Introduction à l'administration de systèmes UNIX

M. Codutti

27 mars 1997

Résumé

Ces notes accompagnent la série d'exposés que j'ai donné sur l'administration de machines UNIX. Il s'agit d'une retranscription de mes transparents avec quelques mots de liaison. Il ne s'agit en aucun cas d'un cours complet et indépendant sur l'adminsitration UNIX. Vous trouverez en annexe un lexique de quelques acronymes et termes spécifiques.

Table des matières

Chapitre 1

Introduction

Dans ce chapitre, nous exposons quelques généralités sur l'administration des machines (rôle, super-user, types d'interactions avec la machine et différences entre les UNIX).

1.1 Rôles d'un administrateur

Un administrateur est amené à effectuer des tâches plus ou moins régulières dont

- Installation des machines et de leur OS. Bien que les machines comportent en général un système pré-installé, il est souvent préférable de le réinstaller afin de le configurer selon ses besoins (essentiellement en ce qui concerne les partitions du disque)
- Gestion des utilisateurs (ajout, retrait)
- Ajout de composants hardware (RAM, disques, ...)
- Sauvegarde régulière (essentiellement les fichiers utilisateurs)
- Ajout de logiciels
- Configuration des services (mail, imprimantes, NIS, DNS, ...)
- Surveillance du système (performance, sécurité)
- Délivrance d'informations aux utilisateurs, assistance.

1.2 Unix ou Unixs

Chaque UNIX est différent, surtout au niveau de l'administration, cependant on distingue 2 grands groupes, BSD et Sys V .

BSD (Berkeley)	Sys V (AT & T)
SunOS (Sun)	Solaris (Sun)
Linux (PC)	IRIX (Silicon)
	HP-UX (HP)

Il reste des différences à l'intérieur d'un groupe notamment en ce qui concerne les outils de haut niveau.

1.3 Le Super User

La gestion se fait via le compte `root`. Il faut veiller à bien choisir le mot de passe et à ne pas le diffuser. L'idéal est d'avoir un et un seul administrateur par machine ou groupe de machines. En tout cas, un seul qui puisse modifier le système. Pourquoi ?

- L'administration est souvent affaire de choix personnels. Il faut veiller à une certaine cohérence.
- Il existe de nombreux liens entre les différentes parties du système. D'où l'importance de l'historique qui sera mieux appréhendé si une seule personne s'en occupe.

Quoi qu'il en soit, il faut veiller à noter ce qu'on fait (en cas d'absence ou de remplacement, ou pour se rappeler de ce que l'on a fait il y a longtemps)

1.4 Interactions avec la machine

L'administrateur peut agir avec le système à différents niveaux

1. On peut utiliser des outils graphiques de plus en plus nombreux et développés (`admintool` sur Solaris , `sam` ailleurs).
   Il sont en général intuitifs mais bien souvent limités et fort différents d'un OS à l'autre. De plus, ils ne peuvent être intégrés dans un script pour automatiser une tâche plus conséquente.
2. On peut également utiliser des commandes UNIX (par ex: `useradd`)
   Ces commandes ont un champ d'action plus grand que la méthode 1 mais restent limitées. Elles sont également plus difficiles à utiliser.
3. Enfin, il est possible de se *salir les mains* en travaillant directement sur les fichiers de configurations.
   Cette méthode est la plus difficile mais permet un contrôle plus grand.

En pratique, tout cela est question de convenances. Si vous connaissez les outils graphiques, autant les utiliser s'ils répondent à vos besoins. D'autre part, si vous connaissez la méthode 3, les autres viennent d'elles-mêmes. C'est pourquoi, dans ces notes, je parlerai des méthodes 2 et 3.

Chapitre 2

Gestion des utilisateurs

Dans ce chapitre, nous voyons comment ajouter ou supprimer des utilisateurs dans le système. Créer un compte pour un utilisateur implique 3 étapes

1. L'enregistrer dans le système (login, passwd, ...)
2. Lui attribuer une *home_directory*, espace réservé sur le disque dur.
3. Configurer son compte.

Comment ? Comme souvent, cela peut se faire via un outil graphique, via une commande (`useradd`) ou en manipulant les fichiers. Ici, nous décrivons comment le faire au niveau le plus bas. Nous introduisons tous les concepts qui permettent de comprendre instantanément la méthode graphique.

2.1 Enregistrer un utilisateur

Un utilisateur est caractérisé par une ligne dans le fichier `/etc/password`. Voici sa structure

```
login:passwd:uid:gid:comment:home:shell
```

et sa signification

login. Il s'agit du nom de l'utilisateur. C'est un alphanumérique de 8 caractères maximum.

passwd. Il apparaît dans ce fichier sous forme codée.

uid. Il s'agit d'un numéro **unique** de l'utilisateur. Compris entre 0 et 65535. Les 100 premiers nombres sont par convention réservés au système et ne correspondent pas à des utilisateurs normaux.

gid. Numéro de groupe. Chaque utilisateur appartient à un groupe principal. Il pourra également appartenir à des groupes secondaires. Cette notion de groupe interviendra au niveau des permissions sur les fichiers.

comment. Nom complet de l'utilisateur.

home. Chemin complet de la directory attribuée à l'utilisateur

shell. Chemin complet du shell, le programme qui interagit avec l'utilisateur et qui permet de taper des commandes (`csh, sh, bash, tcsh, ...`).

Pour éditer ce fichier, il est recommandé d'utiliser la commande `vipw`, au lieu de `vi`, ce qui assure que vous êtes le seul à modifier le fichier (cohérence).

2.2 Mot de passe

Le mot de passe apparait codé dans le fichier `/etc/passwd`. Vous devez dès lors laisser ce champ vide et utiliser une commande pour le définir. Cette commande est

`passwd` *`login`*

qui définit ou modifie le mot de passe de *`login`*.

Le mot de passe doit être composé de 6 à 8 caractères (au delà, ils ne sont pas pris en compte). Voici quelques règles pour bien choisir un mot de passe

- Mélanger chiffres, lettres et caractères spéciaux.
- Utiliser les premières lettres des mots d'une phrase facile à retenir.
- Utiliser des approximations phonétiques ou visuelles.
- Ne pas utiliser des dérivés du login.

Exemples de bons mots de passe : per6val, Tyfo1de, Aster1*, lsldvd
Exemples de mauvais mots de passe : barbara, stones, kepler1, Batman, godgod, Palermo.

Qu'est-ce qui permet de dire qu'un mot de passe est mauvais ? Le fichier `/etc/passwd` est public (toute personne qui a un compte sur la machine peut le lire). Grâce à cela, ont été développés des logiciels (comme `crack`) qui, à partir d'un dictionnaire et de règles de transformations, codent des milliers de mots de passe jusqu'à trouver une version codée qui correspond à celle se trouvant dans le fichier `/etc/passwd`. On trouve ainsi le mot de passe en clair. Un mauvais mot de passe est donc un mot de passe découvert par un tel logiciel. Il vaut toujours mieux pour un administrateur lancer un tel programme avant que quelqu'un d'autre ne le fasse. Nous en reparlerons dans un chapître consacré à la sécurité.

Pour contrecarrer cette faille, certains systèmes (surtout Sys V) ont introduit le fichier `/etc/shadow` (lisible uniquement par `root`) qui ne contient que le mot de passe qui disparaît alors de `/etc/passwd`. Si on ajoute un utilisateur à la main, cela implique d'éditer les 2 fichiers.

Certains systèmes (à nouveau surtout Sys V) introduisent également le concept de *Password Aging* qui permet d'imposer une durée de vie limitée à

un mot de passe et ainsi forcer les utilisateurs à en changer régulièrement. Les avis sont partagés quant à cette possibilité car cela amène souvent les utilisateurs à utiliser alternativement deux mots de passe.

Notons enfin que certains systèmes (surtout Sys V , vous l'aurez compris) sont exigeants lorsqu'un mot de passe est introduit et utilisent quelques règles pour refuser des mots de passe trop faciles. Ce système existe par exemple sur les machines du centre de calcul.

2.3 Home Directory

La home directory est destinée à accueillir tous les fichiers d'un utilisateur. Pour la créer, la séquence ressemble à

```
mkdir /home/bob
chown bob /home/bob
chgrp 100 /home/bob
```

2.4 Shell

La création d'un compte implique également de placer dans la home directory des fichiers de configuration dont le fichier associé au shell (`.cshrc`, `.profile`, `.basdrc`, ...). Nous reviendrons dans un autre chapitre sur ce que cela implique.

2.5 Supprimer un utilisateur

Pour supprimer temporairement un utilisateur ou lui empêcher momentanément l'accès, il suffit de remplacer le mot de passe codé dans `/etc/passwd` par le caractère `*` (il s'agit là d'une convention, tout autre caractère peut convenir). Comme aucun mot de passe ne peut donner ce caractère une fois codé, cela bloque l'accès au compte.

Pour supprimer définitivement un compte, il faut

1. *Backuper* la home
2. Détruire la home
3. Supprimer l'utilisateur dans `/etc/passwd`
4. Supprimer les quelques fichiers qui peuvent traîner hors de sa home (notamment la mailbox)

Il est conseillé de ne jamais réutiliser un *uid*, ce qui pourrait avoir pour conséquence d'attribuer à un nouvel utilisateur des fichiers de l'ancien qui traînent encore dans le système.

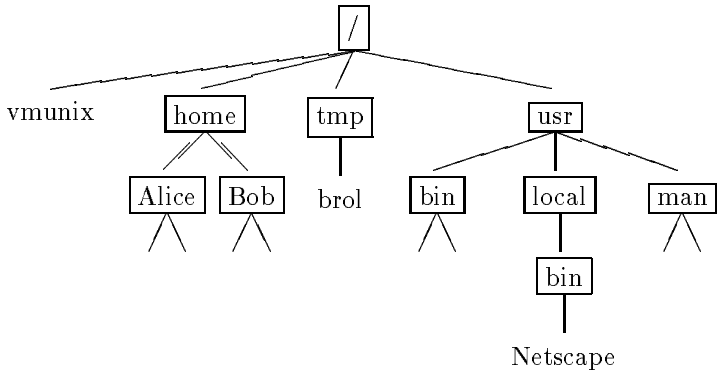
Chapitre 3

Le système de fichiers

Le système de fichiers (*filesystem* en anglais) est le concept par lequel on accède à toutes les informations du système. Cela comprend bien sûr tous les fichiers d'informations (textes, programmes, ...) mais également des *pseudo-fichiers* qui vont faire le lien avec les composants hardwares.

3.1 Structure

Le filesystem a la structure suivante



On peut le voir comme un arbre dont les noeuds sont des directory et les feuilles des *fichiers* au sens large (cela peut être des directory vides ou des fichiers spéciaux).

Un fichier du filesystem est identifié par le chemin à parcourir à partir de la racine. Exemple: `/usr/local/bin/netscape`. Il est également possible de spécifier le chemin relativement à une *directory courante*. Exemple: si la directory courante est `/home`, on peut se contenter de `Bob`.

3.2 Quelques répertoires standards

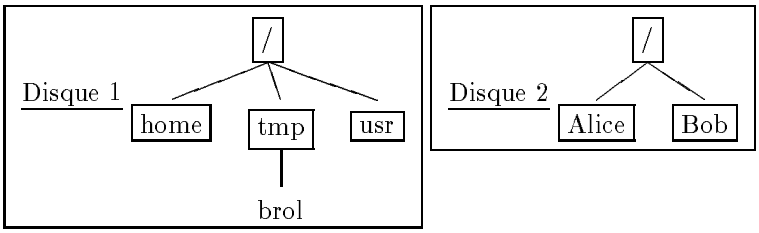
Voici une liste, non exhaustive, de directory que l'on rencontre dans la plupart des systèmes avec une courte decription. Elles seront détaillées par la suite.

Directory	Rôle
`/bin`, `/sbin`	Commandes de base nécessaires au démarrage
`/etc`	fichiers de configuration
`/lib`, `/usr/lib`	librairies
`/tmp`, `/usr/tmp`	fichiers temporaires (détruits au démarrage)
`/usr/bin`, `/usr/sbin`	Commandes du système (orienté utilisateur)
`/usr/5bin`, `/usr/ucb`	*idem*
`/usr/man`	les manuels
`/var`	fichiers dont la taille varie fortement
`/var/adm`, `/var/log`	fichiers de contrôle de l'activité
`/usr/local`, `/opt`	fichiers locaux

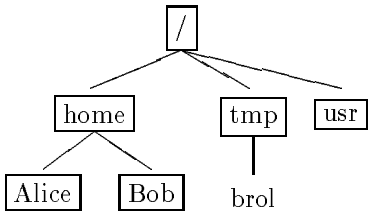
3.3 Les partitions

Si d'un point de vue logique il n'existe qu'un filesystem, physiquement il en va autrement. Une machine peut posséder plusieurs disques durs, un lecteur de CD-ROM, un lecteur de disquettes. Tous ces supports d'informations conteniennent un filesystem. Tous ces filesystems vont être regroupés pour n'en former qu'un seul.

Exemple: soit 2 disques durs contenant les filesystems suivants



Ces deux filesystems sont *montés* en un seul lors du démarrage de la machine pour donner



Notons que le disque 1 doit contenir une directory `home` pour que le deuxième puisse s'y raccrocher. Notons également que ce deuxième commence par `/` comme tout filesystem. Il n'y a aucune référence à `home` et il pourrait d'ailleurs être monté ailleurs. La manière dont les filesystems sont combinés est déterminée par un fichier de configuration que nous verrons à une autre occasion.

Notons enfin que tout disque dur peut être divisé en *partitions* qui sont des disques durs logiques (max. 8) au sein d'un même disque dur. Ce partitionnement aura pour intérêt de séparer des fichiers qui ont des objectifs différents. Nous verrons cela lorsque nous parlerons de l'installation d'un OS.

3.4 Vérifier l'occupation

Voyons à présent deux commandes qui permettent de vérifier l'occupation des disques.

La commande `df` (`df -k` sur Sys V) donne l'occupation des différentes partitions.

Exemple:

```
mcodutti@cso8:df
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0      19047   13032    4115    77%    /
/dev/dsk/c0t0d0s6     480919  344046   88783    80%    /usr
/dev/dsk/c0t0d0s3      76967   23926   45351    35%    /var
/dev/dsk/c0t0d0s5    1085262  851078  125664    88%    /opt
```

Examinons le résultat de cette commande (certaines lignes ont été supprimées pour simplifier notre propos).

Filesystem fichier spécial du file system relié à une partition physique. Nous verrons à un autre moment comment lire cela. Ici, nous avons les partitions 0, 6, 3 et 5 du disque 0.

kbytes l'espace total disponible sur la partition

used l'espace actuellement occupé

avail l'espace disponible

capacity l'occupation exprimée en pourcentage

Mounted on l'endroit où cette partition est montée sur le filesystem

Vous aurez remarqué que `used+avail` $\neq$ `kbytes`. Un certain pourcentage d'une partition (souvent 10%) est gardé en réserve. Il pourra être utilisé mais la capacité indiquera 100% avant et des messages d'alerte apparaîtront.

