Module n°2

ADMINISTRATION SYSTEME

Auteur : Labo-Linux
Version 0.1 – 10 novembre 2003
Nombre de pages : 126

Paris Academy of Computer Science SUPINFO Paris Ecole Supérieure d'Informatique

Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

Table des matières

Gestion des permissions sous Unix

1.1. Introduction

Nous allons voir dans cette partie comment s'effectue la gestion des permissions sur les fichiers d'un système Unix. Chaque fichier appartient à la fois à un utilisateur et à un groupe définis respectivement dans les fichiers */etc/passwd* et */etc/group*.

Ceci est la base de notre modèle de permissions. Pour connaître l'utilisateur et le groupe propriétaire d'un fichier, il suffit d'utiliser la commande **ls -l**. On obtient, par exemple, les informations sur les propriétaire du fichier */usr/bin/jed* de la façon suivante :

```
jms@tsunami doc $ ls -l /usr/bin/jed
-rwxr-xr-x 1 root wheel 200920 Oct 16 11:57 /usr/bin/jed
```

Dans l'exemple précédent, le binaire */usr/bin/jed* appartient à l'utilisateur **root** et au groupe **wheel**.

Le modèle de permissions Unix permet de régler trois niveaux d'autorisations pour chaque objet de notre système de fichier :

1. Autorisation de l'utilisateur propriétaire
2. Autorisation du groupe propriétaire
3. Autorisation des autres utilisateurs (non propriétaires)

Classe	Nomenclature	Signification
user	u	Propriétaire du fichier/répertoire.
group	g	Membres du groupe possédant le fichier/répertoire.
other	o	Tous les autres utilisateurs du système.

1.2. Identité dans le système

Avant de détailler le fonctionnement du modèle de permissions Unix, il est bon de connaître quelques commandes nous informant sur notre identité à l'intérieur du système.

Ainsi, nous allons voir comment récupérer notre nom d'utilisateur et les groupes auxquels nous appartenons grâce aux deux commandes **whoami** et **groups**.

1.2.1. Qui suis-je ?

A moins d'avoir récemment effectuer la commande **su**, l'identifiant utilisateur vous représentant est celui que vous avez utilisé lorsque vous vous êtes authentifié afin d'accéder aux ressources du système.

Pour connaître ce dernier, tapez **whoami** au prompt shell :

```
jms@tsunami doc $ whoami
jms
jms@tsunami doc $ su root
Password:
root@tsunami doc # whoami
root
```

1.2.2. A quels groupes appartient-on ?

Pour connaître l'ensemble des groupes auxquels l'utilisateur appartient, il suffit d'utiliser la commande **groups** :

```
jms@tsunami doc $ groups
users wheel audio
```

Dans l'exemple précédent, l'utilisateur *jms* est membre des groupes *users, wheel et audio*.

Si l'on souhaite connaître à quels groupes appartient un utilisateur, il suffit de spécifier son identifiant en paramètre à la commande **groups** :

```
jms@tsunami doc $ groups root
root: root bin daemon sys adm disk wheel floppy dialout tape video
```

1.2.3. Rappels sur la commande "ls -l"

Un dernier point avant d'aller plus loin est le rappel de la commande **ls -l**. Pour cela, observons son comportement :

```
jms@tsunami doc $ ls -l /usr/bin/jed
-rwxr-xr-x 1 root wheel 200920 Oct 16 11:57 /usr/bin/jed
```

Le premier champ **-rwxr-xr-x** est une représentation *symbolique* des droits ou permissions du fichier.

Le premier symbole de ce champ (-) défini le type de fichier, qui est dans ce cas ci un fichier standard. Parmi les types de fichiers on trouve également :

- 'd' : représente un répertoire (*directory*)
- 'l' : représente un lien symbolique (*symbolic link*)
- 'c' : représente un périphérique à caractère (*character special device*)
- 'b' : représente un périphérique à blocs (*block special device*)
- 'p' : fifo
- 's' : socket

1.3. Lecture des permissions d'un fichier

Reprenons la commande **ls -l** vue précédemment. Son résultat sur le fichier */usr/bin/jed* nous donne :

```
jms@tsunami doc $ ls -l /usr/bin/jed
-rwxr-xr-x 1 root wheel 200920 Oct 16 11:57 /usr/bin/jed
```

Nous avons vu également que le premier caractère du champ **-rwxr-xr-x** décrivait le type de notre fichier. Examinons à présent les 9 derniers caractères de ce champ.

Pour cela, nous allons découper ce champ en sous champs de 3 caractères chacun, représentants respectivement les permissions accordées sur le fichier au propriétaire de ce dernier, celles accordées au groupe propriétaire du ce fichier, celles attribuées aux autres utilisateurs de notre système. Ainsi, nous obtenons :

- **rwx** : permissions utilisateur (*u*)
- **r-x** : permissions groupe (*g*)
- **r-x** : permissions autres (*o*)

Chaque caractère **r,w, ou x** décrit les permissions accordées selon le tableau suivant :

Caractère symbolique	Permission accordée
r	Lecture du fichier
w	Modification du fichier (y compris supprimer !)
x	Exécution du fichier

En rassemblant ces informations, nous analysons que tout le monde peut exécuter le fichier */usr/bin/jed* mais que seul le propriétaire du fichier (root) peut le modifier.

Ceci signifie que les utilisateurs peuvent copier ce fichier mais que seul root peut le modifier ou le supprimer.

1.4. Changement des propriétaires d'un fichier

Dans un premier temps, il est de nature à se demander comment changer l'utilisateur ou le groupe propriétaire d'un fichier ou d'un objet de notre système de fichier.

Pour cela, nous utiliserons respectivement les commandes **chown** (*change owner*) et **chgrp** (*change group*).

Chacune de ces commandes peuvent être suivies d'une liste *n* d'arguments représentant des noms de fichiers.

Notez qu'il n'est pas autorisé à un autre utilisateur que root d'utiliser la commande **chown**.

La commande **chgrp** peut être utilisée quant à elle, par tous utilisateurs souhaitant modifier l'appartenance d'un fichier à un groupe. Ceci n'est possible que si l'utilisateur appartient lui-même au groupe propriétaire de ce fichier.

```
root@tsunami tmp # ls -l toto
-rw-r--r-- 1 root root 0 Nov 13 14:57 toto
root@tsunami tmp # chown jms toto
root@tsunami tmp # chgrp wheel toto
root@tsunami tmp # ls -l
total 0
-rw-r--r-- 1 jms wheel 0 Nov 13 14:57 toto
```

Il est également possible de modifier ces deux informations en une seule fois en utilisant la commande **chown** la façon suivante :

```
root@tsunami tmp # ls -l toto
-rw-r--r-- 1 root root 0 Nov 13 14:57 toto
root@tsunami tmp # chown jms.wheel toto
root@tsunami tmp # ls -l
total 0
-rw-r--r-- 1 jms wheel 0 Nov 13 14:57 toto
```

Enfin, pour des raisons pratique, il est possible de modifier récursivement tout un répertoire en attachant l'option **-R** aux commandes **chown** et **chgrp**. Par exemple :

```
root@tsunami tmp # chown -R jms.wheel /home/jms
```

1.5. La commande chmod

Les commandes **chown** et **chgrp** nous permettent de changer l'utilisateur et le groupe propriétaires d'un fichier, mais seule la commande **chmod** nous permet de changer les permissions **r**, **w** et **x** que nous voyons dans notre **ls -l**.

La commande **chmod** demande 2 arguments au moins :

```
chmod mode liste_de_fichiers
```

mode
: Permissions à appliquer

liste_de_fichiers
: Liste des fichiers concernés par la modification demandée par **chmod**

Par exemple :

```
root@tsunami tmp # chmod +x exemple.sh
```

Ainsi, notre *mode* est +x. Comme vous avez dû le comprendre, le mode **+x** demande à la commande **chmod** de rendre ce fichier exécutable à la fois par ses propriétaires mais également par l'ensemble des autres utilisateurs du système.

Si nous souhaitions annuler cette modification, nous effectuerions :

```
root@tsunami tmp # chmod -x exemple.sh
```

1.5.1. Composition symbolique de la commande chmod

Souvenez-vous que nous avions divisé notre champ de permissions **-rwxr-xr-x** en 3 triplets :

- *rwx* : représentant l'utilisateur propriétaire (**u**)
- *r-x* : représentant le groupe propriétaire (**g**)
- *r-x* : représentant les autres utilisateurs (**o**)

Il est ainsi possible de combiner chacun des caractères symboliques **u**, **g** et **o** aux lettres **r**, **w** et **x** afin de régler les permissions d'un fichier de manière plus fine.

En effet, il est commode de ne pouvoir changer qu'un ou deux triplets à la fois. Pour ce faire, spécifiez le caractère symbolique du triplet que vous souhaitez modifier avant le signe + ou – du mode de permission. On peut ainsi effectuer la commande suivante :

```
jms@tsunami tmp $ ls -l chmod.ex
---------- 1 jms users 0 Nov 13 15:36 chmod.ex

jms@tsunami tmp $ chmod u+rwx,g+rwx,o+rx chmod.ex

jms@tsunami tmp $ ls -l chmod.ex
-rwxrwxr-x 1 jms users 0 Nov 13 15:36 chmod.ex
```

La commande **chmod** de l'exemple précédent nous a permis d'ajouter les droits de lecture, modification et exécution aux propriétaires du fichier et les droits de lecture et d'exécution aux autres utilisateurs du système.

Pour retirer une partie de ces droits nous effectuerions par exemple :

```
jms@tsunami tmp $ ls -l chmod.ex
-rwxrwxr-x 1 jms users 0 Nov 13 15:36 chmod.ex

jms@tsunami tmp $ chmod go-x chmod.ex

jms@tsunami tmp $ ls -l chmod.ex
-rwxrw-r-- 1 jms users 0 Nov 13 15:36 chmod.ex
```

Enfin, sachez qu'il existe en supplément aux indicateurs + et - de la commande chmod, l'indicateur =.

Ce dernier indique à la commande **chmod** d'appliquer le mode de permission indiqué et pas un autre. Dans l'exemple suivant, nous indiquerons à chmod de mettre à zéro (-) tous les caractères symboliques représentant les permissions mis à part **r** et **x** et ceci pour tous les utilisateurs :

```
jms@tsunami tmp $ chmod =rx chmod.ex
```

```
jms@tsunami tmp $ ls -l chmod.ex
-r-xr-xr-x 1 jms users 0 Nov 13 15:36 chmod.ex
```

1.5.2. Composition numérique des modes de permissions

Nous n'avons utiliser jusqu'à présent que les modes "symboliques" pour spécifier les changements de permissions à effectuer par la commande **chmod**. Il existe cependant une autre façon, couramment utilisée, pour spécifier le mode de permissions à l'aide de 4 octets pouvant, comme il se doit, prendre les valeurs de 0 à 7.

Le premier de ces quatre octets va nous permettre de régler les attributs "spéciaux" du fichier dont nous reparlerons plus tard.

Les trois derniers octets représentent respectivement les droits associés à l'utilisateur (u), au groupe (g) et aux autres utilisateurs (o).

