applications actives, le paramétrage du système, la mise à niveau du système et des logiciels.

Le concepteur de logiciels dans l'environnement Linux doit connaître la spécificité de ce système pour adapter ses techniques de développement. Dans une certaine mesure, le concepteur de logiciel doit avoir une partie des compétences de l'administrateur.

2. Gérer l'espace disque

L'espace disque est découpé en **partitions** (physiques et/ou logiques, système MBR ou EFI/GPT).

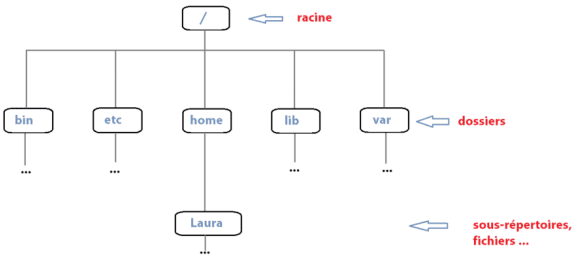
Sous Linux, chaque partition est convertie en **système de fichiers**, sauf une partition appelée SWAP qui sert de complément à la mémoire RAM. Cette zone est gérée par le système de façon transparente pour l'administrateur.

Le système de fichier Linux est constitué d'une zone appelée **superblock** qui contient les noms des fichiers et leur emplacement dans la **zone de données**.

`df` permet de consulter l'état des systèmes de fichiers en cours d'utilisation par le système.
`fdisk, gdisk, parted` permettent de créer des partition et de leur affecter un type (Unix, Ext4, Swap, …) ;
`e2fsck` permet de vérifier l'intégrité du système de fichiers ;
`mkfs` permet de convertir la partition en système de fichiers (formatage logique) ;

2.1. Gérer l'arborescence des fichiers

Le système de fichiers est organisé sous forme d'une **arborescence de dossiers** avec une racine (*root*) unique.
Dans le superblock, chaque **fichier** est identifié par un numéro **Inode** (unique dans le système de fichier).



A un Inode est associé toutes les caractéristiques du fichier :

- Un nom (ou plusieurs noms si on utilise le système du lien avec la commande **`ln`**) ;
- Un type. La notion de fichier est très étendue sous Linux. Un fichier peut être de type normal (-) (programme ou donnée), de type dossier (type **d**), de type spécial (fichier d'accès à un périphérique (type **b** ou **c**), tube nommé (type **p**), lien symbolique (type **l**), ou dynamique (fichiers du dossier /proc qui donnent des informations sur l'état du système)).
- Des informations sur le propriétaire du fichier et sur les droits d'accès à ce fichier.
- Le nombre de liens de ce fichier (voir §2.3).
- La taille du fichier, ou les caractéristiques propres à son type.
- La date d'accès, de dernière modification, de changement de droits d'accès.
- L'emplacement du contenu du fichier.

Liste de fichiers obtenue avec la commande :

```
# ls -li
   1200 crw--w--w-   1 root     root       4,   0 fév  1  2002 /dev/tty0
   4501 brw-rw----   1 root     disk       3,  65 aoû  6  2000 /dev/hdb1

 123691 drwxr-xr-x   8 root     root         32768 nov  4 18:52 dev
 123649 drwxr-xr-x  29 root     root          8192 nov  7 11:36 etc
 200937 drwxr-xr-x   2 root     root          4096 jui 29  2000 floppy
      2 drwxr-xr-x  12 root     root          4096 sep 26 14:20 home
  46370 drwxr-xr-x   5 root     root          4096 déc  5  2000 lib
     11 drwxr-xr-x   2 root     root         16384 déc  5  2000 lost+found
  77291 drwxr-xr-x   4 root     root          4096 jan 24  2002 mnt
      1 dr-xr-xr-x  59 root     root             0 nov  4 19:52 proc
   4732 -rwx------   1 root     root           210 avr 10  2001 restore_passwd
  92738 drwx--x--x  16 root     root          4096 oct 23 18:21 root
 170017 drwxr-xr-x   5 root     root          4096 avr 30  2002 sbin
 190424 drwxr-xr-x   2 root     root          4096 oct 18  2001 swap
```