Une commande complémentaire est `du`, elle donne l'espace total occupé par une directory.

Exemple:

```
root@lit1:du -s /var/*
5000    /var/adm
790     /var/mail
345     /var/spool
1259    /var/tmp
```

La taille est souvent exprimée en Kilobytes mais parfois aussi en 1/2 Kbytes (surtout Sys V)

3.5 Types de fichiers

Après avoir étudié le filesystem dans son ensemble, voyons à présent le type de fichiers que l'on peut y rencontrer. Cette liste n'est pas exhaustive

directory

fichiers ordinaires. On entend par là la plupart des fichiers. Aussi bien les textes, les exécutables, les inputs et outputs de programmes, ...

character/block devices files. Il s'agit de pseudo-fichiers qui sont, en fait, un lien vers un périphérique (Disque, terminal, sortie parallèle, ...). Ils sont tous dans la directory `/dev` ou `/devices`.

hard/symbolic links. Ils permettent de donner plusieurs noms à un même fichier et, par extension, de rencontrer un même fichier à plusieurs endroits dans le filesystem.

3.5.1 Directory

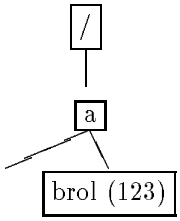
Il faut savoir que, dans le système, un fichier est identifié par un *inode*, numéro unique dans la partition. Une directory est un fichier contenant une table associant un nom à un inode

Exemple:

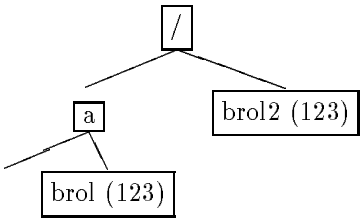
Nom	inode
.	123 *(directory courante)*
..	126 *(directory parent)*
`vmunix`	345
`etc`	12
`bin`	7

3.5.2 Liens physiques

Prenons comme exemple le file system suivant



où le nombre entre parenthèses indique le inode associé au fichier. La commande `ln /a/brol /brol2` aura pour effet de créer un fichier pointant vers le même fichier physique.

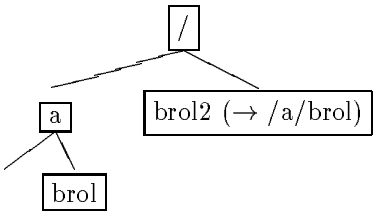


A partir de là, toute action sur `brol2` sera répercutée sur `brol`. Si on supprime `brol`, `brol2` existera toujours. Pourquoi ? Un fichier physique a un compteur reprenant le nombre de références dans le filesystem. Effacer un fichier dans le filesystem revient à *décrémenter* le compteur. Ce n'est que lorsqu'il tombe à 0 que le fichier est physiquement effacé.

Attention: ce mécanisme ne fonctionne qu'à l'intérieur d'une même partition et ne permet pas de créer des liens entres des fichiers de partitions différentes (ce que peuvent faire les liens symboliques).

3.5.3 Liens symboliques

Reprenons l'exemple précédent. La commande `ln -s /a/brol /brol2` va créer un fichier ASCII `/brol2` dont le contenu sera le fichier pointé `/a/brol`



Lorsqu'une action sera intentée sur `/brol2`, le système constatera qu'il s'agit d'un lien symbolique, lira le chemin du fichier pointé et répercutera l'opération vers celui-ci.

Ce genre de liens fonctionne même si le fichier pointé n'existe pas encore et surtout, il fonctionne **entre les partitions**. Ces liens permettent de dissocier l'emplacement physique d'un fichier de son emplacement logique.

Exemple: Vous décidez d'installer tous les fichiers associés au logiciel Netscape dans une même directory pour ne pas les éparpiller. Mais pour que l'exécutable soit trouvé par le système, il doit se situer dans une certaine autre directory. Pas de problème, il suffit de créer un lien symbolique de la directory des exécutables vers la directory netscape.

3.6 Les permissions

Chaque fichier est possédé par un *propriétaire*, appartient à un *groupe* et est associé à des *permissions* représentées par 12 bits. En voici leur description

r,w,x read, write et execute. Il existe 3 triplets. Un pour le propriétaire, un pour le groupe et un pour le reste des utilisateurs.

setuid modifie les permissions d'un exécutable

setgid modifie le groupe d'un exécutable en action

sticky l'ancien sens de ce bit devient obsolète et les différents OS l'ont récupéré pour des usages divers. En Solaris , par exemple, il permet de créer une certaine privacité dans des directory autrefois publiques comme `/tmp`

3.6.1 Effets de r,w et x

Le sens de ces permissions est différents pour les fichiers normaux et les directory

- Pour un fichier

  r permet de lire le fichier

  w permet d'écrire (modifier, append, tronquer)

  x indique qu'il s'agit d'un programme. Peut être exécuté.

- Pour une directory

  r permet de voir le contenu

  w permet d'écrire dans la directory (ajouter, effacer, renommer un fichier)

  x permet de traverser la directory

Voici quelques exemples pour mieux comprendre les liens entre les permissions et les opérations permises. Supposons que dans la directory où nous sommes, existe une directory `brol` contenant le fichier `brol2`. Dans le tableau nous indiquons diverses opérations avec les permissions minimales pour que cette opération puisse se réaliser.

brol	brol2	action
r		`ls`
x		`cd brol`
x	r	`cat brol/brol2`
wx		`rm brol/brol2` (demande une confirmation si brol2 n'a pas w)
x	w	modifier `brol/brol2`
wx		`touch brol/brol3`
x	w	`cat brol/brol3 >brol/brol2`
wx		`mv brol/brol3 brol/brol2`

Le résultat le moins intuitif est que l'on peut supprimer un fichier qui ne possède pas de permission en écriture s'il se trouve dans une directory qui, elle, possède cette permission.

3.6.2 Le setuid

Prenons un exemple pour comprendre le fonctionnement de ce bit. Supposons qu'il existe un programme `Nettoye` appartenant à l'utilisateur `untel` tel que `Nettoye dir` détruise la directory `dir`. Supposons, en outre, qu'il existe une directory `brol` appartenant à ce même `untel` et que lui seul peut la détruire.

Si l'utilisateur `telautre` tape `Nettoye brol`, il se verra refuser l'opération car le programme va s'exécuter avec les permissions de `telautre`. Sauf si le *setuid* bit est mis, auquel cas il va s'exécuter avec les permissions de `untel`, le propriétaire du fichier `Nettoye`, et la destruction sera acceptée.

3.7 Quelques commandes pour manipuler les permissions

Voyons comment lire et modifier les permissions d'un fichier ainsi que le propriétaire et le groupe et comment définir les permissions par défaut.

3.7.1 La commande `ls`

```
mcodutti@cso8:ls -lg
total 135
drwx------   2 root   other       512 Jan 10  1996 Mail/
-rw-r--r--   1 root   other     42932 Dec 13 09:33 core
```

Avant de décrire comment sont affichées les permissions, voyons les autres informations données par la commande `ls`.

- Le nombre en deuxième position indique le nombre de références vers le fichier physique (cf liens physiques),
- vient ensuite le proriétaire du fichier et le groupe auquel il appartient,

- nous avons ensuite la taille (en bytes) du fichier,
- puis la date,
- et enfin, le nom du fichier (suivi d'un `/` dans le cas d'une directory).

Les permisssions, quant à elles, sont représentées par 10 caractères dont la signification est la suivante:
Le premier indique le type de fichier

- pour un fichier normal

d pour une directory

l pour un lien symbolique

c pour un character device file

b pour un block device file

Les suivants indiquent les permissions (r,w et x) pour le propriétaire, le groupe et le reste du monde (dans cet ordre). Le troisième caractère (`x`) peut toutefois apparaître différemment pour indiquer d'autres permissions

- Le bit `x` du propriétaire peut apparaître comme un `s` pour indiquer que le *setuid* est mis (`S` si le `x` n'est pas mis)
- Le bit `x` du groupe suit la même modification pour le *setgid*
- Le bit `x` du reste du monde apparaît comme un `t` si le *sticky bit* est mis (à nouveau, `T` si le `x` n'est pas mis)

3.7.2 Modifier les permissions

Les permissions associées à un fichier peuvent être modifiées via la commande `chmod perm file` où `file` est le fichier en question et `perm` indique les permissions. Celles-ci peuvent être indiquées de 2 façons:

numériquement. Notation octale des 12 bits de permissions dans l'ordre : setuid, setgid, sticky, r owner, w owner, x owner, r group, w group, x group, r world, w world, x world.

Exemple: 4755 indique `rwxr-xr-x` + setuid = `rwsr-xr-x`

symboliquement par l'expression `who op perm` où

who est `u` (owner), `g` (group), `o` (world) ou `a` (all)

op est + pour ajouter une permission, - pour en enlever ou = pour mettre exactement des permissions (idem approche numérique)

perm est `r, w, x, s` ou `t`

3.7.3 Modifier le propriétaire et le groupe

Cela se fait via les commandes

```
chown owner file
chgrp group file
```

Attention: Cette commande est généralement réservée au `root` même si certains OS permettent également que le propriétaire actuel d'un fichier les utilise.

3.7.4 Permissions par défaut

La commande `umask` spécifie les bits qui sont mis à 0 lors de la création d'un fichier.

Exemple: Avec la commande `umask 022` une directory, normalement créée avec les permissions 777, aura 755 et un fichier, normalement créé avec les permissions 666, aura 644.

Ce masque est généralement défini dans le fichier d'initialisation du shell.

Chapitre 4

Les processus

Un **processus** est une instance d'un programme en cours d'exécution. Il est caractérisé par du *code*, des *données* et des attributs dont

PID Numéro du processus (**unique** à un moment donné)

état prêt à tourner?

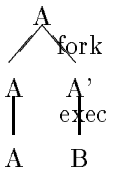
priorité politique d'attribution du CPU

propriétaire détermine les permissions du processus

PPID PID du processus parent

4.1 Création d'un processus

Les processus naissent par duplication/différenciation. Voyons par exemple comment un processus A va créer un processus B



Le processus A' est identique au processus A sauf en ce qui concerne les points suivants (non exhaustif)

- A' reçoit un nouveau PID
- Son PPID est le PID de A
- L'accounting est réinitialisé

Voyons un exemple de code C mettant en évidence les appels systèmes qui effectuent pratiquement cette création.

```
A:    int kidpid;
      kidpid = fork();
      if( kidpid==0 ) { /* fils */ exec(B); }
      /* père. kidpid donne le pid du fils */
```

4.2 Les états d'un processus

Le processeur exécute chaque processus *prêt* par petits intervalles. Macroscopiquement, c'est comme si plusieurs processus avançaient en parallèle. Mais un processus n'est pas toujours prêt. Voici quelques uns de ses états possibles

CPU c'est l'heureux élu. Il tourne

Runnable prêt à être exécuté

Sleeping en attente d'un événement, d'une ressource

Zombie prêt à mourir

Stoppé interdit de CPU

Swappé complètement sur le swap

4.3 Les signaux

Les **signaux** permettent d'agir sur un processus (arrivée d'une donnée, temps écoulé, ...). Lorsqu'un processus reçoit un signal, 3 possibilités s'offrent à lui

1. Il peut avoir prévu comment le gérer
2. Il peut avoir décidé de l'ignorer (si possible)
3. Une action par défaut est entreprise

Lorsqu'un signal est reçu, et s'il n'est pas ignoré, le contrôle est passé automatiquement à la routine de gestion de ce signal (fournie par le processus dans le cas 1 ou par le système dans le cas 3). Une fois ce traitement fini, on revient à l'endroit exact dans le processus avant l'interruption (sauf si le traitement du signal a entrainé la mort de ce processus).

4.3.1 Quelques signaux utiles

On reprend ici une liste des signaux les plus utiles du point de vue administration. Nous indiquons si un processus peut décider de l'ignorer et s'il peut fournir sa propre routine de gestion.

Num.	Nom	Description	Défaut	Ignorer?	Gérer?
1	HUP	Eveil	Terminer	Oui	Oui
2	INT	Interrompre (CTRL-C)	"	"	"
3	QUIT	Quitter	"	"	"
9	KILL	Tuer	"	Non	Non
	STOP	Stop	Stop	"	"
	TSTP	Stop Clavier (CTRL-S)	"	Oui	Oui

Pour certains signaux, le numéro n'est pas indiqué car il varie d'un système à l'autre.

4.4 Les priorités

Le processeur est attribué par tranches aux processus *prêts* en fonction de leur priorité. Cette priorité dépend de

classe du processus (système, real-time, time-sharing, intercatif, ...)

temps CPU déjà utilisé

Nice paramètre modifiable par l'utilisateur

Plus *nice* est petit, plus la priorité est grande. En BSD elle varie de -19 à 19. En Sys V elle va de 0 à 39.

Un utilisateur ne peut qu'augmenter la valeur *nice* d'un processus lui appartenant, alors que le super-user a tous les pouvoirs.

4.5 Le swap

Le **swap** (ou **mémoire virtuelle** en français) est une mémoire auxilliaire sur le disque dur. Il est utilisé comme extension à une RAM trop petite.

Il faut savoir qu'un processus occupe des *pages* de mémoire (dont la taille dépend de l'OS). Les pages occupées mais non utilisées (parce que le processus n'est pas prêt ou se trouve pour le moment dans une autre partie du code par exemple) peuvent être transférées sur le swap afin de soulager ainsi la RAM (c'est ce qu'on appelle le **swap out**). Si nécessaire, elles seront rappelées automatiquement en RAM (**swap in**).

Ce mécanisme est très utile dans un environnement multi-utilisateur où plein de processus *inactifs* cohabitent.

Pour vérifier l'occupation du swap, la commande varie d'un OS à l'autre. Par exemple `swap` (IRIS et Solaris), `free` (Linux), `swapinfo` (HP-UX), `pstat -s` (SunOS).

4.6 Les démons

L'OS n'est pas monolithique mais composé d'une multitude de processus spécialisés (*modularité*). Cela permet notamment de ne charger que ce qui est nécessaire ou d'upgrader facilement une partie.

Un **démon**, c'est cela : un processus du système tournant en tâche de fond et remplissant un rôle spécifique.

Exemple: `sendmail` qui envoie et reçoit les mails ou `in.rlogind` qui accepte les connections rlogin

Dans les OS modernes, ils peuvent être lancés automatiquement si nécessaire.

4.7 Commandes pour gérer les processus

Voyons à présent quelques commandes pour visualiser les processus et modifier leur caractéristiques.