Ainsi, lorsque nous spécifions à la commande **chmod** le mode **0777** elle modifiera les permissions des triplets (u), (g) et (o).

Remarquez que l'octet des attributs spéciaux à été mis à **0**. Nous verrons plus tard quelle est sa signification.

0	0	0	0	0	0	0	0	0	0	0	0
SUID	SGID	Sticky	r(user)	w(user)	x(user)	r(grp)	w(grp)	x(grp)	r(oth)	w(oth)	x(oth)

Le tableau suivant représente les permissions (r), (w) et (x) et leur équivalent en bit sur un octet :

Permission	Valeur en bit
r	4
w	2
x	1
-	0

Ainsi, on obtient le mode **-rwxr-xr-x** en ajoutant les valeurs octales de chaque triplets les unes à la suite des autres :

```
-   := 0
rwx := 7
r-x := 5
r-x := 5

jms@tsunami tmp $ chmod 0755 chmod.ex
jms@tsunami tmp $ ls -l chmod.ex
-r-xr-xr-x 1 jms users 0 Nov 13 15:36 chmod.ex
```

1.6. Création de fichiers et umask

Lorsqu'un processus crée un nouveau fichier, il lui attribue des permissions par défaut.

Très souvent, le mode attribué est **0666**. Le fichier est donc créé en lecture / écriture pour tout le monde ce qui peut ne pas être assez restrictif.

Heureusement, une valeur appelée *umask* est consultée à chaque fois qu'un nouveau fichier doit être créé.

Le système d'exploitation utilise alors la valeur retournée par **umask** pour réduire les permissions accordées par défaut.

Pour connaître la valeur **umask** courante, il suffit de taper :

```
jms@tsunami doc $ umask
0022
```

Sous Linux, le umask par défaut est **0022**, ce qui permet aux autres utilisateurs de lire votre fichier mais pas de le modifier. En effet, 0666 - 0022 nous donne 0644.

1.6.1. Composition du mode d'umask

Le fonctionnement d'**umask** se résume par la phrase suivante. **umask** spécifie les permissions qui ne doivent pas être attribuées à un fichier nouvellement créé. Les permissions sont retirées selon le tableau suivant :

Caractère symbolique	Valeur en bit	Permission retirée
r	4	Lecture
w	2	Modification
x	1	Exécution
-	0	Aucune

Le mode d'**umask** se réalise de la même façon que le mode du **chmod**. Un bon exemple est de régler l'**umask** de l'utilisateur root au mode **0077**. Les autres utilisateurs du système n'auront alors aucunes permissions sur ces fichiers créés :

```
root@tsunami root # umask 0077
root@tsunami root # echo 1 > /tmp/umask.ex

root@tsunami root # ls -l /tmp/umask.ex
-rw------- 1 root root 2 Nov 13 17:25 /tmp/umask.ex
root@tsunami root # su jms
jms@tsunami root $ cat /tmp/umask.ex
cat: /tmp/umask.ex: Permission denied
```

1.7. Les attributs spéciaux - suid, sgid et sticky bit

1.7.1. Introduction

Lorsque vous vous connectez à un système d'exploitation de type Unix, un nouveau processus, votre shell par défaut, est démarré.

Vous saviez sans doute cela mais ce qui vous ignoriez peut-être est que ce nouveau processus a été démarré en utilisant votre identifiant utilisateur (**uid**) et votre groupe principal (**gid**). De cette sorte, votre shell par défaut ne peut accéder qu'aux fichiers dont vous êtes propriétaire.

En réalité, les utilisateurs que nous sommes, sont complètement dépendants d'autres programmes qui effectuent les opérations en leur nom. En lançant un programme, il hérite de votre **uid** et ne peut accèder qu'aux fichiers dont l'accès vous a été autorisé.

Ainsi, le fichier de mots de passe */etc/passwd* ne peut pas être modifié directement par un utilisateur du système autre que root, son propriétaire, puisque le symbole 'w' n'est activé que pour lui.

```
jms@tsunami doc $ ls -l /etc/passwd
-rw-r--r-- 1 root root 1731 Oct 28 13:06 /etc/passwd
```

Cependant, un utilisateur classique du système doit pouvoir modifier ce fichier (au moins de manière indirecte) quand il veut changer son mot de passe par exemple. Comment tout cela fonctionne-t-il ?

1.7.2. Un petit peu de théorie...

Heureusement pour nous, le modèle de permissions des principaux systèmes UNIX permet de régler deux attributs spéciaux, le **suid**, le bit utilisateur et le **sgid**, le bit du groupe.

Bit identifiant l'utilisateur
Lorsqu'un binaire possède le mode **suid**, il est démarré et s'exécute dans l'environnement du propriétaire de ce programme, plutôt que dans l'environnement de l'utilisateur qui a demandé l'exécution de ce dernier.

Si nous regardons de plus près le binaire /**usr/bin/passwd,** nous pouvons voir qu'il appartient à root :

```
jms@tsunami doc $ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passwd
```

Remarquez que le **x** du triplet des permissions de l'utilisateur a été remplacé par un **s**.

Ceci indique que pour ce programme particulier, le bit d'exécution et le bit suid sont activés. A cause de cela, lorsqu'un utilisateur exécute la commande **passwd**, cette dernière s'exécutera dans l'environnement de root, le propriétaire du binaire.

Comme **passwd** s'exécute avec les droits de root, le fichier */etc/passwd* pourra être modifié sans aucun problème.

On règle les bits **suid** et **sgid** à l'aide de la commande **chmod**. Voici un exemple plaçant le bit **suid** sur un fichier :

```
root@tsunami root # chmod u+s /usr/bin/passwd
root@tsunami root # ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passwd
```

Bit identifiant le groupe
Le bit **sgid** fonctionne de façon similaire.
Il permet, lors de l'exécution d'un fichier binaire, de spécifier que l'environnement dans lequel il sera lancé sera celui de son groupe propriétaire plutôt que celui de l'utilisateur qui demande son exécution.

D'une façon similaire, on ajoute ou on retire le bit **sgid** d'un fichier :

```
root@tsunami root # ls -l /usr/bin/passwdgrp
-rwxr-sr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passgrp

root@tsunami root # chmod g-s /usr/bin/passwdgrp

root@tsunami root # ls -l /usr/bin/passwdgrp
-rwxr-xr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passgrp
```

Remarque : Place des suid et sgid
Le bit **suid** et le bit **sgid** occupent la même place que le bit **x** (pour le **suid** au niveau des permissions du propriétaire du fichier et le **sgid** au niveau du groupe propriétaire) dans le retour d'un **ls -l**.

Si la permission **x** est aussi mise en place sur le fichier, il se transformera en **s** minuscule. Si par contre, la permission **x** n'est pas activée, ce dernier se transformera en **S** majuscule.

```
tsunami root # ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passwd

root@tsunami root # chmod 4644 /usr/bin/passwd

root@tsunami root # ls -l /usr/bin/passwd
-rwSr--r-- 1 root root 26712 Sep 12 06:57 /usr/bin/passwd
```

De plus, bien que les bits **suid** et **sgid** puissent être pratiques dans certains cas, une mauvaise utilisation de ces attributs spéciaux peut conduire à de graves problèmes de sécurité.

Il est ainsi recommander de placer ces attributs spéciaux à un minimum de fichiers.

1.7.3. Les permissions et les répertoires

Jusqu'à présent nous avons présenté le modèle de permissions Unix par rapport aux fichiers contenus sur un tel système. Il est temps de nous tourner vers les permissions de répertoires.

Ces derniers utilisent les mêmes caractères symboliques **r**, **w** et **x** que les fichiers mais leur interprétation est quelque peu différente comme nous l'explique le tableau suivant :

Caractère symbolique	Valeur en bit	Autorisation
r	4	Lister, lire le répertoire
w	2	Créer un fichier dans ce répertoire
x	1	Accès accordé au répertoire
-	0	Aucune

Notez que sans l'autorisation **x** les fichiers présents dans le répertoire ne sont pas accessibles. Sans l'attribut **r**, il est impossible de lister le contenu d'un répertoire, mais vous pouvez toujours accéder à un fichier contenu dans cette arborescence du moment que vous connaissez le chemin complet de ce fichier :

```
root@tsunami doc # chmod 711 /etc

jms@tsunami doc $ ls /etc
ls: /etc: Permission denied

jms@tsunami doc $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
sync:x:5:0:sync:/sbin:/bin/sync
```

Bit sgid et répertoires

Placer un bit **sgid** sur un répertoire permet de faire appartenir automatiquement tout fichier créé dans ce répertoire au groupe propriétaire de ce dernier.

Cette composante sera essentielle lorsque vous voudrez créer un répertoire partagé par un groupe de personnes appartenant au même groupe système.

```
root@tsunami root # mkdir /home/users
root@tsunami root # chgrp users /home/users/
root@tsunami root # chmod g+s /home/users/
root@tsunami root # ls -l /home
drwxr-sr-x 2 root users 4096 Nov 13 20:47 users
```

Dès lors, chaque utilisateur membre du groupe **users** peut créer des fichiers à l'intérieur du répertoire */home/users* qui seront automatiquement affilié au groupe propriétaire du répertoire.

En fonction de la valeur **umask** de l'utilisateur, les fichiers créés dans ce répertoire ne seront peut-être pas lisibles, modifiables ou exécutables par un autre utilisateur que le propriétaire.

Bit Sticky et répertoires

Par défaut, tout le monde peut renommer ou supprimer un fichier qui se trouve à l'intérieur d'un répertoire ou l'écriture est autorisée.

Ce fonctionnement peut-être bon lorsqu'il s'agit de répertoire utilisé par un seul et unique utilisateur.

Par contre, lorsque plusieurs utilisateurs se servent d'un même répertoire pour y stocker des données, comme par exemple */tmp* ou */var/tmp*, ce fonctionnement se révèle inadapté.

En effet, puisque n'importe qui peut avoir un accès en écriture à l'intérieur de ce répertoire, tout le monde peut effacer ou renommer les fichiers de tout le monde, même sans en être son propriétaire.
Il serait alors difficile d'utiliser ce genre de répertoire pour stocker des données, puisque n'importe qui pourrait les effacer en tapant :

```
rm -rf /tmp
```

Encore une fois, un attribut spécial est là pour régler notre problème. En plaçant le bit **sticky** sur un répertoire, la seule personne autorisée à effacer un fichier contenu à l'intérieur du répertoire est son propriétaire. La commande utilisée pour régler ce bit est **chmod** à laquelle on passe le mode **+t** de la façon suivante :

```
root@tsunami root # chmod +t /tmp
root@tsunami root # ls -l /
drwxrwxrwt 10 root root 4096 Nov 13 20:24 tmp
```

Exemple :

```
florent@tsunami doc $ touch /tmp/test
florent@tsunami doc $ su jms
Password:
jms@tsunami doc $ rm /tmp/test
rm: remove write-protected file '/tmp/test'? y
rm: cannot unlink '/tmp/test': Operation not permitted
```

Par défaut toutes les distributions Linux place un bit **sticky** sur le répertoire */tmp* mais vous vous rendrez compte que ce bit est utile dans bien des cas.