2.1.1. Gérer les fichiers (commandes usuelles)

Nom	Fonction	Exemple
cd	Changer de dossiers	`cd /home/toto`
	retour à son dossier de base (home directory)	`cd`
pwd	Connaître le dossier courant	`pwd`
rm	Effacer le fichier toto	`rm toto`
rm -r	Effacer le dossier *rep-perso* et son contenu	`rm -r rep-perso`
mkdir	Créer le dossier *essai-rep*	`mkdir essai-rep`
rmdir	Effacer le dossier vide *rep-essai*	`rmdir essai-rep`
cat	Afficher le contenu d'un fichier texte	`cat mon-fichier.txt`
tail	Affiche les 10 dernières lignes d'un fichier texte	`tail access.log`
head	Affiche les 10 premières lignes d'un fichier texte	`head access.log`
more	Idem **cat** mais page par page	`more essai.txt`
less	Idem **more** mais possibilité de défiler en arrière	`less essai.txt`
df	Liste et occupation des systèmes de fichiers	`df -v`
cp	Copier un fichier vers un autre	`cp toto1 toto2`
mv	Déplacer ou renommer un fichier	`mv toto titi`

find	rechercher un fichier sur certains critères
	Rechercher sur une partie du nom :
	Rechercher par type de fichier :
	Rechercher les *.exe et effectuer un chmod sur chaque :
		`find / -name tot*`
		`find / -type b`
		`find / -name *.exe -exec chmod 555 {} \;`

2.1.2. Gérer les droits d'accès

Pour accéder à Linux, il faut obligatoirement s'identifier par un nom d'utilisateur (login) et un mot de passe (password).
Les utilisateurs appartiennent également à un ou plusieurs groupes.
Les utilisateurs et les groupes sont gérés par l'administrateur (Nom d'utilisateur : **root**).
Sur certaines distributions (Debian, Ubuntu, RaspberryOS), l'utilisateur *root* est masqué et les commandes d'administration doivent être précédées de la commande **sudo**.

Les commandes shell pour gérer les utilisateurs et les groupes sont :
Création / modification : `useradd, usermod, groupadd, groups, passwd`
Suppression : `userdel, groupdel`

Lorsqu'un utilisateur crée un fichier, ce fichier comporte automatiquement certains réglages :
Il appartient à l'utilisateur qui l'a créé (UID User IDentifier = numéro associé au nom de l'utilisateur créateur) ;
Il appartient au groupe principal de l'utilisateur créateur (GID Group IDentifier) ;
Ses droits d'accès sont fixés de la façon suivante :
Le propriétaire (l'utilisateur créateur pour l'instant) peut lire et écrire ce fichier
Les membres du groupe auquel il appartient (celui du créateur pour l'instant) peuvent lire ce fichier
Les autres utilisateurs peuvent lire ce fichier.
Tous ces réglages peuvent être modifiés ultérieurement.

NB : Applications au langage C :
Les appels systèmes comme open, fopen, creat, … font appel à ces notions de droits d'accès par utilisateurs, groupes, ou autres.

Exemple : Visualisation des réglages avec la **commande ls** (ls –li dans l'exemple):

```
      14913 -rw-r--r--   1 yann     compta          82 nov  7 12:52 essai1.txt
```

`-` Type de fichier
`rw-` 1er bloc de 3 : r=lecture w=écriture pour le propriétaire (yann)
`r--` 2ème bloc de 3 : r=lecture pour le groupe propriétaire (compta)
`r--` 3ème bloc de 3 : r=lecture pour les autres.

`yann` 1er nom : celui de l'utilisateur propriétaire
`compta` 2ème nom : celui du groupe propriétaire

A. Signification des droits

Les droits (rwx) n'ont pas la même signification selon qu'ils concernent un fichier ou un dossier.

Pour un **fichier** :	r	droit de lire le contenu du fichier
	w	droit de modifier ou d'effacer le fichier
	x	droit d'exécuter le fichier (si c'est un programme …)
Pour un **dossier** :	r	droit de lister le contenu du dossier
	w	droit de créer ou d'effacer des fichiers dans le dossier
	x	droit d'aller dans le dossier (avec la commande `cd`)

B. Modifier des droits d'accès : CHMOD, CHOWN, CHGRP

Pour modifier les droits d'un fichier, il faut soit être l'utilisateur **root**, soit être propriétaire du fichier.

Modification des permissions : **`chmod options fichier`**

Les options de la commande chmod peuvent se présenter sous deux formes :

a) Sous forme chiffrée OCTAL:

Codage en octal, chaque lettre ayant une valeur numérique.

r = 4 w = 2 x = 1

Comme il y a 3 droits (r,w,x), et donc 8 combinaisons possibles, on utilise un chiffre en octal pour exprimer la combinaison souhaitée :

0	---	4	r--
1	--x	5	r-x
2	-w-	6	rw-
3	-wx	7	rwx

Exemple : `chmod 750 fichier` donne les droits `rwxr-x---`

Soit : `rwx` (4+2+1=7) : lecture/ écriture / exécution pour l'utilisateur propriétaire

`r-x` (4+0+1=5) : lecture / exécution pour le groupe propriétaire

`---` (0+0+0=0) : aucun droit pour les autres.