4.7.1 `ps` sous BSD

La syntaxe de la commande `ps` est différente sous BSD et Sys V . Nous les voyons donc séparément. Commençons par la version BSD .

```
root@cso8:ps -aux | head
USER       PID %CPU %MEM   SZ  RSS TT       S    START  TIME COMMAND
tcouss    3319 93.5  5.0 6752 6288 pts/9    R 11:15:54  5:01 mapleV -p 7,6
mcodutti 28320  3.0  7.410928 9376 ?        S 17:17:10  2:04 xemacs
```

La signification des colonnes est la suivante (pour les moins évidentes)

% CPU Moyenne utilisation du CPU

% MEM Pourcentage de mémoire RAM occupée

SIZE Taille totale du processus

RSS Taille RAM effectivement occupée (le reste est en swap)

TIME temps total CPU attribué (en min:sec)

4.7.2 `ps` sous Sys V

```
root@cso8:/usr/bin/ps -ef | more
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Dec 22 ?        0:00 sched
    root     1     0  0   Dec 22 ?        4:22 /etc/init -
```

La signification des colonnes est la suivante (pour les moins évidentes)

PPID PID du parent

STIME date de début du processus

C lié à la priorité

TTY terminal attaché au processus

4.7.3 Top

`top` est un programme qui donne des informations utiles sur les processus les plus actifs. Il n'est pas disponible par défaut sur tous les processus mais il est *gratuit*.

Par rapport à `ps`, il a les avantages suivants

- rafraichissement dynamique de l'écran
- infos sur le swap, la RAM
- infos sur la charge CPU
- permet de voir/modifier le *nice*

Cette commande est pratique pour surveiller un système.

4.7.4 `kill`

Permet d'envoyer un signal à un processus
Exemple: `kill -HUP 1` pour envoyer le signal 1 au processus 1.

4.7.5 Nice

Pour voir le *nice* d'un processus, utiliser la commande `top`. Pour modifier ce *nice*, utiliser `top` ou `renice` (BSD). Pour donner une priorité quand on lance un processus, `nice valeur commande`.

4.8 Nohup

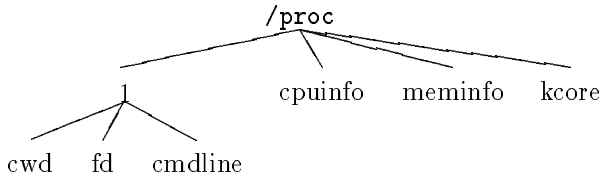
Lorsqu'un utilisateur se déconnecte, le shell envoye un signal `SIGHUP` à tous les descendants. Souvent (si le processus ne le capte pas ou ne l'ignore pas) cela amène à tuer un processus en background. La solution est d'utiliser la commande `nohup` qui va lancer le processus avec une option pour ignorer le `SIGHUP`.

Exemple: `nohup batch &`

Notons toutefois que certains shells (dont les dérivés du C-shell) font un `nohup` automatique si on utilise le `&`.

4.9 /proc

Il s'agit d'une *pseudo-directory* reprenant les caractéristiques des processus (pour consultation). Cette directory se retrouve essentiellement sous Sys V mais également sous Linux . Elle ressemble à



Elle contient une directory par processus dont le nom est son PID. Elle contient également des fichiers reprenant des infos sur le système.

Il est vivement recommander de ne pas y toucher.

Chapitre 5

Allumer et éteindre un ordinateur

Normalement, il suffit de mettre sous tension une machine et après un temps plus ou moins long, le système est chargé et la machine prête à travailler. Nous allons voir quelles sont les étapes de cette mise en route afin de pouvoir modifier la séquence de démarrage (pour démarrer sur le CD/ROM par exemple) ou réparer en cas de problème.

En gros, la séquence de démarrage implique les étapes suivantes que nous analyserons plus en détail

1. En tout premier lieu, c'est le **moniteur** qui a la main
2. Il effectue alors du **bootstrapping**
3. Enfin le **kernel** est chargé et la machine est initialisée.

5.1 Le moniteur

Le **moniteur** est un petit programme en PROM dans la machine (plus ou moins équivalent au Setup d'un PC). Il a pour rôle d'identifier le périphérique de démarrage et d'initier le bootstrapping. Il procède en général automatiquement mais il est possible de l'interrompre. On dispose alors d'une interface pour

- Vérifier les composants hardwares
- Modifier la séquence de démarrage
- Indiquer des options de démarrage

On peut y accéder de 2 façons

1. En l'interrompant par une séquence appropriée de touches (dépend de la machine)
2. En sortant de UNIX.

5.1.1 Le moniteur sur SUN

Comme illustration, voyons quelques aspects du moniteur SUN.

La séquence de touches permettant d'y accéder est `STOP-A`. Il existe en fait 2 versions de ce moniteur (l'ancien et le nouveau). L'ancien se reconnait par le prompt `>` alors que le nouveau a pour prompt `ok`. Toutefois, le nouveau moniteur peut se mettre en mode ancien. Il suffit alors de taper `n` pour voir apparaître le prompt du nouveau mode.

Voici quelques commandes et leur signification :

`boot disk|cdrom|net` pour booter sur le disque, le cdrom ou le réseau. `disk` est un alias vers un disque bien précis (configurable)

`probe-scsi` pour répertorier les périphériques SCSI

`boot -s` pour booter en mode *single-user*

`boot -r` à utiliser si on a ajouté un périphérique depuis le dernier démarrage.

5.2 Kernel

Le **kernel** est le coeur de UNIX. C'est lui qui va (parfois indirectement) mettre en place UNIX. Il est souvent appelé `/unix`, `/vmunix`, `/vmlinuz`, ... Les opérations effectuées sont

1. Détecter les périphériques et les configurer
2. Créer les processus spontanés
3. Mode single-user (optionnel)
4. Scripts de démarrage
5. Mode multi-user

5.3 Les processus spontanés

Les premiers processus ne sont pas créés par la méthode habituelle mais *spontanément*. Leure nombre et leur nature dépendent fortement de l'OS. Toutefois, il y aura toujours le processus `init`, le plus important. C'est lui qui va prendre en charge, une fois lancé, le reste de la configuration de la machine et lancer les autres processus nécessaires.

5.4 Mode single-user

Après avoir commencé l'initialisation, le processus `init` peut passer en mode single user (si cette option a été demandée au démarrage ou lors du précédent arrêt).

Dans ce mode, l'opérateur prend le contrôle sur un OS minimal (uniquement partition root et aucun démon lancé). Ce mode permet de réparer des problèmes qui pourraient empêcher un démarrage correct (partition endommagée, mauvais script de démarrage, ...).

L'opérateur a, dans ce mode, les permissions du root. Ce qui pose un problème de sécurité. De nombreux OS demandent le mot de passe du root avant d' entrer dans ce mode.

On en sort en tapant `CTRL-D`

5.5 Scripts de démarrage

Les scripts de démarrage vont configurer la machine. Voici quelques opérations effectuées :

1. Donner un nom à l'ordinateur
2. Définir la zone horaire
3. Vérifier les disques
4. Les assembler (monter en un seul filesystem)
5. Nettoyer `/tmp`
6. Configurer le réseau
7. Lancer les démons

Le mécanisme mis en oeuvre est très différent entre BSD et Sys V . Nous allons donc les examiner séparément.

5.5.1 Cas des systèmes BSD

C'est le cas le plus simple. Le processus `init` exécute tous les scripts de la forme `/etc/rc*` (`/etc/rc.d/*` pour Linux). Il commence en fait par `rc.boot` qui va appeler les autres.

Mais le plus important est `rc.local` car c'est celui-là que l'on va modifier selon ses besoins.

Exemple: ajouter une commande pour démarrer automatiquement un gestionnaire de licence.

5.5.2 Cas des systèmes Sys V

Dans ce cas, c'est beaucoup plus compliqué que pour BSD (mais c'est aussi plus riche et plus souple) Tous d'abord, on introduit le concept de *niveau* de fonctionnement.

0 Arrêt

1 ou s single-user

2 multi user (ressources non exportées)

3 multi user complet (défaut)

6 reboot

Démarrer ou arrêter revient à changer de niveau. A chaque niveau vont être associés des scripts.

Inittab

Le fichier `/etc/initttab` défini les scripts à lancer lorsqu'on entre dans un niveau.

Exemple: Voici une ligne extraite de la version Solaris

```
s3:3:wait:/sbin/rc3 >/dev/console 2>&1 </dev/console
```

s3 nom arbitraire

3 concerne l'entrée dans le niveau 3

rc3 script à exécuter

wait il faut attendre que le script soit fini avant de passer à autre chose

Les scripts de niveau

A chaque niveau (disons 3 pour faciliter notre propos) on associe, entre autres, un script `/etc/rc3` qui va lancer tous les scripts se trouvant dans la directory `/etc/rc3.d`.

En fait, cette directory contient des scripts dont le nom suit une règle précise. Cela commence par la lettre `S` (pour start) ou `K` (pour kill). Vient ensuite un numéro et enfin un nom mnémotechnique.

Exemple: `S88sendmail` est un script dont le rôle est de démarrer `sendmail` le démon dévoué au mail.

Le script `/etc/rc3` va d'abord lancer tous les scripts de la forme `K*` dans l'ordre des numéros avec l'option `stop`

Exemple: `K20lp stop`

Il va ensuite lancer tous les scripts de la forme `S*` avec l'option start

En fait, ces scripts sont des liens symboliques vers la directory `/etc/init.d` qui contient tous les scripts de tous les niveaux.

init.d

Voyons par un exemple ce que peut contenir cette directory et comment tout cela est relié.

Supposons que l'on ait un gestionnaire de licence à démarrer au niveau 3 et à tuer au niveau 0. On va créer le script `/etc/init.d/licence` qui ressemblera à

```
#!/bin/sh
case ``$1'' in
   'start')
      # on lance ici le démon
   'stop')
      # on tue ici le démon
esac
```

Il ne reste plus qu'à créer les liens symboliques qui vont définir son utilisation

```
ln -s /etc/init.d/licence /etc/rc3.d S20licence
ln -s /etc/init.d/licence /etc/rc0.d K20licence
```

5.6 Eteindre un ordinateur

Il existe plusieurs façons d'éteindre un ordinateur. Décrivons cela en commençant par le plus sûr.

5.6.1 `shutdown`

Cette commande est la plus sûre et la plus propre. Elle permet de

- programmer un arrêt à l'avance
- prévenir les utilisateurs (message explicatif)
- empêcher les logins si le shutdown est imminent

Elle possède également des options pour indiquer à quel niveau on veut arriver

- arrêter la machine
- passer en mode single user
- rebooter la machine

La syntaxe exacte est différente d'un système à l'autre

Exemple: En Linux , `shutdown -h 23:00 'Upgrade Systeme'` indique qu'on programme un arrêt à 23 heures. Les utilisateurs seront prévenus par le message indiqué.

5.6.2 `halt`

Cette famille de commandes (on trouve aussi **`reboot`** ainsi que, sur BSD uniquement, **`fasthalt`** et **`fastboot`**) est tout aussi sûre que la commande précédente mais est bien plus brutale. Personne n'est prévenu et l'action est immédiate. Notons que le **`fast`** dans les 2 dernières commandes n'indique pas que l'arrêt doit être rapide mais bien le prochain démarrage. Partiquement, ces commandes écrivent un fichier vide dont le nom est **`/fastboot`** puis passent la main à leur homologue (**`halt`** pour **`fasthalt`** et **`reboot`** pour **`fastboot`**). Au démarrage, le système voit la présence de ce fichier et se met en mode *fast*, ce qui équivaut à ne pas vérifier les disques (l'opération la plus longue). De plus, le fichier **`/fastboot`** est détruit.

5.6.3 séquence de touches

Cette solution est à éviter au maximum car elle peut aboutir à une inconsistance sur le disque, ce qui signifie généralement la perte de fichiers. Ceci est du au système de caches qui fait qu'une écriture sur le disque est d'abord gardée en mémoire dans un zone tampon pour être écrite en bloc, ce qui présente l'avantage d'accélérer les accès au disque.

Notons que ceci n'est pas vrai pour Linux où la touche **`ALT-CTRL-DEL`** est *remappée* sur la commande **`reboot`**.

5.6.4 Power off

En plus de présenter le même danger que la méthode précédente, celle-ci peut provoquer des dégâts physiques sur des vieux disques pour lesquels la tête de lecture ne se met pas automatiquement en zone sûre lors d'une coupure de courant.

Chapitre 6

Les device drivers

Les **device drivers** (gestionnaires de périphériques en français) sont des programmes qui servent d'interface entre l'OS et une composante matérielle (disque, sortie série, terminal, ...)

Ils permettent d'accéder aux périphériques de manière uniforme. Ils traduisent les opérations en fonction des particularités des périphériques.

Un driver peut gérer plusieurs périphériques et fonctionner en mode *caractère* (lecture/écriture séquentielle des informations) ou mode *bloc* (accès par blocs)

6.1 /dev

On accède aux drivers via des pseudo-fichiers dans la directory `/dev` (+ `/devices` sous Solaris)

Exemple: `cat /etc/passwd >/dev/printer` pour imprimer le fichier password.

Un pseudo fichier est associé à un driver via un numéro **majeur** et un numéro **mineur**.

majeur désigne le driver

mineur spécifie un des périphériques parmi ceux gérés par le driver ou une autre manière de le considérer (densité, bloc/caractère, ...)

Exemple:

```
cso6!root:ls -l /dev/sd0a
brw-r-----  1 root       7,   0 Nov 21  1995 /dev/sd0a
```

On voit que le driver 7 s'occupe de disques durs et que la partition a du disque 0 est connue comme périphérique 0 par ce driver.

6.2 Nomenclature

Les noms des pseudo-fichiers associés aux drivers suivent (en général) les quelques règles suivantes:

1. quelques lettres pour indiquer ce que c'est + des chiffres pour les distinguer entre eux
2. un `r` en début indique l'accès caractère (raw) plutôt que bloc
3. chez certains OS, les drivers similaires sont regroupés dans une même sous-directory. (`disk`, `tape`, ...)

6.2.1 Les disques

SunOS `(r)sds3g` pour indiquer la partition `g` (la 7ème) du disque numéro 3. **Attention:** ce numéro est différent du numéro SCSI du disque.

Linux (IDE) `hda2` indique la 2ème partition du disque A (désigné C sous DOS)

Solaris `(r)dsk/c0t2d0s3` où

c0 contrôleur (driver) 0

t2 Numéro SCSI 2

d0 drive numéro 0 de ce driver

s3 partition 3

Irix `(r)dsk/dks1d6s7` pour le contrôleur 1, disque 6, partition 7

HP idem Irix

6.2.2 Les tapes

Le nom indique également la *densité* et s'il faut rebobiner la bande à la fin de l'opération.

SunOS `(n)rst0`

(n) non rewind

rst raw Scsi Tape

0 tape 0 en densité basse

Solaris `rmt/0(lbn)`

rmt Raw Magnetic Tape

0 tape 0

l low density (aussi m,h,u,c)

b BSD

n non rewind

Irix `rmt/0m(n)` tape 0 (n pour non rewind)

HP `(nr)tape` (nr) pour nonrewind.