1.8. Composition numériques des permissions spéciales

L'ensemble des autorisations spéciales est stocké sur un octet et prend les valeurs 4, 2, 1 et 0 comme selon le tableau ci-dessous :

Caractère symbolique	Permission accordée
suid	4
sgid	2
sticky	1
RAZ	0

Ainsi, on peut régler un sticky bit sur le répertoire */home/users* de la façon suivante :

```
root@tsunami root # chmod 1777 /home/users
root@tsunami root # ls -l /home
drwxrwxrwt 1 root root 26712 Sep 12 06:57 users
```

Ou encore composé une permissions spéciale sur un binaire pour qu'il s'exécute en s'appuyant sur l'environnement de l'utilisateur et du groupe propriétaire du fichier :

```
root@tsunami root # chmod 6755 /usr/bin/passwd
root@tsunami root # ls -l /usr/bin/passwd
-rwsr-sr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passwd
```

Et comme vous vous en doutiez, les attributs spéciaux peuvent être retirés en plaçant la valeur octale à 0 :

```
root@tsunami root # chmod 0755 /usr/bin/passwd
root@tsunami root # ls -l /usr/bin/passwd
-rwxr-xr-x 1 root root 26712 Sep 12 06:57 /usr/bin/passwd
```

2. Le système de fichier sous Linux

2.1. Définition

Pour UNIX, un système de fichiers est un périphérique formaté en vue de stocker des fichiers. Il se trouve aussi bien sur un disque dur que sur une disquette, un CD-ROM ou tout autre support permettant un accès aléatoire (à la différence d'une bande magnétique => accès séquentiel).
Le format exact et les moyens par lesquels les fichiers sont enregistrés sont sans importance, le système offre une interface commune à tous les types de systèmes de fichiers qu'il reconnaît.

2.2. L'organisation du système de fichier

Sur quelqu'Unix (sauf peut-être Darwin) que ce soit, vous ne devriez pas avoir de problèmes pour vous retrouver, puisque utilisant sensiblement la même arborescence de fichiers.

La plupart du temps, les Unix essaient de se conformer au FHS ("File Hierarchy Standard"). Le principe général est le regroupement par type de fichier ou par type d'utilisation au sein d'un même répertoire.

Par exemple, les fichiers d'aide de type man seront regroupés dans les répertoires "/usr/man", "/usr/local/man". Voici comment s'organise un système de fichiers de type Unix en général :

/
- bin/ commandes de bases pour tous les utilisateurs
- boot/ fichiers concernant le bootloader
- dev/ périphériques
- etc/ configuration générale du système
- home/ répertoire d'arrivée des utilisateurs
- lib/ bibliothèques
- mnt/ point de montage
- root/ répertoire d'arrivée pour root
- sbin/ éxecutables systèmes
- opt/ applications supplémentaires
- proc/ fichiers qui permettent de modifier certaines propriétés du noyau
- usr/ programmes, librairies utilisés par l'utilisateur
  - bin/ exécutables spécifiques aux utilisateurs
  - sbin/ exécutables systèmes indispensables
  - lib/ bibliothèques indispensables au développement
  - share/ données indépendantes de l'architecture
  - include/ contient les fichiers headers
  - X11R6/ tout ce qui concerne X Window
  - bin/ arborescence locale
- var/ les fichiers des services
  - lock/ fichiers bloqués
  - log/ fichiers de log
  - cache/ cache d'applications
  - lib/ informations concernant l'etat de certains exécutables
  - mail/ boîtes au lettres des utilisateurs
  - opt/ données des variables pour /opt
  - run/ données sur des exécutables en cours d'exécution
  - spool/ files d'attentes
  - tmp/ données temporaires sauvegardées même après un reboot

Suivant les UNIX sur lesquels nous pouvons nous retrouver, certains autres répertoires peuvent apparaître, comme "/proc" qui contient quantités d'informations sur le noyau GNU/Linux remises à jour en temps réel.

Dans les chapitres qui vont suivre, nous allons nous déplacer dans les différentes parties de notre arborescence mettant ainsi le doigt sur l'utilité de chacune d'elle.

2.3. Les différents systèmes de fichiers

Il existe sous linux différents systèmes de fichiers :
Chaque type possède ses propres caractéristiques et limites. Le système de fichier MS-DOS limite par exemple les noms de fichier à huit caractères. Linux gère également les partitions VFAT et NTFS (lecture seule) qui permet de monter des partitions Windows.
Actuellement les systèmes de fichiers les plus utilisés sont : l'EXT3, le reiserfs et XFS.

2.3.1. Ext2

C'est le système de fichiers de base. Il est assez performant et peu sensible à la fragmentation, mais en cas de redémarrage brutal du système (coupure de courant) on peut perdre des données car les écritures sur le disque ne sont pas faites immédiatement (pour des raisons de performance).

De plus au redémarrage il faut utiliser un utilitaire qui vérifie l'intégrité du disque (FSCK), ce processus de vérification peut-être assez long.

2.3.2. Ext3

C'est un système de fichier journalisé, et qui est maintenant couramment utilisé par défaut sur les principales distributions de Linux (à la place d'ext2).

La journalisation consiste à enregistrer les modifications dans un journal donc en cas de problème il suffit de refaire le journal pour réparer la partition.

Un exemple simplet serait de dire que lorsque que vous copiez un fichier A dans un répertoire B, le système va écrire dans un journal la transaction avant de commencer la copie. Il effacera la transaction si tout s'est correctement déroulé.

C'est une évolution d'Ext2 avec la journalisation. L'avantage de ce système est que l'on peut convertir directement une partition ext2 vers ext3 et inversement, il est même possible d'effectuer la conversion sur des partitions montées. L'opération ne prenant que quelques secondes.
De plus Ext3 gère la journalisation des meta-données et la partition peut être montée en tant qu'ext2 si besoin est. Le défaut est que l'ext3 a les mêmes limitations que l'ext2.

2.3.3. Reiserfs

C'est le système de fichiers dont on parle le plus en ce moment. Il est intégré au noyau 2.4. Il utilise un b-arbre pour le stockage des données. Son avantage est d'être disponible en standard à l'installation de certaines distributions et surtout d'être très performant.

Ses défauts sont une mauvaise gestion des exportations NFS (partage en réseau) et l'absence de support pour les quotas. On peut considérer que pour une utilisation normale c'est un bon choix.

2.3.4. XFS

C'est le système de fichiers utilisés sur les stations SGI depuis 1993. Il a été récemment porté sous linux. Pour pouvoir l'utiliser il faut patcher le noyau. C'est un système basé sur des arbres b+ avec une allocation dynamique des inodes. De plus c'est un système 64 bits ce qui permet de gérer des fichiers de 9 Exaoctets.

Xfs étant utilisé sur les serveurs SGI, la gestion SMP est bien plus performante que pour d'autres systèmes de fichiers. Il est aussi possible d'associer une description à chaque fichier et après de faire des recherches sur ces données. Xfs est peut être à l'heure actuelle le plus abouti et le plus fiable des systèmes de fichiers grâce a sa longue carrière sur les stations SGI.

2.3.5. JFS

C'est le système de fichiers créé par IBM pour ses serveurs. La version actuellement portée sous Linux n'est pas encore assez fiable pour une utilisation avec des données importantes.

2.4. Superblock

Lorsqu'un système de fichier ext2 est crée, le metadata spécifique de fichiers est stocké dans un superblock. Cette donnée est essentielle pour l'opération du système de fichiers.

C'est pourquoi des copies du système de fichier superblock sont créées (par exemple tous les 8192 blocs pour les petits systèmes de fichiers). La commande dumpe2fs peut être utilisée pour afficher une grande partie des données stockées dans le superblock d'un système de fichiers.

2.5. Les liens

Les liens sont utiles pour faire apparaître un même fichier dans plusieurs répertoires, ou sous des noms différents. Ils évitent les duplications et assurent la cohérence des mises à jour.

On distingue en fait deux sortes de liens :

2.5.1. Les liens en durs (hardlinks)

Les liens durs associent deux ou plusieurs fichiers à un même espace sur le disque, les deux fichiers restant indépendants.

```
[mathieu@escaflowne mathieu]$ ln linux.txt ./README.txt
```

Le fichier README.txt est créé dans le répertoire ./. On peut constater que ces 2 fichiers ont la même taille.

```
[mathieu@escaflowne mathieu]$ ls -l
-rw-rw-r-- 2 student student 141 nov 16 10:12 linux.txt
-rw-rw-r-- 2 student student 141 nov 16 10:12 README.txt
```

Au niveau gestion ils sont indépendants, tout en partageant le même espace disque et donc le même inode. Toute modification de l'un, modifie l'autre !

Mais la suppression de l'un, casse le lien, mais ne supprime pas physiquement l'autre.

2.5.2. Les liens symboliques (sym links)

```
[mathieu@escaflowne mathieu]$ ln -s linux.txt ./README-SYBOL.txt
```

Le fichier README-SYBOL.txt fait référence à linux.txt on peut le voir avec *ls –l* :

```
[mathieu@escaflowne mathieu]$ ls -l
-rw-rw-r-- 2 student student 141 nov 16 10:12 linux.txt
lrwxrwxrwx 1 student student 9   nov 16 10:16 README-SYBOL.txt -> linux.txt
-rw-rw-r-- 2 student student 141 nov 16 10:12 README.txt
```

On peut noter que le lien symbolique n'a pas la même taille que le fichier source. La suppression du fichier source entraînera un changement de comportement du fichier lien qui sera considéré comme "cassé" ("broken").

3. Gestion des disques

Que vous souhaitiez utiliser une partition fat32, ntfs, hpfs, iso9660 ou bien d'autres, il y a de grandes chances pour que ce soit supporté. De même, alors que vous estimez que votre espace disque n'est plus suffisant, il vous suffira de "monter" une partition de votre nouveau disque dur et le "brancher" à n'importe quelle partie de votre arborescence.

Une fois la petite tâche administrative effectuée, votre arborescence semblera être la même que celle que vous aviez auparavant, l'espace disque en plus.

3.1. Partitionnement

Les outils de partitionnement seront différents d'un Unix à l'autre. Dans l'état actuel de la documentation, nous allons nous intéresser au monde Linux.

Pour cette tâche, l'utilitaire sous GNU/Linux s'appelle "fdisk" auquel nous allons passer en paramètre le chemin vers le périphérique à partitionner.

3.1.1. Organisation de quelques périphériques dans le "/dev/"

Nous avons vu précédemment que nous avions dans le répertoire "/dev/" les entrées de divers périphériques. Ces entrées sont caractérisées comme des fichiers. Ils peuvent d'ailleurs très bien se comporter comme tels.

Sur certains d'entre eux, essayer de faire un cat ou tout simplement de rediriger la sortie d'un fichier ou d'un programme vers eux, vous obtiendrez un certain nombre de choses inattendues.

Un test probant consiste à rediriger un fichier binaire vers "/dev/audio", vos oreilles devraient apprécier.