NB : L'option –R applique la modification dans les sous-dossiers.

Ex : `chmod -R 750 dossier`

b) Sous forme de lettres UGO:

La lettre **« u »** pour le propriétaire (user) ; « **g »** pour le groupe ; « **o »** pour les autres (others)

Exemple : `u+x` pour ajouter le droit **x** à l'utilisateur

`ugo+rwx` pour tout ajouter à tous

La commande **`chown`** change le nom de l'utilisateur propriétaire (ex : `chown jdupont toto`)

La commande **`chgrp`** change le nom du groupe propriétaire (ex : `chgrp elève toto`)

NB1 : La commande **`chown jdupont:elève toto`** change en même temps le propriétaire et le groupe propriétaire de toto.

NB2 : Une commande chmod, chown ou chgrp au niveau d'un dossier et avec l'option –R provoque le changement de façon récursive dans le dossier et ses sous-dossiers (ex : `chmod -R 777 rep_perso`)

C. Droits spéciaux :

Il existe un quatrième champ pour le paramétrage des droits.
Il s'utilise de la façon suivante :

`Chmod 1775 toto` : (1= stiky byte)

Résultat :

`-rwxrwxr-T  2 santoine profs  186 nov  7 13:29 dossier1`

Effet : seul le *santoine* aura le droit d'effacer ses fichiers dans le dossier *dossier1*

`Chmod 2755 toto` : (1= setgid : Set GroupIDentifier)

Résultat :

`-rwxr-sr-x  2 santoine profs  186 nov  7 13:29 toto`

Effet : Ce fichier est exécutable (droit x) et tout le monde peut l'exécuter. Mais le SETGID indique que celui qui l'utilise aura les mêmes droits que le groupe « prof » pendant l'exécution.

`Chmod 4755 toto` : (1= setuid : Set UserIDentifier)

Résultat :

`-rwsr-xr-x  2 santoine profs  186 nov  7 13:29 toto`

Effet : Ce fichier est exécutable (droit x) et tout le monde peut l'exécuter. Mais le SETUID indique que celui qui l'utilise aura les mêmes droits que l'utilisateur « santoine » pendant l'exécution.

Exemple d'utilisation :
Ces deux derniers réglages sont utilisés par exemple par la commande **`passwd`** (fichier `/usr/bin/passwd`) qui change le mot de passe d'un utilisateur. Les fichiers des mots de passe sont en accès interdits pour l'utilisateur normal.
Celui qui change sont propre mot de passe doit pourtant accéder à ces fichiers en mode écriture pour installer son nouveau mot de passe. Pour l'y autoriser le fichier exécutable **`/usr/bin/passwd`** appartient à `root` (l'administrateur) et le bit SETUID est positionné.
Ces réglages constituent des possibilités de failles dans la sécurité des accès.

D. UMASK

La command `umask` permet de connaître la valeur par défaut des droits d'accès lors de la création d'un fichier.
La valeur renvoyée par `umask` correspond au complément de la valeur octale utilisée par `chmod`.

Ex : 022 renvoyé par `umask` correspond à 755 utilisé par `chmod`.

Il est possible de modifier la valeur de UMASK en suivant cette même logique.

Ex : pour forcer `rw- r-- ---` à la création des fichiers, il faut taper préalablement `umask 137`.