6.2.3 Les disquettes

SunOS `(r)fd0c`

Solaris `(r)diskette0`

Linux `fd0`

6.2.4 Les sorties séries

Les sorties séries peuvent être désignées sous différents noms. On retrouve souvent `ttya` et `ttyb` ou `cua0` et `cua1` qui réfèrent aux mêmes sorties mais avec une configuration différente.

6.2.5 Les terminaux

Chaque terminal physique mais également virtuel (une fenêtre telnet ou rlogin) est associé à un fichier dont le nom contient `tty` mais également d'autres lettres pour indiquer le type et des chiffres pour les différencier.

6.2.6 La console

le fichier `console` est toujours relié à la console

6.2.7 Rien

Le fichier `null` est toujours relié à rien, ce qui permet de jeter de l'information.

Chapitre 7

Backup et compression

Ce chapitre aborde le problème délicat des backups. Nous évoquerons également, en quelques mots, les outils de compression.

7.1 Compression

Compresser un fichier consiste à le retranscrire sous un autre format afin qu'il prenne *moins de place*. Il n'est plus directement lisible par un être humain mais cela permet de gagner de 50% à 60% pour du texte courant. Si l'espace disque est rare, il est bon de compresser les gros fichiers que l'on utilise peu. Il faut noter que ceci est différent du codage lié aux aspects de sécurité.

7.1.1 `compress`

Tous les UNIX fournissent la commande `compress`.

`compress file` remplace `file` par `file.Z`, sa version compressée.
`uncompress file` rempalce `file.Z` par `file`.

7.1.2 `gzip`

Gzip (GNU zip) est très répandu et peut être obtenu gratuitement (s'il ne se trouve pas déjà sur l'OS) via le site FTP du GNU.

`gzip file` remplace `file` par `file.gz`
`gunzip file`, opération inverse.

7.1.3 Autres formats

Voici, pour information, quelques autres formats de compression que l'on retrouve surtout dans le monde des PC.

extension	programme
zip	pkzip
rar	rar
arj	arj
exe	programme à exécuter, contient le fichier compressé + le décompresseur.
lha	lha

et deux formats liés aux MAC.

extension	programme
sit	Stuffit (par exemple)
hqx	idem

7.2 Backup

L'information contenue sur les disques est souvent plus importante que l'ordinateur lui-même. Il est impératif de s'assurer contre la perte d'informations due à

- une défaillance matérielle
- une destruction par un logiciel
- une erreur de l'utilisateur (`rm *`)
- un désastre (incendie, tremblement de terre, raz de marée, ...)

La stratégie adoptée va dépendre de

- la quantité, le roulement et l'importance de l'information
- la somme que l'on est prêt à engager.

Les backups peuvent se faire sur

Disquettes. Trop faible quant au volume et à la fiabilité

Tape. Format QIC. Capacité de 40 Mb à 2.5 Gb. Encombrant et archaïque.

Exabyte. Format Vidéo 8mm. Capacité de 2 Gb à 40 Gb.

DAT. Format Digital Audio Tape 4mmm. Capacité de 4 Gb à 32 Gb.

L'exabyte et le DAT sont les meilleurs choix. Ils sont proches en terme de coût et de fiabilité bien que le marché semble manifester une légère préférence pour le DAT.

7.3 `tar`

`tar` collecte plusieurs fichiers en un seul, ce qui facilite leur manipulation.

créer: `cd /; tar cf /tmp/home.tar home`

contenu: `tar tf /tmp/home.tar`

extraire: `cd /tmp; tar xf home.tar home/bob`
(va extraire `bob` dans `/tmp/home/bob`)

sur bande: `cd /; tar cf /dev/nrst0 home`

compression: `cd /; tar zcvf /tmp/home.tgz home`
(Tar de GNU uniquement)

7.4 Se promener sur une bande: `mt`

Il est possible de mettre plusieurs enregistrements à la suite sur une bande
Exemple: `tar cf /dev/nrst0 /home; tar cf /dev/nrst0 /usr` donne

/home : /usr :

Le lecteur de bandes met des repères pour pouvoir retrouver les débuts d'enregistrements CEPENDANR

– il n'y a pas de table des matières

– il ne connaît rien du format du contenu (tar, dump, autre, ...)

C'est pourquoi, il faut **tout noter soigneusement**.
Pour se promener sur la bande, on dispose de la commande `mt`

`mt rew`	pour rebobiner
`mt fsf n`	pour avancer de n enregistrements
`mt off`	pour éjecter la bande

Exemple:

`mt rew; mt fsf 2`	on se retrouve au début du 3ème
`tar cf /dev/nrst0 /tmp`	Sauver `/tmp` et fin du 3ème
`mt rew`	début du 1er
`tar xf /dev/nrst0`	extraire `/home` et fin du 1er
`mt fsf 1`	début 2ème (**Attention**)
`tar xf /dev/nrst0`	extraire `/usr` et fin du 2ème

7.5 Dump

`dump` est la méthode la méthode la plus appropriée aux backups. En effet,

- on peut tout *backuper* (`tar` ne peut pas backuper `/dev`)
- respecte les liens (moins facile avec `tar`)
- notion de **backup incrémental**

Un backup se fera à un certain *niveau* (de 0 à 9). Un backup de niveau i consistera à sauver tout ce qui a été modifié depuis le dernier backup à un niveau inférieur. Un backup de niveau 0 revient à tout backuper.

7.6 Stratégie de backup

La stratégie de backup à adopter dépend

- du nombre de bandes disponibles
- de l'époque à laquelle on veut pouvoir remonter et avec quelle précision
- de la facilité de récupération
- du temps de backup

Exemple: 1 backup hebdomadaire de niveau 0 avec 5 bandes à tour de rôle + 1 backup annuel que l'on garde en permancence

Cet exemple

- permet de remonter de semaines en semaines jusqu'à 1 mois ou d'années en années.
- nécessite des backups complets, ce qui est long
- permet une restauration facile car tout est sur une bande

Exemple: 1 backup annuel de niveau 0 (on garde la bande) + 1 backup mensuel de niveau 0 (12 bandes à tour de rôle) + 1 backup hebdomadaire de niveau 5 (4 bandes à tour de rôle).

Cet exemple

- permet de remonter de semaines en semaines jusqu'à 1 mois, de mois en mois pendant 1 an et d'années en années.
- ne nécessite qu'un backup complet en début de mois, ce qui est plus rapide.
- peut requérir 2 bandes lors de la restauration.

7.7 syntaxe de dump

La commande pour un `dump` ressemble à

```
dump 0uvf /dev/nrst0 /dev/rsd0g
```

où

0 niveau (ici full backup)

u mettre à jour l`/etc/dumpdates` qui retient les dates et niveaux des backups (essentiel pour les backups partiels)

v verbeux

f on spécifie le nom du tape (si différent tape par défaut)

Attention: On spécifie le pseudo-fichier lié à une partition en mode *raw* (`rsd0g`) et pas le filesystem correspondant (`/usr`). Mais certains OS le permettent.

Solaris : la commande est `ufsdump`

Si `dump` se méprend sur le tape (densité, longeur), utiliser des options (s,d) pour l'éclairer.

7.8 Remote dump

Il est possible d'effectuer un backup sur le tape d'une autre machine. La syntaxe est

```
rdump 0uf orca:/dev/nrst0 /dev/rsd0g
```

Attention: Il faut les permissions réseau (`.rhosts`)

Cette commande permet de centraliser les backups en écrivant un script qui lance les backups de tous les disques d'un groupe de machines.

Exemple: soit 3 machines (A,B,C) avec un tape branché sur B. On peut lancer sur A un script qui ressemble à ce qui suit pour backuper les 3 machines.

```
# script sur A pour lancer le backup
      rdump B:tape partitions
rsh B  dump   tape partitions
rsh C rdump B:tape partitions
```

7.9 Récupération

Pour récupérer des fichiers sauvés sur bande, il faut

1. déterminer sur quelle(s) bande(s) se trouvent les fichiers à restaurer.
2. restaurer dans l'ordre chronologique

3. choisir la directory où on restaure. En effet, la restauration se fait dans la directory courante, d'où
   - dans `/tmp` puis les déplacer
   - directement à la bonne place

une partie	`restore xvf /dev/nrst0 fichiers`
tout	`restore rvf /dev/nrst0`
réseau	`rrestore xvf orca:/dev/nrst0 fichiers`
interactif	`restore ivf /dev/nrst0`

La méthode interactive lance un programme qui permet de se ballader dans le filesystem sauvé sur la bande et d'indiquer ce qui doit être restauré.

ls idem `ls` UNIX (contenu)

cd dir idem `cd` UNIX (change directory)

add file—dir ajoute le fichier ou la directory (récursif) dans la liste des fichiers à restaurer. Il apparaîtra avec un `*` lors d'un `ls`.

extract lance la restauration.

Chapitre 8

Tâches périodiques

Certaines opérations doivent être effectuées régulièrement (nettoyer le disque, vérifier le système, ...). UNIX fournit un mécanisme pour lancer des commandes à intervalles réguliers (`cron`). Il s'agit d'un démon tournant en permanence et utilisant des fichiers de configuration (`crontab`) lui indiquant ce qui doit être lancé et quand.

La directory `/var/spool/cron/crontabs` (`/usr/spool/cron/crontabs` sous HP) contient un fichier par utilisateur. Ce fichier associé à un utilisateur a pour nom son login et contient toutes les commandes à lancer pour son compte. Il a le format suivant

```
# commentaire
minute heure jour mois jour_semaine commande
```

Exemple: `0,30 8-20 * * * verif`
indique que la commande `verif` doit être lancée toutes les heures précises et les heures 30, tous les jours entre 8h et 20h30.

Les fichiers de configuration sont gérés via `crontab`

`crontab -l`	liste le contenu du fichier
`crontab -e`	pour éditer (en `vi`) le fichier
`crontab -e user`	pour éditer le fichier de `user` (uniquement par `root`)

Chaque fois qu'une commande est lancée, l'utilisateur reçoit un mail contenant le résultat (l'output) de la commande.

Chapitre 9

Le shell

Un **shell** est un programme servant d'interface avec l'utilisateur. Il accepte des commandes et lance leur exécution. Il y a en fait plusieurs shells qui se ressemblent tous un peu avec quelques fonctionnalités différentes et quelques différences de syntaxe

`sh`	Bourne Shell (début des années '70)
`csh`	C-Shell (fin des années '70)
`ksh`	Korn-Shell. `sh` + facilités du `csh`
`bash`	Bourne Again SHell. Soutenu par GNU. `sh` + facilités interactives (comme les flèches)
`tcsh`	`csh` + flèches

9.1 Les scripts

Un **script** est une suite de commandes (dans un fichier) que le shell va exécuter comme si on les tapait interactivement.

Pour lancer un script, il y a deux possibilités.

1. `source script` (en `(t)csh` et `bash`)
   `. script` (en `(k)sh` et `bash`)
2. Le rendre *exécutable* (permission `x`) et taper simplement son nom.

Les deux solutions ne sont pas tout-à-fait identiques en ce sens que, dans la deuxième, un sous-shell, distinct du shell courant, est lancé pour exécuter ces commandes.

Si on veut exécuter un script avec un autre shell (ou un tout autre programme) que le shell courant, il faut l'indiquer par une première ligne bien précise dans le fichier.

```
#!/bin/sh
```

en tête d'un fichier indique que le programme `/bin/sh` (le Bourne shell ici) doit être lancé pour traiter le fichier.

9.2 Initialisation

Lorsqu'il démarre, le shell lit un script d'initialisation dans la HOME de l'utilisateur. On peut ainsi configurer le shell et l'environnement UNIX.

`sh`

`.profile` (au login)

`(t)csh`

`.login` (au login)
`.(t)cshrc`
`.logout` (au logout)

`bash`

`.bash_profile` ou `.bash_login` ou `.profile` (au login)
`.bashrc` (si ce n'est pas un shell associé à un login)
`.bash_logout` (au logout)

Sur certains OS, il y a également lecture d'un fichier global avant. Si ce n'est pas le cas, on peut simuler cette faculté en faisant commencer le fichier local par un `source` du script commun.

9.3 Les variables

Il faut faire la distinction entre les variables locales, qui ne sont accessibles que dans le shell, et les variables globales (on dit plutôt *d'environnement*) qui sont passées à tous les fils lancés par le shell (comme les commandes)

`(t)csh`

L'assignation d'une variable locale se fait par `set VAR=valeur` et celle d'une variable globale par `setenv VAR valeur`. La liste des variables locales (globales) connues ainsi que leur valeur courante s'obtient par `set` (`setenv`).

`(ba)sh`

Ici, une variable globale est une variable locale qui a été exportée. L'assignation se fait par `VAR=valeur` et une variable est exportée avec `export VAR`. Pour une liste des variables, `set`. Alors que `env` ne donne que les variables exportées.

Dans tous les shells, on fait référence à la valeur d'une variable par `${VAR}` ou `$VAR` si cela ne provoque pas d'ambiguïté.

Exemple: (en Bourne Shell)

```
# var1=1
```

```
# var2=2
# export var2
# sh                          (en entre dans un autre shell)
# echo $var1
# echo $var2
2
```

9.3.1 PATH

Cette variable d'environnement défini où le shell cherche les commandes.

1. Si le nom contient le caractère `/`, il est interpreté comme un chemin (relatif ou absolu) Exemple: `#/usr/bin/more` ou `#./brol`

2. Sinon, le shell cherche un fichier de ce nom dans un des répertoires spécifiés par `PATH` (dans l'ordre).
   Exemple:

   ```
   #echo $PATH
   /bin:/usr/bin:.:/usr/bin/X11
   #more                              (trouvé car dans /usr/bin)
   ```

Pour des raisons de sécurité, il est vivement recommandé de ne pas avoir `.` dans son `PATH`.

En `(t)csh`, cette variable est lue une fois au démarrage pour des raisons d'efficacité. Si on la modifie et si on souhaite que le shell en tienne compte, il faut utiliser la commande `rehash`.

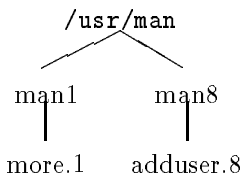
Notons également que certaines commandes sont internes au shell et non pas définies dans des fichiers externes. Elles sont ainsi plus rapides et toujours accessibles.

Exemple: `cd`, `source`, `alias`, `echo`, `kill`, ...

9.3.2 MANPATH

Les manuels accessibles via la commande `man` sont regroupés en sections (un répertoire par section).

Exemple:



Ces versions sont dans un langage appelé `nroff` (penser à LaTeX). Il sont compilés lorsqu'on y fait appel. Parfois, on dispose également des versions précompilées (répertoires `cat1`, ...).