Les disques sont différenciés suivant qu'ils sont IDE ou SCSI. Sur la plupart des installations de votre GNU/Linux, les périphériques IDE seront caractérisés par des entrées du type "hdα" où α correspond à l'emplacement du périphérique sur la chaîne IDE qui sera définit par une lettre.

Ainsi, si nous souhaitons accéder au périphérique "master" du premier contrôleur IDE, le périphérique se nommera "hda".
De même, si nous souhaitons accéder à la seconde partition de ce même disque, il suffira d'ajouter le numéro de la partition au nom du périphérique (pour l'exemple : "hda2").

Pour les périphériques SCSI, la démarche reste la même, hors mis que les entrées seront du type "sd*alpha*" pour le disque, "scd*num*" pour les lecteurs cd…

Même si cette notation est la plus fréquemment rencontrée, une autre approche du répertoire "/dev/" se développe petit à petit, c'est celle que propose le "devfs".

Le principe consiste à rendre plus lisible le contenu du "/dev/" avec des répertoires symbolisant par exemple que nous souhaitons accéder à un périphérique SCSI, sur telle chaîne, tel ID. . .Voici ce que pourrait donner l´accès à un lecteur cd scsi :

```
/dev/scsi/host0/bus0/target4/lun0/cd
```

C'est quand même bien plus compréhensible. Cette approche est du même type que celle qu'on peut retrouver sur IRIX (Unix de Silicon Graphics) par exemple.

Si vous ne savez pas comment retrouver l'appellation de certains périphériques, l'utilitaire "dmesg" devrait vous venir en aide.

Voici un exemple de comment peuvent se représenter les périphériques via "dmesg" qui reprend les messages donnés par le noyau lors du boot où l'on peut distinguer le nom qu'a affecté le noyau à deux disques SCSI et leurs partitions :

```
...
SCSI device sda: 8925000 512-byte hdwr sectors (4570 MB)
/dev/scsi/host0/bus0/target3/lun0: p1
SCSI device sdb: 17836668 512-byte hdwr sectors (9132 MB)
/dev/scsi/host0/bus0/target6/lun0: p1 p2 p3 < p5 p6 >
...
```

3.1.2. Edition de la table des partitions

Une fois que nous avons cerné le périphérique à éditer, nous allons pouvoir éditer la table des partitions.

Sur un système GNU/Linux, vous allez être contraint de créer au moins deux partitions. La première va contenir le "/", et la seconde va être le "swap".

Pour le reste, si vous souhaitez avoir une partition pour "/usr", "/tmp" ou autre, libre à vous, et c'est même plutôt conseillé pour un soucis d'intégrité du système.

Nous verrons comment procéder pour associer par exemple "/dev/hda3" avec "/usr". Imaginons que nous souhaitons partitionner notre "master" sur le premier contrôleur IDE. Nous voulons disposer d'une partition de "swap" de 128 Mo et le reste du disque pour le répertoire root. Voyons par la pratique:

```
$ fdisk /dev/hda
The number of cylinders for this disk is set to 1826.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 1826 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
```

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p

Partition number (1-4): 1
First cylinder (1-1826, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1826, default 1826): +128M

Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 1826 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 17 136521 83 Linux

Command (m for help): n
Command action
e extended
p primary partition (1-4)
e

Partition number (1-4): 2
First cylinder (18-1826, default 18):
Using default value 18
Last cylinder or +size or +sizeM or +sizeK (18-1826, default 1826):
Using default value 1826

Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 1826 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 17 136521 83 Linux
/dev/hda2 18 1826 14530792+ 5 Extended

Command (m for help): n
Command action
l logical (5 or over)
p primary partition (1-4)
l
First cylinder (18-1826, default 18):
Using default value 18
Last cylinder or +size or +sizeM or +sizeK (18-1826, default 1826):
Using default value 1826

Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 1826 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 17 136521 83 Linux
/dev/hda2 18 1826 14530792+ 5 Extended
/dev/hda5 18 1826 14530761 83 Linux

Command (m for help): t
Partition number (1-5): 1
Hex code (type L to list codes): L
0 Empty 18 AST Windows swa 61 SpeedStor a6 OpenBSD
1 FAT12 1b Hidden Win95 FA 63 GNU HURD or Sys a7 NeXTSTEP
2 XENIX root 1c Hidden Win95 FA 64 Novell Netware b7 BSDI fs
3 XENIX usr 1e Hidden Win95 FA 65 Novell Netware b8 BSDI swap
```

```
4 FAT16 <32M 24 NEC DOS 70 DiskSecure Mult c1 DRDOS/sec (FAT-
5 Extended 39 Plan 9 75 PC/IX c4 DRDOS/sec (FAT-
6 FAT16 3c PartitionMagic 80 Old Minix c6 DRDOS/sec (FAT-
7 HPFS/NTFS 40 Venix 80286 81 Minix / old Lin c7 Syrinx
8 AIX 41 PPC PReP Boot 82 Linux swap da Non-FS data
9 AIX bootable 42 SFS 83 Linux db CP/M / CTOS / .
a OS/2 Boot Manag 4d QNX4.x 84 OS/2 hidden C: e1 DOS access
b Win95 FAT32 4e QNX4.x 2nd part 85 Linux extended e3 DOS R/O
c Win95 FAT32 (LB 4f QNX4.x 3rd part 86 NTFS volume set e4 SpeedStor
e Win95 FAT16 (LB 50 OnTrack DM 87 NTFS volume set eb BeOS fs
f Win95 Ext'd (LB 51 OnTrack DM6 Aux 8e Linux LVM f1 SpeedStor

10 OPUS 52 CP/M 93 Amoeba f4 SpeedStor
11 Hidden FAT12 53 OnTrack DM6 Aux 94 Amoeba BBT f2 DOS secondary
12 Compaq diagnost 54 OnTrackDM6 9f BSD/OS fd Linux raid auto
14 Hidden FAT16 <3 55 EZ-Drive a0 IBM Thinkpad hi fe LANstep
16 Hidden FAT16 56 Golden Bow a5 BSD/386 ff BBT
17 Hidden HPFS/NTF 5c Priam Edisk

Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap)

Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 1826 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Start End Blocks Id System
/dev/hda1 1 17 136521 82 Linux swap
/dev/hda2 18 1826 14530792+ 5 Extended
/dev/hda5 18 1826 14530761 83 Linux

Command (m for help): w
Syncing disks
```

Il est clair, qu'hormis son nom, "fdisk" est très loin de son homologue sous DOS.

Une remarque concernant les partitions bootables ou non : si vous souhaitez "booter" sur une partition plutôt qu'une autre, lors de l'invite de commande de "fdisk", il est possible de taper la commande "a" qui va vous demander sur quelle partition mettre le "flag" indiquant où vous souhaitez démarrer.

Autre remarque, l'identifiant affecté aux partitions suit la notation suivante :

- Les partitions primaires auront un identifiant allant de 1 à 4 compris
- Les partitions logiques auront des identifiants supérieurs ou égaux à 5

3.2. Créer un système de fichiers

La commande **mkfs** (MaKe Files System) est destinée à créer des systèmes de fichiers. Cette opération est analogue au "**formatage haut niveau**"de disquettes et de disques durs sur d'autres systèmes d'exploitation.
Chaque type de système de fichier possède sa propre commande mkfs : mkfs.ext2, mkfs.ext3...
En réalité, mkfs n'est qu'un frontal qui appelle le programme adéquat, selon le type de système de fichier choisi.

La création d'un système de fichier au moyen de **mkfs** s'obtient grâce à la syntaxe suivante :

```
# mkfs -t <type> <periphérique> [<blocs]
```

Où <type> désigne le type de système de fichiers à créer, <périphérique> indique le périphérique sur lequel l'opération doit s'effectuer (par exemple /dev/fd0 pour une disquette) et <blocs> spécifie la taille finale exprimée en blocs de 1024 octets. Sauf dans des cas très particuliers, le nombre de blocs est trouvé automatiquement par la commande de création du système de fichiers.

Ainsi pour créer un système de fichier ext2 sur la 3ème partition du premier disque dur IDE on écrira :

```
# mkfs -t ext2 /dev/hda3
```

Ce qui est équivalent à :

```
# mkfs.ext2 /dev/hda3
```

L'utilitaire **mke2fs** permet aussi de créer un système de fichiers ext2.

Pour créer un système de fichiers ext3 (ext2 journalisé) :

```
# mke2fs -j /dev/hda3
```

Lors de la création de système de fichiers sur disquettes, il est en général préférable d'effectuer un formatage de bas niveau (indispensable s'il s'agit de disquette vierge) grâce à l'outil **fdformat** :

```
# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440kB.
Formating ... done
Verifying ... done
```

L'option **-n** permet de s'affranchir de l'étape de vérification (ce qui n'est pas recommandé).

3.3. La swap

3.3.1. Définition

L'espace de swap est un terme général désignant l'espace disque utilisé pour augmenter la quantité apparente de mémoire disponible dans la machine.
Sous Linux cet espace permet d'implémenter la pagination sur disque, grâce à laquelle les pages de mémoires (4096 octets) sont écrites sur le disque lorsque la mémoire physique (RAM) devient insuffisante. L'espace swap n'est PAS une alternative à un manque de mémoire physique, en effet les accès des disques durs sont beaucoup plus élevés que les accès à la mémoire physique.

3.3.2. Les différentes formes de swap

Linux reconnaît deux formes de zones de swap : une partition dédiée du disque dur, ou bien un quelconque fichier. Linux supporte jusqu'à 16 zones de swap (peu importe la taille).Quoi qu'il en soit, il est préférable de créer deux petites partitions de swap sur des disque différents qu'une seule.

Une zone de swap sur partition accroît bien souvent les performances du système, car elle garantit des blocs contigus sur le disque. Les fichiers de swap sont parfois éclatés en blocs non contigus, ce qui augmente considérablement les temps d'accès, ils sont en général créés qu'occasionnellement en plus de la partition afin de répondre à un besoin momentané. Les fichiers swap sont un excellent moyen d'obtenir de la mémoire à la demande.

La commande free permet de connaître l'état des zones de swap ainsi que la mémoire physique:

```
#free
             total       used       free     shared    buffer cached
Mem:        511488     355872       4596          0      5864 374824
-/+ buffers/cache:     129604     385284
Swap:       521600          0     521600
```

Toutes les valeurs sont exprimées en bloc de 1024 octets. Ce système dispose en l'occurrence de 511488 blocs (environ 514 Mo) de mémoire physique, dont 455872 (environ 355 Mo) sont utilisée. A savoir que la quantité de RAM (ligne Mem) qui apparaît dans la colonne "total" est toujours sous-évaluée car la mémoire occupée par le noyau lui-même n'est pas prise en compte.

La colonne "shared" indique la **quantité de mémoire physique partagée** entre plusieurs processus en cours.

La colonne "buffer" montre la **quantité de mémoire utilisée** par le cache disque du noyau.

La troisième ligne concerne **la zone de swap**. Dans cet exemple, on observe que la swap n'est pas utilisée (colonne used). Linux n'a recours à la zone de swap qu'en dernier ressort, c'est-à-dire lorsque la RAM vient à manquer.

3.3.3. Créer une zone de swap

La première étape consiste à créer une partition ou un fichier destiné à accueillir cet espace swap. Pour une partition, il faudra utiliser **fdisk**, pour créer un fichier swap, il faut faire appel à la commande **dd.**

Par exemple pour créer un fichier swap de 8Mo :