2.2. Gérer plusieurs systèmes de fichiers

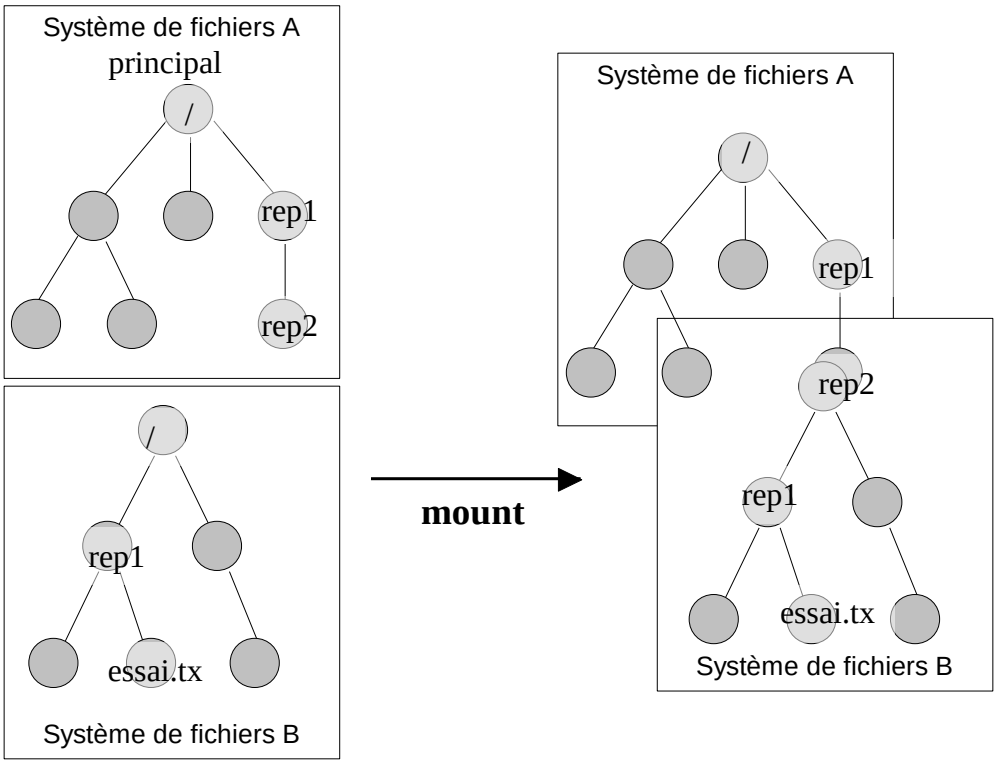
Chaque système de fichier est organisé en arborescence indépendante avec une racine.
Le système de fichier qui stocke le noyau Linux qui sert au démarrage est considéré comme le système de fichiers principal.

Les autres systèmes de fichiers peuvent être reliés (commande **`mount`**) à un dossier quelconque du système de fichier principal (ou d'un autre système de fichiers).

On passe ainsi d'un système de fichiers à un autre par une simple opération de changement de dossier, de façon transparente pour l'utilisateur.

Cette solution permet par exemple d'augmenter la capacité de stockage dans un dossier en le reliant à un disque de grande capacité.

Exemple :



Avant le **mount**, le système de fichiers B n'est pas accessible :
Opération de montage. Ex : `mount /dev/hdb1 /rep1/rep2`
où /dev/hdb correspond au fichier de périphérique qui gère la 2ème partition du 1ier disque IDE.

Après le **mount**, le fichier essai.txt est accessible par le chemin : `/rep1/rep2/rep1/essai.txt`

La commande **umount** rompt le lien logique entre les systèmes de fichiers (ex : `umount /dev/hdb1`)

La commande *mount* permet d'accéder à des systèmes de fichiers de types différents, comme des partitions Windows, ou des **disques réseaux** (Protocole NFS – *Network File System*).

Le fichier `/etc/fstab` contient la liste des systèmes de fichiers à « monter » au démarrage.

2.3. Utiliser les liens sur un fichier

A l'origine, le lien classique (commande `ln`) était simplement un moyen de donner plusieurs noms à un même fichier, à condition que tout se passe dans la même partition logique. Ce système n'est plus trop utilisé à cause de cette restriction qui devient gênante sur les systèmes multidisques actuels. On préfère utiliser le lien symbolique.

Le **lien symbolique** (commande `ln -s`) fonctionne comme les raccourcis de Microsoft Windows. Le lien symbolique est un fichier indépendant qui contient uniquement un pointage sur le fichier lié. Il occupe donc un numéro Inode différent sur le disque mais reste de taille très faible (quelques octets). Si le fichier pointé est effacé, le lien symbolique demeure mais il n'est plus utilisable.

Le système de démarrage des « services » Linux (dossier /etc/init.d) utilise largement ce procédé pour activer ou désactiver les « services[1] ».

Illustration du lien symbolique : un fichier différent, une taille différente, un type différent.

```
# ls -li
total 4
  14913 -rw-r--r--   1 root     root           82 nov  7 12:52 essai.txt

# ln -s essai.txt essai2.txt
# ls -li
total 8
  14913 -rw-r--r--   2 root     root           82 nov  7 12:52 essai.txt
  14912 lrwxrwxrwx   1 root     root            9 nov  7 12:55 essai2.txt -> essai.txt
```

[1] Un Service est une application exécutée en arrière plan sur un système d'exploitation multitâche.