`MANPATH` définit tous les chemins où l'on peut trouver des manuels

Exemple:

```
# echo $MANPATH
/usr/man:/usr/local/man:/usr/X11/man
# man more
```

va chercher dans les répertoires `/usr/man/mani`, `/usr/local/man/mani` et `/usr/X11/man/mani` un fichier de la forme `more.i` où `i` est un symbole parmi une liste dépendant de l'OS (souvent 1 à 9 ainsi que `l` et `n`).

9.3.3 LD_LIBRARY_PATH

Un programme utilise des fonctions définies dans des librairies. Ces fonctions peuvent être incorporées

1. dans l'exécutable lors de la compilation. C'est la méthode classique. On parle de compilation statique.
2. ou lors de l'exécution, dans le processus, à partir de la librairie sur le disque. On parle de compilation dynamique.

La deuxième méthode s'impose de plus en plus. Elle permet d'obtenir un exécutable moins gros et permet des upgrades de la librairie sans recompilation des programmes. Toutefois, cela implique que la librairie doit être présente partout où on lance le programme.

`LD_LIBRARY_PATH` définit où chercher les librairies dynamiques (en plus des chemins par défaut).

Exemple:

```
mcodutti@lit1:echo $LD_LIBRARY_PATH
/usr/local/lib:/usr/ucblib:/usr/openwin/lib:/usr/dt/lib
mcodutti@lit1:ldd /bin/more
        libintl.so.1 =>  /usr/lib/libintl.so.1
        libc.so.1 =>     /usr/lib/libc.so.1
        libw.so.1 =>     /usr/lib/libw.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
mcodutti@lit1:ls -l /usr/lib/libc.*
-rw-r--r--   1 bin       1192328 Oct 25  1995 /usr/lib/libc.a
lrwxrwxrwx   1 root           11 Jun  7  1996 /usr/lib/libc.so -> ./libc.so.1
-rwxr-xr-x   1 bin         664048 Oct 27  1995 /usr/lib/libc.so.1*
```

La commande `ldd` indique toutes les librairies qui vont être impliquées par l'exécution d'un programme. La version en `.a` d'une librairie est la statique alors que la dynamique est un lien symbolique vers une version particulière de la librairie. Un programme ne pourra s'exécuter si la version présente est fort différente de celle prévue.

9.3.4 Le prompt

Le **prompt** (invite en français) est le petit texte que le shell affiche à l'écran pour signaler qu'il est prêt à recevoir la commande suivante. On le définit via la variable `prompt`

Exemple: (`tcsh`)

```
# set prompt="%n@%m:%/ "
mcodutti@lit1:/ULB/staff/mcodutti
```

9.3.5 TERM

Les opérations particulières sur un terminal (gras, souligné, effacer, se positionner, ...) se font via des caractères spéciaux (différents d'un type de terminal à un autre).

Les caractéristiques de chaque type de terminal sont stockées dans un fichier (`termcap`) ou une arborescence (`terminfo`)

La variable `TERM` donne l'identificateur du terminal utilisé (vt100, tvi925, xterm, ...).

9.3.6 Autres variables

USER login de l'utilisateur

HOME home directory

SHELL le nom du shell courant

PWD la directory courante

HOSTNAME nom de la machine

HOSTTYPE type de machine

EDITOR éditeur préféré (utilisé par certaines commandes)

TAPE nom du tape (utilisé par `mt` par exemple)

PRINTER nom de l'imprimante par défaut (utilisé par `lpr`)

UID user ID

GID group ID

9.4 Les alias

Un **alias** permet de donner un nom à une suite de caractères (un raccourci, en somme).

Pour définir un alias, la syntaxe est `alias nom=valeur` en `(ba)sh` (attention! pas d'espace autour de `=`) et `alias nom valeur` en `(t)csh`.

Pour voir la liste des alias définis, `alias`.

Pour effacer un alias, `unalias nom`

Exemple:

```
# touch brol1 ; touch brol2
# rm brol1
# alias rm "rm -i"
# rm brol2
rm: remove brol2 (y/n)? y
```

Tous les shells sauf `sh` donnent également un sens spécial au caractère `~`.

- `~` indique la home de l'utilisateur
- `~login` indique la home de l'utilisateur ayant ce login

Exemple:

```
# cd ~alice/Mail
# pwd
/home/alice/Mail
# whoami
mcodutti
# cd ~/brol
# pwd
/home/mcodutti/brol
```

9.5 Les quotes

Les **quotes** sont des opérateurs qui modifient le sens de leur argument. Il y a en 4 `'`, `\`, `"` et `` ` ``

- `'` groupe des caractères et supprime leur interprétation

  Exemple:

  ```
  # touch brol1 brol2
  # touch 'brol3 brol*'
  # ls -l
  -rw-r--r--   1 mcodutti        0 Feb 22 19:21 brol1
  -rw-r--r--   1 mcodutti        0 Feb 22 19:21 brol2
  -rw-r--r--   1 mcodutti        0 Feb 22 19:21 brol3 brol*
  ```

- \ idem mais uniquement sur le caractère suivant.

  Exemple: `# touch brol3\_brol4`

- " idem à ' mais les variables sont interpretées.

  Exemple:

```
# set var=brol
# touch '${var}1'
# touch "${var}2"
# ls -l
total 6003
-rw-r--r--   1 mcodutti        0 Feb 22 19:24 ${var}1
-rw-r--r--   1 mcodutti        0 Feb 22 19:24 brol2
```

- ` censé englober une commande. Elle est exécutée et remplacée par son résultat (output). Notons qu'il y a substitution de variables.

  Exemple:

```
# echo uname
uname
# echo `uname`
SunOS
# set var=uname
# echo `$var`
SunOS
```

9.6 Redirections

Chaque fichier ouvert par un processus se voit assigner un numéro (le *file descriptor*). Un processus commence avec 3 descripteurs :

0 standard input (par défaut, relié au clavier)

1 standard output (par défaut, relié à l'écran)

2 standard error (par défaut, relié à l'écran)

On peut toutefois rediriger ces file descriptors. Cette technique est couramment utilisée un UNIX pour lire les données et/ou sauver les résultats dans un fichier. Voyons comment cela fontionne en `(ba)sh` :

- `<file` : redirige l'input vers `file`
- `>file` : redirige l'output vers `file`. créé ou vidé.
- `>>file` : redirige l'output vers `file`. append.
- `2>&1` : associe le file descriptor 2 au fichier référencé par le file descriptor 1 (c-à-d redirige l'erreur vers l'output).
- `cmd1|cmd2` : l'output de `cmd1` est redirigé vers l'input de `cmd2`.

9.7 Substitution de fichiers

Lorsqu'il reçoit une commande, le shell interprete certains caractères et les remplace en fonction des fichiers présents dans le filesystem. Cela permet d'indiquer en raccourci un ou plusieurs fichiers.

- `*` : remplace $n \leq 0$ caractères (pas `.`)
- `?` : remplace 1 seul caractère (sauf `.`)
- `[abc]` : remplace `a` ou `b` ou `c`

Exemple:

```
# ls -a
. .. .cshrc .login res1 res1b res2 res2b res3
# ls *
res1 res1b res2 res2b res3
# ls res?b
res1b res2b
# ls res[12]
res1 res2
```

9.8 History

Le shell retient les commandes précédentes, ce qui permet d'y faire référence. Ce mécanisme n'existe toutefois pas en `sh`. Dans les autres shells, la portée est déterminée par une variable d'environnement.

- `history` : donne la liste des précédentes commandes
- `!57` : rappelle la commande numéro 57
- `!rm` : rappelle la commande la plus récente débutant par `rm`
- `!!` : rappelle la dernière commande
- `^s1^s2^` : rappelle la dernière commande où `s1` est remplacé par `s2`
- + beaucoup d'autres ...

Notons qu'en `bash` et en `tcsh`, on peut utiliser les flèches pour revenir aux commandes précédentes et les modifier.

9.9 Ordre d'évaluation d'une commande

Lorsque le shell reçoit une commande, il va l'interpreter en fonction de tous les mécanismes que l'on vient de voir (history, alias, ...). Il est bon de connaître

plus en détail comment il procède pour mieux comprendre le résultat de certaines commandes.

Voici, par exemple, l'algorithme d'interprétation du C-Shell :

1. Traiter l'history (`!, ^`)
2. Mettre la commande dans l'history
3. Séparer en mots
4. Alias
5. Traiter les redirections et backgroung (`<,>,|,&`)
6. Remplacer les variables par leur valeur (pas dans `''`)
7. Traiter le back quoting (`` ` ``)
8. Expansion des noms de fichiers
9. Exécuter la commande

9.10 Les paramètres

On peut passer des arguments à un script. Dans celui-ci, on y fait référence par

- `$1`, ..., `$9` pour les arguments 1 à 9
- `$*` pour tous les arguments
- `$0` pour le nom du programme

Exemple:

```
# cat brol
echo "$0-$2-$*"
# brol a b c d e
brol-b-a b c d e
```

9.11 Evaluation d'expressions *booléennes*

Il faut savoir que tout processus se termine par un appel à la fonction **`exit`** qui indique au système que le processus est terminé et peut devenir un *zombie*. Lors de l'*exit*, le processus fournit un nombre (le statut) indiquant comment cela s'est passé, destiné au processus père. Par convention, 0 indique que tout s'est bien passé et un nombre différent de 0 indique un code d'erreur. Ainsi, toute commande renvoie également un code de statut.

La commande **`test`** evalue une expression *booléenne* et renvoie un statut en conséquence. 0 si l'expression est vraie et une valeur non nulle si elle est fausse.

Elle permet de tester des expressions impliquant des nombres ou des chaînes de caractères mais également de tester des propriétés de fichiers (permission, existence,...)

La syntaxe est `test expression` ou de manière équivalente, dans un script, `[ expression]`.

9.12 Structure

Comme tout langage, le shell fournit des structures de branchement et de répétition. Nous n'entrerons pas dans les détails. Voici le canevas de quelques structures du Bourne shell :

```
# pour une alternative
if command;
then
...
else
...
fi

# pour boucler sur une liste
for name in list; do
...
done

# pour répeter une commande
while command; do
...
done
```

Exemple:

```
# Ce script affiche la liste des fichiers executables de la directory courante
for name in *; do
    if [ -x $name ]; then echo $name est executable; fi
done
```

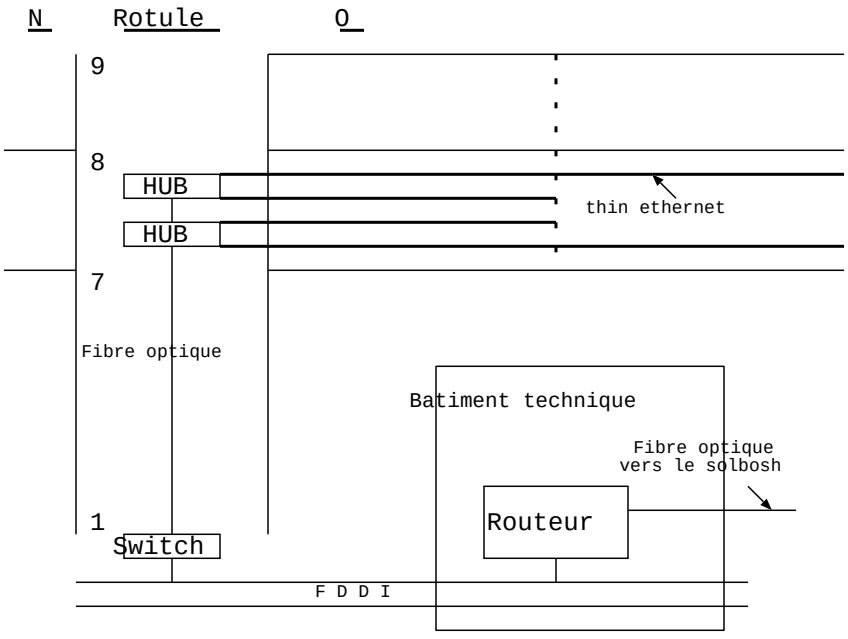
Chapitre 10

Internet

Dans ce chapitre, nous décrivons les concepts de base de l'Internet. Nous illustrerons nos propos en décrivant la structure du réseau dans le batiment NO. Nous verrons comment configurer une machine pour qu'elle utilise le réseau et nous aborderons le problème du routage. Toutefois, ceci n'est en aucun cas un cours complet sur le réseau.

10.1 Architecture du NO

Le schéma suivant reprend l'architecture réseau du batiment NO.



10.2 Connection physique

Les ordinateurs peuvent être physiquement reliés entre eux via, entre-autres,

- **Ethernet** (10 Mb/s)

  10BASE2. *Thin* ethernet. Câble noir qui traverse tous les bureaux. Max. 30 machines, 200 mètres.

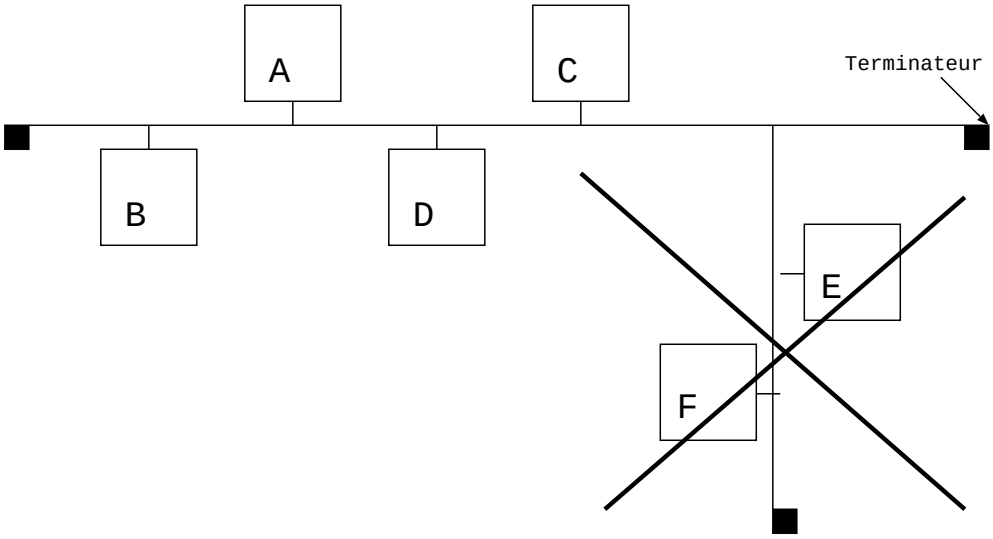
  10BASET. *Twisted pair*. Câble de téléphone.

  10BASEF. Fibre optique. Permet une plus grande longueur.

- **FDDI** (100 Mb/s). Fiber Distributed Data Interface. C'est, par exemple, ce qui est utilisé pour relier les machines importantes du Centre de Calcul. A base de fibre optique.