```
# dd if=/dev/zero of=/swap bs=1024 count=8192
```

Cette commande écrit 8192 blocs de 1024 (soit 8mo) de données issues de **/dev/zero** dans le fichier **/swap**.

Il faut ensuite invoquer la commande **mkswap** afin de formater cette zone :

```
# mkswap -c /swap
```

Si il s'agit d'une partition (par exemple /dev/hda4) :

```
# mkswap -c /dev/hda4
```

L'option –c permet de vérifier/réparer les blocs défectueux (recommandé)

Si il s'agit d'un fichier, il est préférable de redéfinir les permissions :

```
# chmod 0600 /swap
```

3.3.4. Activer désactiver une zone de swap

La commande **swapon** permet de mettre en service la zone de swap :

```
# swapon /swap
```

La zone de swap s'ajoute alors aussitôt au reste de la swap.

Dans le cas d'une partition :

```
# swapon /dev/hda4
```

La commande **swapoff** permet de désactiver une zone de swap :

```
# swapoff /swap

ou bien  :

# swapoff /dev/hda4
```

Note : Attention à ne pas effacer un fichier swap (avec rm) avant de l'avoir désactivé avec **swapoff** !!!

3.4. Vérification des systèmes de fichiers

Il est parfois nécessaire de vérifier l'intégrité des systèmes de fichiers, puis de les remettre en état si des erreurs ou des pertes de données sont constatées. De telles erreurs résultent en général d'arrêts intempestifs de la machine. Le programme fsck (file System CheckK) est destiné à examiner les systèmes de fichier et à corriger les problèmes éventuels. Tout comme mkfs, fsck est un frontal appelant une commande spécifique à chaque type, nommée fsck.<type> (fsck.ext2 par exemple).

Voici comment l'utiliser :

```
# fsck -t <type> <périphérique>
```

Exemple :

```
# fsck -t ext3 /dev/hda1
```

3.5. Association périphérique / point de montage

Une fois nos partitions "prêtes à l'emploi" parce que nous les avons conçu ou tout simplement parce qu'elles existent déjà, il va falloir que l'on puisse y accéder quelque part dans notre arborescence.

Cela peut se faire n'importe où.

Par convention, on dispose dans le répertoire "/mnt/" les répertoires qui vont être associés aux périphériques n'intervenants pas dans l'arborescence "principale" tels que les lecteurs cd, disquette, zip, partitions Windows…

Une fois que nous avons définit l'endroit qui servira pour accéder à une partition ou périphérique, et que nous l'avons créé par la simple utilisation de "mkdir", il va falloir faire l'association périphérique et cette entrée préparée que l'on appelle "point de montage".

Ce processus d'association est naturellement appelé, "montage".

3.5.1. Utilisation de la commande "mount"

La commande "mount", comme son nom l'indique va permettre de monter des partitions, et "umount" les démonter. La syntaxe générale est la suivante :

```
mount -t type_de_fs /dev/peripherique_ou_partition point_de_montage
```

Appelée seule, "mount" affiche les montages actuels avec leur type de fichier respectifs et les options entre parenthèses. Ces informations peuvent également être trouvées dans le fichier "/etc/mtab".

Il est possible, en plus de l'argument "-t", de lui en passer d'autres, dont "-o" qui permet de définir les options.

Elles peuvent par exemple permettre à tous le monde de monter la partition, ou uniquement à certaines personnes ou bien aussi que la partition soit montée uniquement en lecture ou en lecture/écriture.

Exemple de montage d'un lecteur CD :

```
$ mount
```

```
/dev/hda5 on / type ext2 (rw)
none on /proc type proc (rw)

$ ls /mnt/cd
$ mount -o ro -t iso9660 /dev/hdc /mnt/cd

$ ls /mnt/cd
BOB SINCLAR - Champs Elysees
Dimitri from Paris - A Night at the Playboy Mansion
Etienne de Crecy - Tempovision
Live at Queen by DJ Alan Thompson - Trade
Moby - Play
Superfunk - Hold Up

$ mount
/dev/hda5 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/hdc on /mnt/cd type iso9660 (ro)

$ cat /etc/mtab
/dev/hda5 / ext2 rw 0 0
/dev/hda6 /mnt/perso ext2 rw,noexec,nosuid,nodev 0 0
none /dev/pts devpts rw,gid=5,mode=620 0 0
none /proc proc rw 0 0
/dev/hdc /mnt/cd iso9660 ro 0 0
```

Il devient rapidement pénible de définir à la main les arguments de la commande “mount”, surtout pour des lecteurs utilisés souvent comme les lecteurs cd. Heureusement, comme toujours (ou presque), il y a une solution.

3.5.2. Le fichier “/etc/fstab”

Grâce à ce fichier, “mount” va pouvoir faire la relation seul en lui passant seulement, soit le point de montage, soit le périphérique, en paramètre entre les diverses options et le type du système de fichier. Il s’organise comme ceci :

```
périphérique point de montage type fs options dump fsck
```

périphérique
Par exemple “/dev/hda1”

point de montage
Par exemple “/”

type fs
Par exemple “iso9660”

options
Diverses et variées, par exemple “rw”

dump
Mis à un ou à zéro permet de spécifier à l’utilitaire “dump” s’il doit sauvegarder cette partition ou non.

fsck

Mis à 1 pour le "/" et à 2 pour les autres partitions, indique qu'il est nécessaire de faire une vérification du système de fichier (on met à zero si ce n'est pas nécessaire).

Exemple :

```
/dev/sda6   /                           reiserfs    defaults          1 1
/dev/sda1   /boot                       ext2        defaults          1 2
/dev/sda8   /home                       reiserfs    defaults          1 2
/dev/sdc1   /home/mp3                   reiserfs    defaults          1 2
/dev/sdb2   /home/mp3/Blues-Jazz/Blues  reiserfs    defaults          1 2
/dev/hda1   /home/Bordel                reiserfs    defaults          1 2
/dev/cdrom  /mnt/cdrom                  iso9660     noauto,owner,ro   0 0
/dev/dvd    /mnt/dvd                    iso9660     noauto,owner,ro   0 0
/dev/sda5   /usr                        reiserfs    defaults          1 2
/dev/fd0    /mnt/floppy                 auto        noauto,owner      0 0
none        /proc                       proc        defaults          0 0
none        /dev/pts                    devpts      gid=5,mode=620    0 0
/dev/sda7               swap            swap        defaults          0 0
/dev/scd1               /mnt/graveur    iso9660     noauto,owner,ro   0 0
//yahiko/d$             /mnt/wind       smbfs       noauto,owner,user 0 0
//yahiko/e$             /mnt/wine       smbfs       noauto,owner,user 0 0
//yahiko/f$             /mnt/winf       smbfs       noauto,owner,user 0 0
none                    /proc/bus/usb   usbdevfs    devmode=0666      0 0
192.168.1.1:/photos     /mnt/photos     nfs         noauto,user,owner 0 0
/dev/sdd1               /mnt/camera     auto        noauto,owner      0 0
```

Le principe de démontages est très simple, il suffit pour ce faire d'utiliser :

```
umount partition ou point_de_montage
```

Toutefois, il y a une règle à respecter pour démonter une partition, c'est qu'aucune application ne l'utilise.

Il peut en effet arriver que vous souhaitiez démonter la partition pour en activer une nouvelle (Ex : un cd). Si vous êtes placés dans un des répertoires de ce même lecteur (ou si un autre utilisateur utilise des ressources situées sur la partition que vous voulez démonter, il ne vous sera pas possible de le faire.

Il faudra donc arrêter les tâches incriminées avant de pouvoir - correctement - démonter la partition. Cette tâche de savoir "qui utilise quoi" est grandement facilitée par la commande "fuser" (sous GNU/Linux du moins).

Elle va vous permettre d'envoyer des messages aux utilisateurs concernés pour qu'ils arrêtent d'eux-même le processus incriminés, d'envoyer un signal à tous les processus utilisant une certaine partie de notre arborescence…

```
$ fuser -v /

      USER  PID   ACCESS           COMMAND
/     root  1     .rc..            init
      root  2     .rc..            keventd
      root  3     .rc..            ksoftirqd_CPU0
      root  4     .rc..            kswapd
      root  5     .rc..            bdflush
```

```
root  6     .rc..            kupdated
root  94    .rc..            dhcpcd
root  98    .rc..            syslogd
root  101   .rc..            klogd
root  103   .rc..            inetd
[…]
root  130   .rc..            splogger
root  131   .r...            qmail-lspawn
root  132   .r...            qmail-rspawn
root  133   .r...            qmail-clean
root  1461  .r...            agetty
root  9022  .rc..            sshd
ualc  9024  .r...            zsh
root  9297  .rc..            sshd
root  12607 .rc..            sshd
root  12733 .r...            fuser
ualc  32634 .r...            ptrace
ualc  32644 .r...            ptrace
```

3.5.3. Monter des répertoires distants :

De nombreuses ressources de système de fichiers sont connectées par des systèmes en réseau. Les plus communs sont les ressources **Network File System** (NFS) et **Samba** (SMB). Pour accéder à ces systèmes de fichiers, vous devez d'abord connaître **l'hôte (**serveur), et le **nom du partage**. Pour connaître quels systèmes de fichiers un serveur distant exporte, utilisez les commandes suivantes :

Pour NFS :

```
# showmount -e <serveur distant>
```

Pour SMB :

```
# smbclient -L <serveur distant> -N
```

Lorsque vous connaissez l'hôte et le nom du partage, les commandes suivantes serviront à connecter le système de fichiers distant à l'arborescence du système de fichiers local.

Pour NFS :

```
# mount -t nfs serveur_distant:/partage /mnt/rep_locale
```

Pour SMB :

```
# mount -t smbfs //serveur/partage /mnt/rep_locale
```

Le **client NFS** est intégré au système, par contre pour monter un système de fichier SMB vous devez avoir **smbmount** d'installer.

Quelques options supplémentaires :

Pour NFS :

```
bg, fg, intr, nointr, soft, hard, async, sync, rsize et wsize.
```

Pour SMB :

```
username=<arg>,password=<arg>,ip=<arg>,uid=<arg>,workgroup=<arg> et
id=<arg>
```

Exemple :

Monter le répertoire NFS /toto du serveur tati sur /mnt/temp :

```
# mount -t nfs tati:/toto /mnt/temp
```

Monter ephemere sur /mnt/temp:

```
# mount -t smbfs -o username=toto_a,password=toto,workgroup=esi-
supinfo.com //fox/ephemere /mnt/temp
```

Si vous omettez l'option «password » mount vous le demandera par la suite.

Note : l'option -t fstype est facultative.

3.5.4. Gestion de l'espace

Une fois les partitions montées, il est nécessaire, si ce n'est indispensable de savoir quel espace reste disponible sur nos diverses partitions pour savoir combien de DivX on peut encore mettre, quelle taille font tels ou tels fichiers…

Nous avons, pour connaître l'espace restant sur les partitions montées, "df" et pour la taille des fichiers ou répertoires, "du". Les exemples pour ce genre de commandes étant plus fort qu'une explication qui serait juste les reflets de "man df" et "man du", voilà :

```
$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/hda5 3.9G 2.9G 802M 79% /
/dev/hda6 1004M 440M 512M 46% /mnt/Toto
/dev/hdc 598M 598M 0 100% /mnt/EnShort
```

```
$ df -h -T -a
Filesystem Type Size Used Avail Use% Mounted on
/dev/hda5 ext2 3.9G 2.9G 802M 79% /
/dev/hda6 ext2 1004M 440M 512M 46% /mnt/Toto
none devpts 0 0 0 - /dev/pts
```

```
none proc 0 0 0 - /proc
/dev/hdc iso9660 598M 598M 0 100% /mnt/EnShort
```

```
$ du -h -s /sbin
3.8M /sbin

$ du -h /bin/bash
472k /bin/bash
```

Lorsqu'il devient indispensable de trouver de la place, il va falloir utiliser des commandes pour compresser les données trops conséquentes.

Ceci va être rendu possible par des utilitaires comme zip, rar, gzip, bzip2. Les deux derniers de cette liste sont les plus couremments rencontrés sous Unix.

Le plus interressant au niveau intégrité des données et qualité de compression, s'avère être bzip2. Sa grosse particularité consiste à pouvoir récupérer, même partiellement, une archive si elle a été endommagée.

Cependant l'algorithme utilisé est efficace, mais plus lent que les autres. Tout dépend de la machine dont vous disposez et de l'utilisation que vous voulez faire de vos archives.

3.6. tar, gzip, bzip2

3.6.1. tar

```
tar [options] [fichiers]
```

La commande tar est une ancienne commande Unix qui permet aisément d'archiver, c'est-à-dire de réaliser la sauvegarde d'un ensemble de fichiers en un seul fichier, que l'on peut également compresser.

Certaines applications et mises à jour (les noyaux Linux notamment) ne sont livrées que sous ce format.

fichiers

Désigne un ensemble de fichiers ou toute une arborescence précédée d'un chemin absolu (à partir de /) ou relatif.
Si c'est un chemin absolu qui est indiqué, il sera conservé dans l'archive et permettra ensuite un désarchivage à la même position.

options

-x : extraire le contenu d'une archive
-c : créer une nouvelle archive
-t : afficher seulement la liste du contenu de l'archive, sans l'extraire
-f fichier : indiquer le nom du fichier archive
-v : mode bavard

-z : compresser ou décompresser en faisant appel à l'utilitaire gzip
-j : compresser ou décompresser avec l'utilitaire bzip2

Exemples :

Création

1 - Effectuer la sauvegarde de tous les fichiers du répertoire /home/toto dans le fichier save.toto.tar placé dans le répertoire courant :

```
[mathieu@escaflowne mathieu]$ tar -cvf sauve.toto.tar /home/toto
```

2 - Idem, mais le fichier archive est placé dans le répertoire /tmp :

```
[mathieu@escaflowne mathieu]$ tar -cvf /tmp/sauve.toto.tar /home/toto
```

ou

```
[mathieu@escaflowne mathieu]$ tar -c /home/toto > sauve.toto.tar
```

3 - Effectue, en plus de l'archivage, une compression des fichiers :

```
[mathieu@escaflowne mathieu]$ tar -cvzf sauve.toto.tar.gz /home/toto
```

Listing

```
[mathieu@escaflowne mathieu]$ tar -tvf sauve.toto.tar
```

Extraction

1 - Exécute le désarchivage dans le répertoire courant. Si l'archive a été créée par tar -cvf sauve.toto.tar /home/toto, il faut se placer à la racine / pour restaurer exactement le répertoire perso de toto :

```
[mathieu@escaflowne mathieu]$ tar -xvf sauve.toto.tar
```

2 - Décompresse et désarchive :

```
[mathieu@escaflowne mathieu]$ tar -xvfz sauve.tar.gz
```

3 - Ne désarchive dans l'archive que le répertoire désigné :

```
[mathieu@escaflowne mathieu]$ tar -xvfz sauve.tar.gz home/toto/tmp
```

3.6.2. gzip

Elle est utilisée pour compacter un fichier quelconque, et en particulier une archive tar.

Le décompactage se fait par la commande gunzip, ou de manière totalement équivalente par gzip -d.

```
gzip [options] [fichiers]
```

Elle peut décompacter les fichiers .gz, mais aussi les fichiers .z , .Z

options

-1 à -9 : fixe le niveau de compression
-d : décompresse
-c : écrit sur la sortie standard au lieu de remplacer le fichier d'origine (possibilité d'utiliser un pipe)
-l : affiche des infos sur la dé/compression
-r : dé/compresse tous les fichiers du répertoire passé en argument.

Exemples :

1 - Compresse backup.tar et le remplace par le fichier backup.tar.gz d'une taille beaucoup plus réduite :

```
[mathieu@escaflowne mathieu]$ gzip backup.tar /home/toto
```

2 - Compresse au maximum chaque fichier .txt séparément, et les renomme en ajoutant le suffixe .gz :

```
[mathieu@escaflowne mathieu]$ gzip -9 *.txt
```

3.6.3. bzip2

Bzip2 admet la même syntaxe que gzip, mais compresse mieux avec un besoin accru de mémoire.

3.7. Recherches de fichiers

3.7.1. find

```
find [chemin] [options] [expression]
```

Trouve tous les fichiers sur un système qui correspondent à une caractéristique. Find recherche dans les sous répertoires.

Exemple : Pour rechercher tous les fichiers mp3 contenus dans les répertoires personnels des utilisateurs :

```
[mathieu@escaflowne mathieu]$ find /home/ -name "*.mp3"
```

options

-atime +n, (-n) : Trouve les fichiers auxquels on a accédé il y a plus de n jours (ou moins de n jours)

-mtime +n, (-n) : Trouve les fichiers modifiés il y a plus de n jours (ou moins de n jours)

-name ”toto*” : Trouve les fichiers dont le nom commence par toto

-maxdepth n : Définit le niveau maximum de recherche dans les sous-répertoires

-type : Indique le type de fichier à rechercher. -l pour les liens symboliques, -d pour les répertoires

On peut utiliser find pour exécuter une commande sur certains fichiers :

```
find [chemin] [option] -exec [cmd] {} \;
```

Les fichiers trouvés sont passés à cmd comme des paramètres. {} garde la place du nom de fichier. Le point virgule délimite les commandes générées.

Exemple :

Pour effacer tous les fichiers .avi de mon répertoire :

```
[mathieu@escaflowne mathieu]$ cd ~
[mathieu@escaflowne mathieu]$ find . -name "*.avi" -exec rm {} \;
```

Dans ce cas, on peut aussi utiliser -ok qui est identique à -exec mais qui demande une confirmation pour chaque fichier à exécuter.

3.7.2. slocate – locate

Sur les systèmes Red Hat Linux, locate est un lien symbolique de la commande slocate.
La commande slocate utilise une base de données qui est mise à jour régulièrement.

Des fichiers récemment créés ne sont donc pas forcément référencés.
La base de donnée peut être mise à jour manuellement par le super utilisateur avec la commande **slocate –u** ou **updatedb**.

L’avantage de slocate est de permettre une recherche beaucoup plus rapide qu’avec la commande find.

Exemple : recherche tous les fichiers dont le nom comporte la chaîne toto.mp3 :

```
[mathieu@escaflowne mathieu]$ locate toto.mp3
```

4. “Pipes” et Redirections

Nous voici confronté à une des choses qui a contribué à l’engouement de beaucoup de personnes pour le monde Unix. Les “Pipes” et “Redirections”.

4.1. Les flux standards

Les flux standard sont au nombre de trois :

STDIN (0) : STDIN n’est ni plus ni moins que l’entrée au clavier (dite ”entrée standard”). Cela veut dire que tout ce que vous tapez au clavier va passer par ce canal d’entrée/sortie.

STDOUT (1) : STDOUT est ce que l’on appelle la ”sortie standard”. Les informations qui apparaissent à l’écran passent par le canal STDOUT.

STDERR (2) : STDERR est la ”sortie d’erreur standard”. L’utilité d’avoir deux flux de sortie permet de séparer les messages d’erreur et la sortie normale.

Les numéros 0,1,2 désignent respectivement STDIN, STDOUT et STDERR, et peuvent être utilisés dans les commandes shell.

4.2. Redirections

Ainsi, si nous exécutons un programme comme ”ls”, il va afficher le résultat à l’écran, donc sur la sortie standard. Il est possible de rediriger ces différents flux vers des fichiers, d’autres flux…

Ceci se fait par le biais des signes ”**<**"et ”**>**”. Une commande ”**commande >** un fichier” va créer un fichier ”un fichier” contenant le résultat de la commande ”commande”. De même, avec le signe ”**<**"au lieu de ”**>**"aurait utilisé le contenu du fichier ”un fichier” en entrée de la commande ”commande”.

Ca n’est pas très clair, regardez l’exemple :

```
$ cd /tmp
$ ls -l /bin
total 4872
-rwxr-xr-x 1 root root 3180 Jan 22 2001 arch
-rwxr-xr-x 1 root root 54905 May 22 2001 attr
-rwxr-xr-x 1 root bin 477692 Mar 22 2000 bash
-rwxr-xr-x 1 root bin 295012 Sep 3 1999 bash1
lrwxrwxrwx 1 root root 5 Nov 10 2000 bunzip2 -> bzip2
-rwxr-xr-x 1 root bin 27380 May 18 2000 bzip2
-rwxr-xr-x 1 root bin 6708 May 18 2000 bzip2recover
...