- Modem

10.3 Ethernet



La structure doit être linéaire. Il n'est donc pas permis d'y ajouter des machines déjà reliées entre-elles et de créer ainsi un embranchement. Le résultat est non prévisible. Il se peut que tout fonctionne correctement pour ces machines mais que des problèmes apparaissent sur le reste du câble.

Lorsqu'une machine émet, le signal parcourt tout le câble puis est absorbé par les terminateurs. La machine à qui il est adressé le lit lorsqu'il passe.

10.4 CSMA/CD

Ethernet fonctionne suivant le schéma CSMA/CD

- **Carrier Sense**: on peut se rendre compte si quelqu'un parle
- **Multiple Access**: tout le monde peut parler
- **Collision Detect**: on se rend compte si on interrompt quelqu'un

Lorsqu'une machine veut émettre, elle émet. Si il y a une collision, on attend un peu et on renvoie.

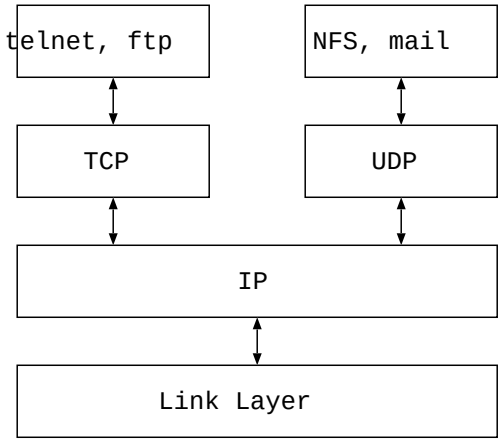
Ce qui est envoyé, c'est un *paquet*, un groupe de bits composé d'un en-tête, identifiant l'expéditeur et le destinataire, un corps de message et une fin de paquet.

10.5 Relier les ethernets

Les segments ethernets peuvent être reliés entre-eux pour accroître le réseau par

1. **repeater, HUB** : ont pour seule fonction de connecter des segments entre-eux en regénérant le signal. Tout ce qui est reçu d'un segment est renvoyé sur tous les autres segments.
2. **Switch, bridge** : Comme un repeater mais, en plus, sélectionne le segment sur lequel le paquet sera retransmis. Cette sélection se fait au niveau Ethernet et fonctionne donc pour Internet mais également pour les autres réseaux se basant sur Ethernet (IPX de Novell, EtherTalk de Apple, ...)
3. **Routeur, Gateway** : Fonctionne comme un switch mais effectue du routage au niveau du protocole IP (Internet). Ne peut donc traiter que des paquets IP. Un gateway peut se charger de convertir des paquets d'un protocole à un autre.

10.6 TCP/IP



Le *link layer* détermine la connection physique et peut être de l'ethernet, du FDDI, du satellite, ... Au dessus, vient se greffer le protocole **IP** (Internet Protocol) qui a pour tâche le transport de paquets IP d'une machine à l'autre.

Au niveau suivant, on trouve deux protocoles qui ont pour but d'acheminer des paquets d'un programme à un autre via le concept de *port*.

TCP. Transport Control Protocol. Il est orienté connection. Un chemin est établi entre deux programmes de deux machines et ce chemin leur est réservé pour l'échange de paquets. Il y a une certaine assurance quant à la bonne arrivée des paquets.

UDP User Datagram Protocol. Chaque paquet est envoyé au coup par coup. Ils peuvent emprunter des chemins différents, ne pas arriver dans l'ordre d'émission, se perdre.

Enfin, on trouve les différentes applications réseau qui se font soit en TCP (par ex: telnet, ftp) ou en UDP (ex: NFS, mail)

Un paquet TCP/IP est composé d'un en-tête IP, d'un entête TCP ou UDP selon le cas et d'un messsage destiné aux programmes. L'en-tête IP contiendra l'adresse de l'expéditeur et celle du destinataire. Si ce paquet est envoyé via Ethernet, il sera encapsulé dans un paquet ethernet.

10.7 Adresse IP

En IP, une adresse de machine est composée de 4 bytes.

Exemple: `134.184.15.9` est l'adresse de `is3`.

Le début de l'adresse indentifie le réseau sur lequel se trouve la machine alors que la fin identifie une machine particulière du réseau. Le tableau suivant

reprend les différents découpages entre partie réseau et partie machine. Un `N` indique que le byte fait partie du réseau alors qu'un `H` indique qu'il est dédié à la machine.

classe	byte 1	schéma	
A	1 à 126	`N.H.H.H`	Réseaux majeurs
B	128 à 191	`N.N.H.H`	Sites larges
C	192 à 223	`N.N.N.H`	réseaux de 250 machines
Autres			expérimental

La partie réseau est allouée par des organismes alors que la partie machine est de la responsabilité de l'administrateur du site.

10.8 Subnet

Les sites possédant des adresses de classe B (ou A) peuvent décider de raffiner les adresses en créant un *subnet*.

Exemple: à l'ULB, une adresse comme `164.15.125.1` est composée de la partie réseau (`164.15`), de la partie subnet (`125`) et du numéro de la machine (`1`).

10.9 Connection au réseau

La mise en service d'une interface réseau se fait via la commande `ifconfig`

.

```
ifconfig eth0 164.15.127.64 up netmask 255.255.255.0 broadcast 164.15.127.255
```

- **eth0**. nom de l'interface (souvent `ie0, le0, ln0, en0, eth0`). On trouve également `lo` ou `lo0` qui représente une carte fictive, en fait une boucle interne qui crée un réseau composé de la seule machine. Cela lui permet de dialoguer avec elle-même en interne sans passer par le câble.
- **adresse IP**
- **netmask** identifie la partie net/subnet du reste. Ici, on a 3 fois 255 pour indiquer que les 3 premiers bytes font partie de l'adresse réseau (net ou subnet).
- **broadcast** adresse pour les *broadcast*, c-à-d les envois à toutes les machines du même réseau. Utilisé par certains programmes pour poser une question lorsqu'on ne sait pas qui peut répondre ou pour poser une même questions à toutes les machines.

La commande `ifconfig eth0` renseigne sur la configuration courante.

10.10 Le routage

Le *routage* consiste à déterminer le chemin à prendre par un paquet pour aller d'une machine à une autre. Ce n'est pas une tâche facile vu la complexité du réseau. Il faut tenir compte des divers chemins possibles, réagir en cas de panne d'un noeud, de surcharge, ...

Cela se fait par une suite de décisions locales. Chaque machine dispose d'une table indiquant à qui envoyer le paquet en fonction de l'adresse IP. En général, une machine connaitra les machines proches et un routeur/gateway pour les autres adresses.

La commande `netstat -r` affiche la table de routage

Exemple:

```
mcodutti@mathpc4:netstat -r
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
localnet        *               255.255.255.0   U     0      0      312 eth0
loopback        *               255.0.0.0       U     0      0        9 lo
default         gate_127.ulb.ac 0.0.0.0         UG    1      0      496 eth0
```

A l'ULB, le gateway d'une machine sera toujours une machine dont l'adresse IP est formée des mêmes 3 premiers bytes et dont le dernier est `254`. Son nom sera `gate_x` où `x` est le byte 3 de l'adresse de la machine.

Pour définir la route par défaut:

```
route add net default 164.15.125.254 1}.
```

Sur certains OS, le fichier `/etc/defaultrouter` est lu au démarrage pour déterminer le routeur par défaut.

10.11 Tests

Pour tester un réseau, on peut utiliser `ping nom` ou `ping adresse_IP`. Il s'agit d'un test de bas niveau qui se contente de vérifier si l'autre machine est en vie. Si cela ne fonctionne pas avec le nom, il faut essayer avec l'adresse car le problème vient peut-être de l'incapacité de passer du nom à l'adresse IP.

Une autre commande est `netstat -i` qui donne des informations sur les paquets envoyés et reçus.

```
mcodutti@lit1:netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      12910693 0     12910693 0     0      0
le0   1500 164.15.123.0  lit1           754902829 6587  884774122 8254  54084972 0
```

On peut remarquer que le nombre de collisions est de l'ordre de 6%. On considère en général comme normal un taux inférieur à 10%.

Chapitre 11

Le Domain Name System

Un utilisateur préfère utiliser un nom de machine mais les programmes de bas niveaux fonctionnent avec des adresses IP uniquement. Il faut donc un mécanisme de traduction entre le nom et l'adresse IP.

Cela peut se faire de 2 façons :

1. par un fichier `hosts` (en local ou via *NIS*)
2. par le DNS (Domain Name System)

11.1 Le fichier `hosts`

Le fichier `hosts` reprend les machines et leur adresse IP (1 ligne par machine)
Exemple:

```
164.15.125.9     cso8 cso
164.15.123.2     lit1 lit
```

Si on donne plusieurs noms, les suivants sont des alias (noms équivalents)

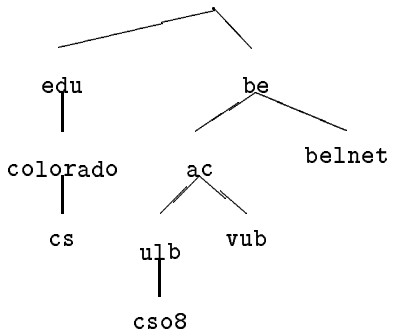
Ce fichier peut exister soit localement (`/etc/hosts`) soit via le réseau (*NIS*).

Avant l'introduction de DNS, cette technique était la seule utilisée. Vu la taille du réseau Internet actuel, un tel fichier ne peut plus contenir toutes les machines du monde. Cela ne serait jamais à jour et des doublons (deux machines ayant le même nom) ne serait pas rare. Il a donc fallu passer à un système plus souple et plus puissant, DNS.

11.2 DNS

Face à l'accroissement du nombre de machines, DNS introduit

1. une hiérarchisation des noms (Exemple: `cso.ulb.ac.be`)



   Un noeud correspond à un (sous)domaine alors qu'une feuille correspond à un une machine.

2. une responsabilité distribuée. A chaque niveau, il existe une autorité responsable de l'attribution des noms au niveau inférieur, ce qui assure l'unicité. A chaque niveau, il y a des *serveurs DNS* qui connaissent les machines du niveau inférieur et *la* machine immédiatement supérieure. Pour utiliser ce service, une machine doit se mettre en *client DNS*.

11.3 serveur DNS

Pour chaque noeud, on définit

1. un serveur principal qui contient la base de données Exemple: à l'ULB, c'est `resu1` qui tient ce rôle

2. (optionel) un ou des serveurs auxilliaires qui reprennent une copie de la base. La mise à jour est régulière. Exemple: au NO, on dispose de `plaine1, 164.15.125.1`

11.4 client DNS

Le client est configuré via le fichier `/etc/resolv.conf`
Exemple: pour les machines du NO, le fichier doit ressembler à

```
mcodutti@cso8:cat /etc/resolv.conf
domain          ulb.ac.be
nameserver      164.15.125.1
nameserver      164.15.59.200
```

Le fichier `hosts` doit toujours contenir les machines qui sont utilisées avant que DNS ne soit mis en route ou en cas de panne des serveurs

11.5 Ordre

DNS et `hosts` peuvent coexister. Dans ce cas, il s'agit de savoir dans quel ordre ils seront utilisés. En fait, cela dépend fortement du système

Solaris

Le fichier `/etc/nsswitch.conf` détermine l'ordre.
Exemple: S'il contient la ligne
`hosts:  nis [NOTFOUND=return] files`
Cela signifie qu'il faut d'abord consulter la NIS puis la DNS. Si la machine n'est pas trouvée, on s'arrête. Si, par contre, on n'a pas eu de réponse pour d'autres raisons (réseau en panne, ...), alors on consulte le fichier local.

HP

On consulte d'abord DNS puis NIS et enfin le fichier `hosts` local.

Irix

Il existe une commande supplémentaire dans `resolv.conf`
`hostresorder bind local`
qui indique qu'il faut d'abord demander à `bind`, le démon DNS puis consulter le fichier local.

SunOS

DNS ne peut fonctionner que si NIS fonctionne (à moins d'installer un *patch*). On a toujours l'ordre suivant: NIS puis DNS

Linux

L'ordre est déterminé par le fichier `/etc/host.conf` qui contient une ligne qui ressemble à
`order hosts, bind`

11.6 Test

La commande `nslookup` permet d'interroger le serveur DNS.
Exemple:

```
mcodutti@cso8:nslookup orca
Server:  plaine1.ulb.ac.be
Address:  164.15.125.1
```

```
Name:    orca.vub.ac.be
Address:  134.184.129.10
Aliases:  orca.ulb.ac.be
```

Le serveur s'identifie (nom et adresse) et donne tout ce qu'il sait à propos d'une machine (nom, adresse, alias)

Chapitre 12

Le Network Information Server

NIS, le Network Information Server, s'appelait autrefois *Yellow Pages* (YP). Le nom a du être modifié pour des raisons de copyright.

NIS est une base de données centralisée pour certains fichiers systèmes. Les plus importants sont `passwd` et `hosts`.

12.1 Mécanisme

NIS est composé d'un serveur maître, éventuellement de un ou quelques serveurs esclaves et, bien sûr, de clients. Les fichiers sont

- tenus à jour sur le serveur maître
- répercutés sur les serveurs esclaves
- consultés par les clients

Ce mécanisme

1. permet une gestion plus aisée vu qu'on ne doit maintenir qu'une version des fichiers
2. assure une cohérence entre les machines
3. dans le cas du fichier `passwd`, permet de disposer des mêmes comptes (avec mêmes mots de passe) sur toutes les machines.

12.2 Domaine

Un serveur sert un *domaine*. Ce domaine n'a rien à voir avec le domaine définit par DNS (`ulb.ac.be`)

```
mcodutti@lit1:domainname
```

```
lit
# pour modifier le domaine
mcodutti@lit1:domainname brol
```

Sur certains OS, le fichier `/etc/defaultrouter` est lu au démarrage pour déterminer le domaine.

12.3 Serveur maître

On peut utiliser comme fichiers distribués

1. les fichiers standards (Exemple: `/etc/hosts`)
2. des copies déposées ailleurs (Exemple: `/var/yp/nis/hosts`) ce qui est plus propre

Un serveur est généralement configuré lors de l'installation. Après, c'est plus compliqué et je ne vais pas le détailler ici.
Les démons prenant en charge ce service sont

`ypserv`	serveur
`ypxfrd`	propage la base vers les esclaves
`rpc.yppasswdd`	modification du mot de passe

12.4 Serveur esclave

Ils sont optionnels. On les utilise pour décharger le maître et/ou pour pouvoir reprendre le relais en cas de panne de celui-ci.

12.5 Client

Il peut être configuré pour

1. demander à un serveur bien particulier
2. prendre le premier qui répond pour un domaine donné

Le démon du client est `ypbind`

12.6 Password

Pour le password, le système va d'abord consulter le fichier local `/etc/passwd` *puis* la NIS pour autant que le fichier local se termine par la ligne

```
+::0:0:::
```

Pour changer le mot de passe sur la NIS, la commande est `yppasswd` au lieu de `passwd`.

12.7 NIS+

NIS+ est le successeur de NIS. Il répond aux mêmes objectifs mais introduit de la sécurité. Toutefois, c'est **TRES COMPLIQUE** et cela ne devient vraiment intéressant que pour de **GRANDS RESEAUX**.

Sun voulant pousser NIS+, dont il est l'instigateur, Solaris ne permet plus de configurer la machine en serveur NIS (elle peut toujours être client NIS). Toutefois, un package comblant cette lacune est disponible sur le Web.

Chapitre 13

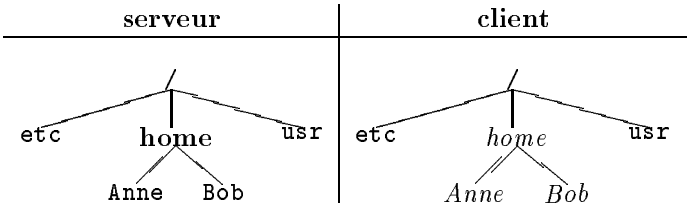
Le Network File System

NFS, le Network File System permet à une machine de *partager* une partie de ses fichiers, les rendant ainsi accessibles à partir d'autres machines.

Les avantages sont:

1. un gain de place
2. une même vision des fichiers sur toutes les machines.

13.1 Mécanisme



Le serveur *exporte* une directory (ici `/home`), rendant ainsi accessible tout ce qu'elle contient.

Le client *monte* explicitement cette directory. Ensuite, tout est transparent. L'accès à ces fichiers se fait comme si ils étaient présents localement.

13.2 Serveur NFS

Le serveur doit *exporter* (on dit aussi *partager*) ses fichiers. Cela se fait via le fichier `/etc/exports`

Exemple:

```
/usr/csoft     -access=cso5:cso6:cso7:cso8
/usr/share/man -ro
```

où on indique qu'on exporte `/usr/csoft` mais uniquement pour ces 4 machines. Par contre, `/usr/share/man` est exporté au monde entier mais en lecture seule.

Les démons impliqués sont

- **mountd** qui répond aux requêtes *mount* du client (voir section suivante)
- **nfsd** qui répond aux requêtes d'accès aux fichiers.

Pour visualiser ce qui est exporté, utiliser la commande `exportfs` . Après avoir modifié le fichier, il faut demander au démon de se resynchroniser. Cela se fait avec via la comamnde `exportfs -av`.

Linux

Sous Linux le format du fichier `/etc/exports` est différent. On trouvera plutôt des lignes du genre

```
/usr/csoft cso5(rw) cso8(rw)
```

pour indiquer qu'on donne l'accès en écriture à `cso5` mais uniquement en lecture pour `cso8`

Solaris

Solaris a décidé de ne pas faire comme tout le monde et utilise plutôt le fichier `/etc/dfs/dfstab` qui ressemble à

```
share -F nfs -o rw=cso8 /usr/csoft
```

Pour visualiser ce qui est exporté: `share`. Pour demander au démon de relire le fichier: `shareall`

13.3 Client NFS

Pour pouvoir utiliser des fichiers exportés, le client doit d'abord *monter* la directory. Pour cela, la commande est `mount`

Exemple: Pour monter la directory `/usr/share/man` de `cso8` sur `cso7`

```
mount cso8:/usr/share/man /usr/share/man
```

Comme toujours avec `mount`, la directory `/usr/share/man` soit exister sur `cso7`. Il s'agira en général d'une directory vide.

Pour visualiser les partitions NFS montées, `mount` ou `df` . Ces commandes donnent des informations sur tous les filesystems montés, pas uniquement via NFS.

Pour démonter un fichier, `umount /usr/share/man`.

13.4 Démarrage

La commande `mount` fonctionne au coup par coup. Pour définir tous les fichiers montés au démarrage, on utilise le fichier `/etc/fstab` (`/etc/vfstab` sous Solaris)

Exemple:

```
mcodutti@mathpc4:cat /etc/fstab
/dev/hda2    swap     swap    defaults   1   1
/dev/hda3    /        ext2    defaults   1   1
/dev/hda1    /dos     msdos   defaults   1   1
none         /proc    proc    defaults   1   1
cso7:/ULB/staff/mcodutti /ULB/staff/mcodutti nfs defaults 0 0
cso:/var/mail            /var/spool/mail     nfs defaults 0 0
```

Les premières lignes donnent les rôles des différentes partitions du disque dur. Les deux dernières concernent NFS.

13.5 Automount

Ce système permet de ne monter une directory que lorsqu'elle est explicitement utilisée. Elle sera démontée automatiquement après un certain temps de non utilisation.

Sur SUN, ce système s'appelle `automount`. Il existe également une version gratuite pour tous les systèmes: `amd`.

13.6 Rdist

Si on utilise NFS pour pouvoir ne gérer qu'une seule copie (qui se modifie peu) mais que l'espace disque n'est pas un problème alors `rdist` est une bonne alternative.

Ce système n'est pas standard mais est gratuit et facile à trouver.

Il fonctionne de la sorte. On définit une machine qui contiendra l'original d'un ensemble de fichiers. Les autres machines recevront automatiquement une copie fidèle. Cette copie est effectuée à la demande. Il peut donc y avoir parfois des différences entre les copies et l'original si celui-ci a été modifié depuis la dernière fois ou la copie a été demandée. Pour faciliter et accélérer le processus, `rdist` ne copie que les fichiers qui ont étés modifés depuis la dernière copie.

Son fonctionnement est déterminé par un fichier de configuration

Exemple: pour maintenir une même directory `/usr/local` sur les machines `cso5,cso6,cso7` et `cso8`, on aura sur `cso8`:

```
# cat /etc/distfile
(/usr/local) -> (cso5,cso6,cso7)
    install -R -y;
    notify mcodutti@cso.ulb.ac.be
```

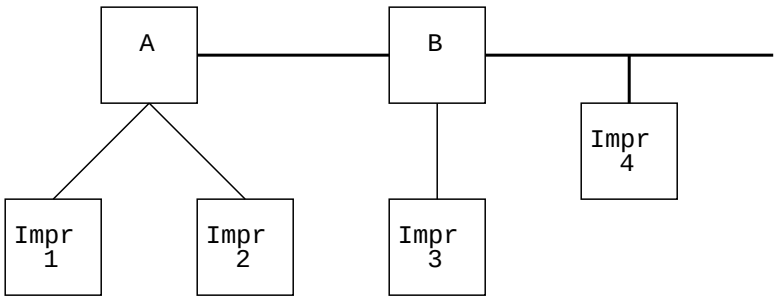
Il est alors lancé par la commande `rdist -f /etc/distfile`. Toute modification du système sera notifiée par mail.

Une bonne solution est de le lancer automatiquement toutes les nuits via le *cron*.

Chapitre 14

Les imprimantes

Dans UNIX, les imprimantes sont facilement partageables d'une machine à l'autre.



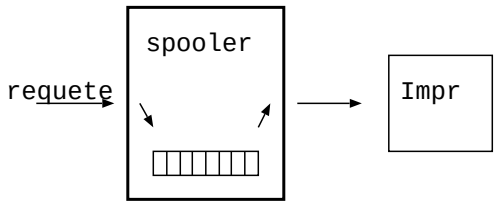
Cela sera totalement transparen pour l'utilisateur qui verra toutes les imprimantes de la même façon.

Le fonctionnement est fort différent enter BSD et Sys V . Les deux sont assez compliqués et nous recommandons d'utiliser un outil graphique si possible. Mais voyons quelques aspects communs au deux systèmes.

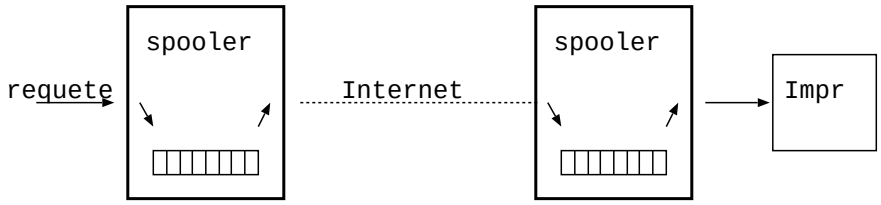
14.1 Le spooler

Le **spooler** est un système qui reçoit les requêtes d'impression, les stocke et les envoye à l'imprimante en séquence. Il y a (généralement) un spooler par imprimante.

local



remote



14.2 Langage

PDL Page Description Language

- matricielle : on indique les points à noircir
- laser ou jet d'encre : on envoye à l'imprimante une description de la page. Exemple: on indique qu'il y a une droite entre 2 points donnés de la page.

Postscript Un PDL très répandu dans le monde UNIX

PCL Un PDL développé par HP pour ses imprimantes. Très répandu dans le monde PC. Les imprimantes HP de haut de gamme comprennent également souvent le Postscript

DPI Dot Per Inch. Indique la résolution de l'imprimante. On retrouve souvent des résolutions de 300x300 et 600x600.

14.3 Le système BSD

Décrivons ici le système d'impression en BSD . Donnons d'abord un aperçu des différents composants.

- `lpr` envoye un fichier au spooler
- `lpd` est lé démon qui gère le spooler
- Les requêtes sont stockées dans une directory dont le nom ressemble à `/var/spool/printername/`

- Le fichier `/etc/printcap` décrit les imprimantes
- `lpq` permet de visualiser les fichiers en attente
- `lprm` permet d'enlever un fichier de la file
- `lpc` permet l'administration des imprimantes (stop, restart, ...)

14.3.1 printcap

`/etc/printcap` est un fichier ASCII qui décrit les imprimantes une après l'autre. Dans le cas d'un imprimante connectée localement.

Exemple: imprimante locale

```
math|lp|DEClaser 5100 at Math Gene Bunker:\
        :lp=/dev/ttyS1:\
        :br#19200:\
        :sd=/var/spool/lp/math:\
        :af=/var/spool/lp/math/acct:\
        :lf=/var/spool/lp/math/log:\
        :sb:\
        :sh:
```

Exemple: imprimante à distance

```
1|cso|Epson ex-1000 at CSO (N4). Connected to cso6:\
        :lp=:\
        :rm=cso6:\
        :rp=cso:\
        :sd=/usr/spool/lp/cso:
```

14.3.2 Commandes orientées utilisateur

Pour imprimer un fichier: `lpr -Pprinter fichier`. Si l'option `-P` n'est pas spécifiée, on prend la variable d'environnement `$PRINTER`. Si elle n'est pas définie, on imprime sur `lp`.

Pour visualiser les fichiers actuellement dans la file d'attente: `lpq -Pprinter`.

Exemple:

```
mcodutti@lit3:lpq -Pcso
cso is ready and printing
Rank   Owner      Job  Files                                 Total Size
1st    ssaeedni   657  standard input                        1908 bytes
```

Pour effacer un fichier de la file: `lprm -Pprinter job`

Exemple:

```
mcodutti@lit3:lprm -Pcso 657
```

14.3.3 Spooler

La directory désignée pour le spooling contient des fichiers de log, de statut, ...

Pour chaque fichier envoyé à l'impression, elle contient également

- Un fichier de la forme `cfA657...` qui contient de sinfos sur le fichier (qui, quand, ...)
- Un fichier de la forme `dfA657...` qui contient le fichier à imprimer proprement dit

`lpr` crée ces deux derniers fichiers et prévient `lpd` qui les envoye sur la bonne machine (si remote) ou les incorpore à la liste (si local).

14.3.4 Commande orientée administrateur

La commande est `lpc` dont la syntaxe est: `lpc command printer`, où `command` est

enable/disable pour ouvrir/fermer une file d'attente

start/stop pour déclencher/interrompre l'impression

down/up combine les 2 précédents

clean pour vider la file d'attente

topq pour placer un fichier en tête de liste

restart pour redémarrer le spooler en cas de problème

14.4 Le système Sys V

Sys V définit le concept de *classe* qui regroupe plusieurs imprimantes. On imprime dès lors sur une *destination* qui peut être une imprimante isolée ou une classe. Si on imprime dans une classe, le système choisit la première imprimante libre.

Pour imprimer un fichier: `lp -d printer fichier`. Si l'option `-d` n'est pas spécifié, on regarde la variable d'environnement `$LPDEST`. Si elle n'est pas définie, on prend l'imprimante par défaut (définie par `lpadmin -d`).

`lpstat` donne des infos sur le statut d'une imprimante
`cancel` permet d'enlever un fichier d'une file
`lpadmin` permet d'ajouter une imprimante

14.5 Les permissions

Le fichier `/etc/hosts.lpd` indique quelles machines ont accès aux imprimantes locales.

Exemple: pour pouvoir imprimer sur l'imprimante `lp` de la machine `A` à partir de `B`, la machine `A` doit posséder le fichier

```
# cat /etc/hosts.lpd}
B
```

Sous HP , le fichier est `/usr/spool/lp/.rhosts`. Sous Solaris , on utilise plutôt la commande `lpsystem`.

14.6 Quelques outils intéressants

JetDirect est une programme qui permet d'envoyer un fichier à une imprimante HP directement connectée au réseau. Il s'occupe de la file d'attente permet la gestion à distance de l'imprimante, ...

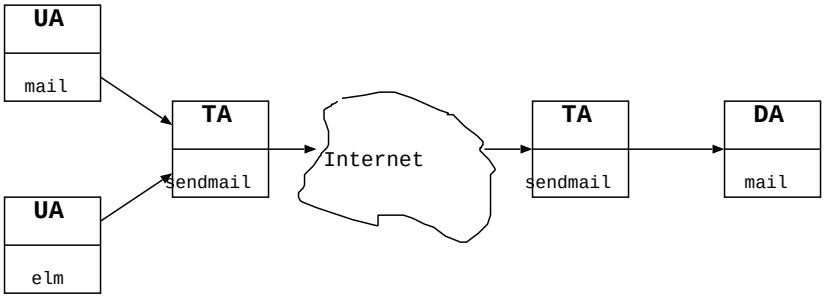
Ghostscript permet de convertir un fichier Postscript en de nombreux formats différents. Dans notre cas, l'utilité des de le convertir en PCL pour une impression Postscript.

mpage convertit de l'ASCII en Postscript. Permet également une certaine mise en page du Postscript (2 pages sur une seule par exemple).

Chapitre 15

Le mail

Voici un schéma décrivant le fonctionnement du mail



On y trouve

User Agent: interface entre l'utilisateur et le système

Transport Agent: s'occupe du transport du mail de machine en machine jusqu'à destination

Delivery Agent: place le mail dans la bonne boîte aux lettres. La boîte aux lettres d'un utilisateur sera `/var/mail/login` ou `/var/spool/mail/login`.

15.1 Adresses

Toute personne disposant d'un compte possède automatiquement une adresse électronique.