$ ls -l /bin > un_fichier
$ cat un_fichier
total 4872
```

```
-rwxr-xr-x 1 root root 3180 Jan 22 2001 arch
-rwxr-xr-x 1 root root 54905 May 22 2001 attr
-rwxr-xr-x 1 root bin 477692 Mar 22 2000 bash
-rwxr-xr-x 1 root bin 295012 Sep 3 1999 bash1
lrwxrwxrwx 1 root root 5 Nov 10 2000 bunzip2 -> bzip2
-rwxr-xr-x 1 root bin 27380 May 18 2000 bzip2
-rwxr-xr-x 1 root bin 6708 May 18 2000 bzip2recover
...
```

Si on utilise plusieurs fois des redirections sur un même fichier avec ce que l'on a vu, le fichier est écrasé à chaque fois. Pour éviter cela il est possible d'utiliser "**<<**" à la place de "**<**" et "**>>**" à la place de "**>**", auquel cas les données seront concaténées dans le fichier.

Exemple :

```
[06:18:12]tito@loktar ~ $ ls linux-lfs/
linux-lfs-flat.vmdk          linux-lfs.vmx                nvram
linux-lfs.vmdk               linux-lfs.vmx.WRITELOCK      vmware.log

[06:20:07]tito@loktar ~ $ ls evolution/
addressbook-sources.xml      config            mail          shortcuts.xml
cache                        gtkrc-mail-fonts  meta          vfolders.xml
camel-cert.db                local             searches.xml  views

[06:20:12]tito@loktar ~ $ ls linux-lfs/ >> fichier
[06:20:50]tito@loktar ~ $ ls evolution/ >> fichier

[06:21:01]tito@loktar ~ $ cat fichier
linux-lfs-flat.vmdk
linux-lfs.vmdk
linux-lfs.vmx
linux-lfs.vmx.WRITELOCK
nvram
vmware.log
addressbook-sources.xml
cache
camel-cert.db
config
gtkrc-mail-fonts
local
mail
meta
searches.xml
shortcuts.xml
vfolders.xml
views
```

Voici un exemple montrant la redirection de la sortie standard et de la sortie d'erreur standard :

Avec l'outil find, on va rechercher un fichier dont le nom comporte "tes" dans le répertoire /etc. On redirige les résultats dans le fichier "fichier" et les erreurs dans le fichier "erreurs" :

2> signifie que l'on redirige l'erreur standard et > signifie implicitement 1>

```
[mathieu@battousai mathieu]$ find /etc/ -name "*tes*" > fichier 2> erreurs
[mathieu@battousai mathieu]$ cat fichier
/etc/CORBA/servers/gnotes_applet.gnorba
```

```
/etc/sysconfig/network-scripts/ifup-routes
/etc/dumpdates
/etc/openldap/ldaptemplates.conf

[mathieu@battousai mathieu]$ cat erreurs
find: /etc/cups/certs: Permission denied
find: /etc/default: Permission denied
```

Notons que l'on peut rediriger l'erreur standard et la sortie standard vers le même fichier :

```
[mathieu@battousai mathieu]$ find /etc/ -name "*tes*" > fichier 2>&1
[mathieu@battousai mathieu]$ cat fichier
find: /etc/cups/certs: Permission denied
find: /etc/default: Permission denied
/etc/CORBA/servers/gnotes_applet.gnorba
/etc/sysconfig/network-scripts/ifup-routes
/etc/dumpdates
/etc/openldap/ldaptemplates.conf
```

Voici un dernier exemple montrant la redirection de l'entrée standard :

```
[mathieu@escaflowne mathieu]$ cat .bashrc
export PS1="\[\033[1;35m\][\u@\h \W]$\[\033[0m\] "
export PATH=$PATH:/usr/local/bin:/usr/local/sbin
BLA=`ps aux | grep XFree86`
if [ ! "$BLA" == "" ]; then
export TERM=xterm-color
fi
alias ncftp='ncftp3'
alias ls='ls -G'
alias vi='vim'

[mathieu@escaflowne mathieu]$ tr 'A-Z' 'a-z' < .bashrc
export ps1="\[\033[1;35m\][\u@\h \w]$\[\033[0m\] "
export path=$path:/usr/local/bin:/usr/local/sbin
bla=`ps aux | grep xfree86`
if [ ! "$bla" == "" ]; then
export term=xterm-color
fi
alias ncftp='ncftp3'
alias ls='ls -g'
alias vi='vim'
```

Avec tr, on a convertit les caractères majuscules en minuscules, le flux d'information venant du fichier .bashrc.

4.3. Pipes

Les "pipes" ("|") quand à eux sont "des voies de communication" d'un processus vers un autre.

Par exemple si nous faisons "**commande 1 | commande 2**", le résultat de la commande "commande 1" va être utilisé en entrée de la commande "commande 2".

Il est possible de chaîner ce type de commande autant que l'on veut. Rien de tel qu'un exemple :

```
$ ls -la /bin > un_fichier
$ grep "bzip" un_fichier
lrwxrwxrwx 1 root root 5 Feb 12 2001 bunzip2 -> bzip2
-rwxr-xr-x 1 root bin 27380 May 18 2000 bzip2
-rwxr-xr-x 1 root bin 6708 May 18 2000 bzip2recover

$ ls -la /bin | grep "bzip"
lrwxrwxrwx 1 root root 5 Feb 12 2001 bunzip2 -> bzip2
-rwxr-xr-x 1 root bin 27380 May 18 2000 bzip2
-rwxr-xr-x 1 root bin 6708 May 18 2000 bzip2recover
```

5. Gestion des processus

Comme nous l'avons constaté, ce n'est pas parce que nous sommes en ligne de commande qu'on se retrouve sous un MS-DOS. Un UNIX ne gère pas qu'une tâche à la fois, et il est possible de "dialoguer" avec elles.

L'intérêt réside notamment dans la capacité "de faire le café" tout en "pressant une orange".

Tout ceci ne peut s'exécuter en même temps si vous n'avez que deux bras (CPU). Il va falloir s'organiser (notion d'ordonnancement et de priorités) pour exécuter les tâches correctement.

Tout à coup, une deuxième paire de bras vous a poussé à l'issue d'une nuit étrange. Vous avez alors le potentiel d'exécuter à peu près deux fois plus de choses en même temps, si toutefois votre cerveau est capable de le gérer (notions de capacité du noyau à utiliser des architectures multi processus).

Il est donc tout naturel d'avoir toute une série d'utilitaires dédiés à consulter l'état des divers processus, à vérifier qu'ils se comportent conformément à ce qu'on attend d'eux, et à modifier leurs priorités.

5.1. Lister les processus

Pour ce faire, nous avons quatre commandes principales :

ps
: Permet de lister les processus suivant un certain nombre de critères.

jobs
: Permet de lister les processus lancés dans le shell courant.

pstree
: Permet de lister les processus dans un arbre indiquant les liens de parentés entre eux.

top
: Permet d'avoir un suivi de l'évolution des processus, et ce faisant, d'effectuer des traitement comme pourrait le faire ps, mais de façon interactive.

Voici un exemple d'utilisation de "ps"

```
$ ps
PID   TTY   TIME        CMD
12758 pts/4 00:00:00    zsh
12872 pts/4 00:00:00    ps

$ ps ax
PID   TTY   STAT  TIME  COMMAND
1     ?     S     2:35  init
2     ?     SW    0:00 [keventd]
```

```
3     ?     SWN   0:00  [ksoftirqd_CPU0]
4     ?     SW    0:00  [bdflush]
6     ?     SW    0:02  [kupdated]
94    ?     S     0:00  /sbin/dhcpcd eth1
98    ?     S     0:26  /usr/sbin/syslogd
101   ?     S     0:17  /usr/sbin/klogd -c 3
103   ?     S     0:00  /usr/sbin/inetd
105   ?     S     0:36  /usr/local/sbin/sshd
108   ?     S     0:00  /usr/sbin/crond -l10
110   ?     S     0:00  /usr/sbin/atd -b 15 -l 1
120   ?     S     0:00  gpm -m /dev/psaux -t ps2
123   ?     S     0:00  qmail-send
125   tty2  S     0:00  /sbin/agetty 38400 tty2 linux
126   tty3  S     0:00  /sbin/agetty 38400 tty3 linux
127   tty4  S     0:00  /sbin/agetty 38400 tty4 linux
```

```
$ ps auxww
USER  PID %CPU %MEM VSZ  RSS  TTY STAT START TIME   COMMAND
root  1   0.0  0.1  344  64   ?   S    Nov02 2:35   init
root  2   0.0  0.0  0    0    ?   SW   Nov02 0:00   [keventd]
root  3   0.0  0.0  0    0    ?   SW   Nov02 0:00   [ksoftirqd_CPU0]
root  4   0.8  0.0  0    0    ?   SW   Nov02 418:12 [kswapd]
root  5   0.0  0.0  0    0    ?   SW   Nov02 0:00   [bdflush]
root  6   0.0  0.0  0    0    ?   SW   Nov02 0:02   [kupdated]
root  94  0.0  0.0  332  52   ?   S    Nov02 0:00   /sbin/dhcpcd eth1
root  98  0.0  0.3  1788 232  ?   S    Nov02 0:26   /usr/sbin/syslogd
root  101 0.0  0.2  1296 180  ?   S    Nov02 0:17   /usr/sbin/klogd -c 3
root  103 0.0  0.4  1772 252  ?   S    Nov02 0:00   /usr/sbin/inetd
root  105 0.0  0.4  2288 264  ?   S    Nov02 0:36   /usr/local/sbin/sshd
root  108 0.0  0.4  1348 300  ?   S    Nov02 0:00   /usr/sbin/crond -l10
root  120 0.0  0.0  1268 8    ?   S    Nov02 0:00   gpm -m /dev/psaux -t ps2
```

Nous avons respectivement affiché les processus de notre terminal courant et nous appartenant, ceux dans tous les terminaux et pour tous les utilisateurs, et ceux dans tous les terminaux pour tous les utilisateurs, avec des informations sur les utilisateurs, l'état de la mémoire et les arguments qui leurs sont passés en entier (ils ne seront pas tronqués par la fin de la largeur du terminal).

Au milieu de toute une quantité d'informations, celles qui vont les plus nous intéresser sont le PID ("Process ID"), et l'état du processus ("STAT").

Le PID est un identifiant unique par processus.

Dans ce cadre, il y a une notion d'héritage : un processus parent qui exécute, par des procédés de type "fork", un autre processus se verra être le père de celui-ci.

En gardant à l'esprit ce principe, il y a donc un processus qui est le père de tous les autres. C'est le cas, et son nom est "init", qui a pour PID de 1. Cette notion d'héritage est particulièrement flagrante avec la commande "pstree" (cf exemple ci-dessous).

Pour ce qui est de la partie "STATUS", les lettres symbolisent ceci :

D

En sommeil imperturbable (ce qui arrive souvent à certain modules kernel)

R

En cours d'exécution

S

En sommeil

T

Stoppé ou "tracé" (notamment pour le débugage)

Z

Zombie (peut arriver lorsqu'un processus n'a pas attendu la fin de l'exécution de l'un de ses fils)

Tous ces indicateurs sont autant de moyens pour connaître à chaque instant l'état d'un processus. (Pour voir s'il est planté par exemple).
Voici, comme cité précédemment, une sortie du programme "pstree" :

```
$ pstree
init-+-Eterm---bash---soffice.bin-+-getstyle-gnome
     |                            `-soffice.bin---3*[soffice.bin]
     |-2*[Eterm---bash---su---bash]
     |-MozillaFirebird---run-mozilla.sh---MozillaFirebird---
                                  MozillaFirebird---4*[MozillaFirebird]
     |-4*[agetty]
     |-bdflush
     |-bonobo-activati
     |-cron
     |-devfsd
     |-dhcpcd
     |-esd
     |-gconfd-2
     |-gnome-panel
     |-kreiserfsd
     |-kscand
     |-ksoftirqd_CPU0
     |-kswapd
     |-kupdated
     |-login---bash---startx---xinit-+-X
     |                               `-gnome-session
     |-login---bash
     |-metacity
     |-metalog---metalog
     |-mini_commander_
     |-mixer_applet2
     |-multiload-apple
     |-nautilus---nautilus---3*[nautilus]
     |-sshd---sshd---sshd---bash---pstree
     `-xscreensaver
```

Cette fois une sortie du programme "jobs" (Attention cette commande est une commande interne au shell) :

```
$ jobs
[1] suspended (tty output) vi pouet
[2] + suspended (tty output) vi toto
[3] - suspended man man
```

Une fois encore, nous pouvons distinguer l'état du processus, un identifiant qui pourra nous servir dans les commandes qui vont suivre.

Voyons désormais une sortie typique du programme "top" permettant de suivre l'évolution d'un ou plusieurs processus dans le temps :

```
$ top
top - 09:49:02 up  5:16,  3 users,  load average: 0.10, 0.07, 0.02
Tasks:  71 total,   1 running,  70 sleeping,   0 stopped,   0 zombie
Cpu(s):   2.0% user,   0.7% system,   0.0% nice,  97.4% idle
Mem:    255124k total,   251676k used,     3448k free,     4608k buffers
Swap:   506512k total,       20k used,   506492k free,   128712k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2101 tito      15   0 68428  43m  33m S  1.3 17.4   0:04.17 soffice.bin
 1215 root      15   0 67116  37m 6144 S  0.3 15.2   1:02.32 X
 2066 tito      15   0  1888 1888 1700 S  0.3  0.7   0:00.02 sshd
 2187 tito      16   0  1000 1000  792 R  0.3  0.4   0:00.39 top
    1 root      15   0   484  484  436 S  0.0  0.2   0:06.04 init
    2 root      15   0     0    0    0 S  0.0  0.0   0:00.43 keventd
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd_CPU0
    4 root      15   0     0    0    0 S  0.0  0.0   0:00.01 kswapd
    5 root      15   0     0    0    0 S  0.0  0.0   0:00.06 kscand
    6 root      25   0     0    0    0 S  0.0  0.0   0:00.00 bdflush
    7 root      15   0     0    0    0 S  0.0  0.0   0:00.00 kupdated
    9 root      18   0     0    0    0 S  0.0  0.0   0:00.03 khubd
   14 root      15   0     0    0    0 S  0.0  0.0   0:00.00 kreiserfsd
  157 root      15   0   884  884  576 S  0.0  0.3   0:00.03 devfsd
  935 root      15   0   580  580  480 S  0.0  0.2   0:00.00 metalog
  936 root      15   0   420  420  356 S  0.0  0.2   0:00.00 metalog
 1091 root      15   0   636  636  544 S  0.0  0.2   0:00.00 cron
 1131 root      16   0  1200 1200  980 S  0.0  0.5   0:00.08 login
 1132 root      16   0  1144 1144  928 S  0.0  0.4   0:00.02 login
 1133 root      15   0   512  512  448 S  0.0  0.2   0:00.00 agetty
 1134 root      15   0   512  512  448 S  0.0  0.2   0:00.00 agetty
 1135 root      15   0   512  512  448 S  0.0  0.2   0:00.00 agetty
 1136 root      15   0   512  512  448 S  0.0  0.2   0:00.00 agetty
 1137 root      15   0   512  512  448 S  0.0  0.2   0:00.00 agetty
 1198 tito      16   0  1340 1340 1124 S  0.0  0.5   0:00.02 bash
 1203 tito      16   0   952  952  848 S  0.0  0.4   0:00.00 startx
 1214 tito      15   0   744  744  644 S  0.0  0.3   0:00.01 xinit
 1232 tito      15   0  3876 3872 1400 S  0.0  1.5   0:00.74 gnome-session
 1235 tito      15   0  7580 7576 1304 S  0.0  3.0   0:01.22 gconfd-2
 1245 tito      15   0  1456 1456 1140 S  0.0  0.6   0:01.02 xscreensaver
 1248 tito      15   0  1456 1456  924 S  0.0  0.6   0:00.61 gnome-smproxy
 1250 tito      15   0  4824 4820 2924 S  0.0  1.9   0:06.29 metacity
 1254 tito      15   0 11240  10m 5388 S  0.0  4.4   0:04.94 gnome-panel
 1257 tito      15   0 11624  11m 5760 S  0.0  4.6   0:00.00 nautilus
 1258 tito      15   0 11624  11m 5760 S  0.0  4.6   0:00.00 nautilus
```

Ici nous avons énormément d'informations concernant l'état général du système avec la mémoire utilisée, le nombre de processus actifs, le taux d'occupation du processeur, et le même type d'informations détaillées cette fois par processus.

La signification des champs est la suivante (et directement tirée du "man top") :

PID
Identifiant du processus

USER
Le propriétaire du processus

PRI
Priorité de la tâche (plus le chiffre est petit plus la tâche est prioritaire).

SIZE
Taille du processus en mémoire incluant la partie donnée et la pile.

RSS
Quantité totale de mémoire occupée par la tâche (les bibliothèques chargées sont comptabilisées pour les programmes ELF).

SHARE
Quantité de mémoire partagée.

STAT
La même définition des lettres que celle faite au-dessus avec en plus “a” ou “<” pour indiquer une priorité inférieure à zéro, “N” pour l’inverse et “W” pour les processus issue du “Swap” (le “man” précise que cela ne fonctionne pas pour les parties inhérentes au noyau).

LIB
Nombre de pages occupées par les bibliothèques (sauf pour les exécutables au format ELF ; autant dire que ça ne fonctionne quasiment jamais).

%CPU
Taux d’occupation du (des) processeur(s) par ce processus. Il peut être supérieur à 100% dans le cadre de machine ayant plusieurs processeurs.

%MEM
Taux d’occupation de la mémoire physique du processus

TIME
Temps total d’exécution au sein du (des) processeur(s) depuis que le processus est lancé.

COMMAND
Il vous est possible de “monitorer” l’état de vos processus et.

Comment interagir avec eux ? Comment les supprimer de la mémoire ?

5.2. Envoi de signaux aux processus

Ce procédé se concentre essentiellement autour de la commande “kill”. Il n’est pas, comme son nom pourrait le laisser supposer, dédié à “tuer” des processus. Il va permettre d’effectuer un certain nombre de tâches à l’un ou plusieurs d’entres eux.

Tout dépend finalement comment les programmes réagissent aux signaux qui leurs sont envoyés.

Pour obtenir la liste des signaux disponibles, il suffit de taper dans l’invite de commande du shell :

```
[mathieu@escaflowne mathieu]$ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
9) SIGKILL      10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2
```

Ensuite, lorsque vous souhaitez envoyer un signal à un processus, il suffit de récupérer son PID et faire :

```
kill [signal] PID
```

Si aucun nom de signal n’est passé en paramètre, c’est le “SIGTERM” qui est utilisé par défaut. Il est souvent utilisé pour arrêter un processus. Si celui-ci persiste, vous pouvez utiliser le “SIGKILL” qui ne peut être intercepté et évité par le processus, le forçant ainsi à se terminer (c’est une méthode très pratique lorsque Netscape plante).

D’autres signaux comme “SIGHUP” permettent aux processus qui le gèrent comme tel de les redémarrer en lisant éventuellement leur fichier de conf.

```
$ ps
PID   TTY   TIME        CMD
10689 pts/1 00:00:00    zsh
12739 pts/1 00:00:00    jed
13040 pts/1 00:00:00    ps

$ kill -9 12739
[1] + killed jed /tmp/bozoenshort
```

5.3. Arrière plan / Avant plan / Détachement

Le shell prévoit en général les outils “bg”, “fg” et “disown” afin de gèrer les processus. Respectivement, ils permettent de mettre en arrière plan, avant plan un processus et de le détacher de son propriétaire.

Le shell fournit également des raccourcis clavier permettant de faire ce genre de tâches. “Ctrl+z” permet notamment de mettre en arrière plan le processus courant.

```
$ man man &
[1] 13095
[1] + suspended (tty output) man man

$ vi pouet &
[2] 13106
[2] + suspended (tty output) vi pouet

$ jobs
[1] - suspended (tty output) man man
[2] + suspended (tty output) vi pouet

$ fg %1
```

Deuxième exemple :

```
[mathieu@escaflowne mp3]$ xmms
^Z
[1]+ Stopped xmms
[mathieu@escaflowne mp3]$ bg
[1]+ xmms &
[mathieu@escaflowne mp3]$ jobs
[1]+ Running xmms &
[mathieu@escaflowne mp3]$ fg %1
xmms
```

Imaginons que vous souhaitez qu'un processus (au hasard un client ftp) continue son exécution, et ce même lorsque vous quittez la session de votre shell. La solution consiste à se désattribuer le processus par le biais de la commande "**disown** PID". Attention, ce faisant vous n'aurez aucun moyen de vous le réapproprier.

Vous pouvez aussi, bien plus simplement, utiliser le programme **screen**, qui va, une fois lancé, émuler un shell. Cela veut dire que, lorsque vous lancerez **screen**, un autre shell tournera à l'intérieur de votre shell courant.

Vous pourrez y lancer des commandes (encore une fois, au hasard, des clients ftp), les détacher ("Ctrl+A" puis "Ctrl+D") et quitter votre session pour aller vous coucher.

Le lendemain matin, vous reviendrez à l'école et vous ouvrirez une session sur l'ordinateur du jour d'avant. Vous pourrez taper la commande **screen -r** pour réouvrir le shell que vous aviez détachée, et voir (par exemple) l'avancement de votre download. Pour sortir de **screen**, tapez "exit" ou "logout", voir même Ctrl+D.

```
[mathieu@escaflowne mathieu]$ screen
[mathieu@escaflowne mathieu]$ ncftpget
ftp://ftp.lip6.fr/pub/linux/distributions/redhat/linux/8.0/en/iso/i386/psyc
he-i386-disc1.iso"
psyche-i386-disc1.iso: ETA: 112:07 3.73/644.03 MB 97.47 kB/s
[detached]

[mathieu@escaflowne mathieu]$ screen -r
[mathieu@escaflowne mathieu]$ ncftpget
"ftp://ftp.lip6.fr/pub/linux/distribution
s/redhat/linux/8.0/en/iso/i386/psyche-i386-disc1.iso"
psyche-i386-disc1.iso: ETA: 87:50 10.04/644.03 MB 123.20 kB/s
```

5.4. Modification des priorités du “Scheduler”

Nous avons vu précédemment que les processus avaient des priorités. Elles sont modifiables aussi bien au moment du lancement du programme, qu’après, respectivement, grâce à “nice” et “renice”.

Nous avons également vu que la priorité d’un processus est d’autant plus grande que la valeur de son “nice” est petite.

Il n’y a plus qu’à se lancer, la syntaxe étant très simple :

```
 nice -n PRIORITE commande args
```

et

```
 renice PRIORITE -p PID
```

Il faut manier l’ordonnancement des processus avec précautions, surtout lorsqu’on est root où il est possible de mettre des priorités de +20 à -20.

Nous voulons modifier la priorité de notre windowmanager, dans notre cas ”gnome”, de PID 1232 (cf top). Ici, nous devons passer root pour accomplir la tâche suivante :

```
$ su -
Password:

# renice -15 -p 1232
230: old priority 0, new priority -15
```

Si nous relançons le programme top, et nous concentrons uniquement sur la ligne qui nous intéresse, nous obtenons :

```
1232 mathieu 2 -15 8820K 7176K select 0:36 0.00% 0.00% gnome-session
```