Exemple: Une personne possèdant le compte `mcodutti` sur `orca.ulb.ac.be` a pour adresse `mcodutti@orca.ulb.ac.be`.

L'adresse est également influencée par

1. Les alias de machines associées au mail

2. Les alias d'adresses globaux

3. Le fichier `.forward` de l'utilisateur

15.2 Aliases

Le fichier `/etc/aliases` définit des alias (parfois le fichier est `/usr/lib/aliases`). Les alias sont de la forme

```
marco:        mcodutti
techniciens:  mcodutti@ulb.ac.be, gpaquet@ulb.ac.be
prof:         :include:/etc/aliases.prof
suggestions:  "/dev/null"
info:         "|programme"
```

15.3 forward

Un utilisateur peut décider de *forwarder* ses mails reçus dans une boîte aux lettres c-à-d les envoyer automatiquement à une autre adresse.

Cela permet, par exemple, de tout concentrer dans une seule boîte si on en possède plusieurs.

Cela se fait via le fichier `.forward` dans la home

Exemple:

```
# cat ~/.forward
mcodutti@cso.ulb.ac.be, \mcodutti@ulb.ac.be
```

Le \ indique que le message envoyé à cette adresse ne devra plus passer par le mécanisme du `.forward` et permet ainsi d'éviter des boucles.

15.4 sendmail

`sendmail` est le démon qui s'occupe de smails. Il est configuré via le fichier `sendmail.cf` que je ne vais pas détailler ici car sa synatxe est vraiment très compliquée (un livre de 700 pages le décrit). Le but de ce fichier est de définir comment les adresses sont comprises et de déterminer qui peut s'en charger.

15.5 Tester

On peut obtenir des infos sur le chemin parcouru

Exemple:

```
root@cso8:mcodutti@lit.ulb.ac.be... Connecting to mailhost (ether)...
220-resu1.ulb.ac.be sendmail 8.6.8.1/3.12.0.ap (resu.test)
220 ESMTP spoken here
```

```
>>> EHLO cso8.ulb.ac.be
250-resu1.ulb.ac.be Hello cso8.ulb.ac.be, pleased to meet you
250-EXPN
250-SIZE
250 HELP
>>> MAIL From:<mcodutti@cso.ulb.ac.be> SIZE=56
250 <mcodutti@cso.ulb.ac.be>... Sender ok
>>> RCPT To:<mcodutti@lit.ulb.ac.be>
250 <mcodutti@lit.ulb.ac.be>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 PAA16766 Message accepted for delivery
>>> QUIT
221 resu1.ulb.ac.be closing connection
mcodutti@lit.ulb.ac.be... Sent (PAA16766 Message accepted for delivery)
```

15.6 Encodage

Le mail fonctionne essentiellement en 7 bits ce qui empêche d'envoyer directement du binaire ou des caractères acentués. Les systèmes modernes contournent le problème par un codage automatique des mails (tel MIME). Sur les vieux systèmes, on utilise la commande **uuencode** qui transforme des binaires et ASCII 8 bits en ASCII 7 bits.

Exemple:

```
# uuencode brol brol1 >brol.uue
# cat brol.uue
begin 644 brol1
M<F]O= ...
...
end
# uudecode brol.uue
# ls
brol brol1 brol.uue
```

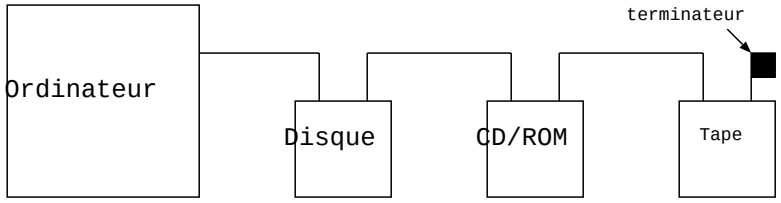
Chapitre 16

les disques

Dans ce chapitre, nous allons voir comment installer un disque SCSI.

16.1 le modèle SCSI

Le standard SCSI (Small Computer Standard Interface) est très répandu dans le monde des stations de travail. Il s'agit du même standard que celui utilisé avec les `Macintosh`.



On peut installer jusqu'à 7 périphériques (disques, lecteurs de bandes ou de CDROM) en série. Chaque périphérique possède un numéro SCSI (0 à 6), souvent réglable à l'arrière de l'appareil. Le terminateur est parfois interne au dernier élément de la chaîne.

La technologie SCSI évolue beaucoup ces derniers temps. Nous avons

- SCSI : 5 Mbytes/s
- SCSI-2
  - Fast: 10 Mbytes/s
  - Fast-Wide: 20 Mbytes/s (16 bits)
  - Differential: Permet d'allonger la chaîne

– Ultra: 40 Mbytes/s

On ne peut pas mélanger du différentiel avec du non-différentiel. Par contre, on peut mélanger du SCSI avec du SCSI 2, fast, wide mais on fonctionne au niveau le plus lent.

16.2 Installer un disque

Les étapes de l'installation d'un disque sont

1. Assigner un numéro SCSI et le brancher
2. Le formatter
3. Le partitionnner
4. Créer les filesystems
5. Le vérifier
6. Configurer le système pour que les partitions soient montées automatiquement

16.2.1 Formatter un disque

La commande est `format` (`mediainit` sous HP) (`fx` sous Irix). L'OS contient une table des disques les plus courants. Si le disque n'est pas repris dans cette table, il faudra aider `format` en indiquant la géométrie du disque (têtes, cylindres, ...). Le formattage permet de détecter les mauvais blocks et les met de côté. Cela peut prendre beaucoup de temps.

16.2.2 Partitions

Un disque est divisé en *partitions* logiques. Cela permet de voir un disque comme un groupe de disques indépendants. En général, on aura également une *fausse* partition qui reprendra tout le disque.

Partitionner un disque permet d'isoler les problèmes, d'avoir un meilleur contrôle sur l'espace occupé et de faciliter les backups.

Elles sont créées avec la même commande utilisée pour formatter les disques (`format`).

16.2.3 Créer un filesystem

Avant de pouvoir utiliser une partition, il faut y créer un filesystem vide (`newfs -v /dev/rsd0g`).

Pour pouvoir booter sur une partition, il faut la rendre bootable. La commande sous Solaris est `installboot`.

16.2.4 Vérifier les disques

La commande `fsck` examine le filesystem d'une partition, détermine les erreurs et tente de les réparer. C'est cette même commande qui est lancée au démarrage de la machine. Parfois, il demande s'il peut détruire des informations qui semblent incohérentes.

Cela peut s'avérer nécessaire après un arrêt brutal de la machine ou un choc.

Chapitre 17

Fichiers de trace

De nombreux programmes (parties du système) sauvent dans des fichiers (log files) des informations sur ce qu'ils font. Ils sont utilisés en cas de problème. Malheureusement, leurs localisations et leurs noms sont tout-à-fait incohérents d'un système à l'autre mais également au sein d'un même système.

Exemple: On trouve souvent ces fichiers dans `/var/adm/`, `/var/log/`, `/var/cron/`, `/usr/adm`, ...

Le fichier *log* utilisé par un programme peut être

1. hardcodé
2. une option de démarrage
3. déterminé par un fichier de configuration
4. définit via *syslog*.

17.1 syslog

syslog est un mécanisme centralisé de gestion des messages de trace. Un fichier de configuration détermine ce qu'il faut en faire en fonction de

1. sa provenance (type) : kern, mail, lpr, user, ...
2. son importance (niveau) : emerg, error, warning, debug, ...

Un message peut être

1. jeté
2. envoyé sur les écrans des utilisateurs connectés
3. sauvé dans un fichier
4. passé à une autre machine (centralisation)

Le démon qui gère ce système est `syslogd`.

17.2 syslog.conf

Le comportement de syslog est déterminé par le fichier `/etc/syslog.conf` qui est un tableau à 2 colonnes. La première indique les messages concernés, la deuxième ce qu'il faut en faire.

Exemple:

```
*.emerg                       /dev/console
*.warning; daemon,auth.info   /var/adm/messages
local0.*                      @loghost
```

Si on modifie ce fichier, il faut prévenir `syslogd` par un `kill -1`.

17.3 Gestion des logs

Les fichiers de trace ont tendance à croître fortement et à saturer les disques. Pour prévenir cela, il y a plusieurs possibilités

1. On peut décider de ne rien garder
2. On peut les remettre à zéro régulièrement
3. On peut faire une rotation et garder quelques fichiers
4. On peut les archiver

Remarquons qu'il est délicat de remettre à zéro un fichier log en cours d'utilisation. Il vaut mieux tuer et relancer le programme qui l'utilise.

Chapitre 18

Sécurité

Le système Unix a été conçu par des chercheurs pour des chercheurs. La problématique de la sécurité n'a pas été envisagée à ses débuts. Unix est essentiellement binaire. On est un utilisateur normal ou un super-user avec tous les droits. Il faut bien se rendre compte que de nombreux trous existent, ont été trouvés, seront trouvés, parfois corrigés.

Ce chapitre n'est pas un cours théorique et approfondi sur la sécurité en Unix. Nous allons plutôt examiner des points élémentaires du système, facile à contrôler mais sensibles par rapport à la sécurité.

18.1 Les mots de passe

En dehors des considérations sur la qualité d'un mot de passe que nous avons vues dans le chapitre correspondant, on peut mettre en évidence les points suivants

- Ne laisser aucun compte sans mot de passe, ce que l'on peut vérifier avec une commande du genre `grep :: /etc/shadow`
- Utiliser le fichier `/etc/shadow` si possible.
- Essayer soi-même de cracker les mots de passe, par exemple avec `crack`.

18.2 Le `setuid` bit

Les programmes appartenant à `root` et ayant le `setuid` bit s'exécutent avec les permissions du `root`, ce qui est dangereux.

On conseille en tout cas de ne pas utiliser ce bit pour des scripts dont l'environnement, et donc le comportement, est facilement modifiable.

Pour détecter tous les fichiers du système ayant ce bit, on peut utiliser la commande `find / -user root -perm -4000 -print`

Certains fichiers ont ce bit dès l'installation du système. Ils sont *censés* être sûrs.

18.3 su

La commande `su` permet à un utilisateur normal d'obtenir les permissions du `root`. Il faut bien sûr fournir le mot de passe pour cela. La commande `su -` est encore plus forte. Elle est équivalente à un *login*.

L'avantage de cette commande est qu'il est possible d'empêcher à quiconque de rentrer directement en `root` à distance. Il sera obligé de se connecter d'abord en tant qu'utilisateur normal puis d'initier un `su`. Ceux-ci étant loggés dans un fichier, il est possible de savoir qui est devenu `root` et quand.

Notons qu'il est aussi toujours possible de restreindre les utilisateurs qui ont accès à cette commande.

18.4 $PATH

Le `$PATH` est une variable cruciale

1. Les directory indiquées ne doivent pas être en écriture pour tout le monde.
2. Ne pas mettre `.` (le point représente la directory courante) dans le `$PATH` du `root`. Ce que l'on peut vérifier par la commande `echo $PATH`.

18.5 rsh

La commande `rsh B commande` initiée sur la machine `A` permet d'exécuter la `commande` sur la machine `B` à partir de `A`.

Il faut pour cela que la permission soit donnée par `B` ce qui peut se faire de deux façons

- Un fichier `/etc/hosts.equiv` qui contienne `A` ce qui donne la permission à tout le monde sur `A` d'exécuter une commande sur `B`.
- Un fichier `.rhosts` dans la home d'un utilisateur ce qui ne donne la permission que pour lui.

Cette facilité est bien sûr très dangereuse car le `root` sur `A` devient `root` sur `B`.

Ce mécanisme est également celui mis en oeuvre pour la commande `rlogin` qui permet une connection à distance.

18.6 Outils

Il existe quelques programmes qui inspectent le système et fournissent un rapport détaillé des failles potentielles. Citons `COPS` qui vérifie les problèmes au sein d'une machine et `SATAN` qui inspecte les failles potentielles liées aux connections réseaux.

Ces logiciels sont toutefois difficiles à installer. Ils doivent connaître précisement votre version de système pour pouvoir identifier ce qui est dangereux de ce qui ne l'est pas. De plus, les rapports qu'ils fournissent sont longs, difficiles à lire et pas toujours très pertinents.

18.7 Divers

Citons d'autres problèmes possibles

- Il faut vérifier les permissions des directory importantes et de celles que l'on crée (voir `umask`)
- Les backups doivent être surveillés si l'information est secrète car n'importe qui peut lire une bande.
- NIS n'est pas sûr
- Sur certains OS, une directory exportée via NFS sera par défaut accessible dans le monde entier.
- Si la machine tourne un FTP anonyme, il faut bien vérifier les permissions des directory accessibles.

Annexe A

Lexique

Nous reprenons ici un lexique de quelques acronymes et termes informatiques que nous utilisons dans ces notes.

background un processus est en background (ou tâche de fond) s'il travaille de lui-même sans être attaché à un terminal. Cela peut être un gros programme lancé par un utilisateur ou un processus système.

booter signifie démarrer (une machine). Cela s'effectue souvent par étapes: un programme va charger un programme un peu plus gros qui ... (*bootstraping*)

BSD désigne ici l'ensemble des OS de type BSD . cf chapitre 1.

CPU (Central Processessing Unit). Le processeur.

démon Un démon est un processus, partie de l'OS, dédié à une tâche bien spécifique et tournant en tâche de fond.

filesystem (système de fichiers en français) est la structure arborescente qui reprend tous les fichiers (au sens large).

gid (Group ID) numéro de groupe principal d'un utilisateur.

Home directory Directory réservée à un utilisateur pour y mettre ses fichiers. C'est la directory dans laquelle on se trouve au départ lorsqu'on se connecte.

kernel (noyau en français). C'est le coeur du système.

login nom par lequel le système connaît un utilisateur.

moniteur c'est un programme en PROM qui s'exécute au démarrage et a pour mission de charger UNIX.

monter monter une directory, c'est l'incorporer quelque part dans le filesystem.

OS (Operating System). UNIX est un OS, un système d'exploitation.

partition un disque dur est généralement divisé en *partitions* qui ne sont jamais que des disques logiques à l'intérieur d'un disque physique.

pid numéro identifiant un processus.

processus instance d'un programme tournant pour le moment.

PROM ou ROM, EPROM, EEPROM sont des mémoires en lecture seule (en opposition à la RAM).

prompt séquence de caractères apparaissant à l'écran lorsqu'on utilise un programme interactif pour indiquer que le programme est prêt à recevoir notre prochaine commande.

`root` login de l'administrateur UNIX.

script ensemble de commandes regroupées dans un fichier que l'on peut lancer d'un coup.

SCSI (Small Computer Standard Interface). Standard pour la connection de disques, tapes et CD/ROM sous UNIX (comme sous Mac).

shell programme de communication entre l'utilisateur et l'OS. C'est le shell qui affiche un prompt, permet de taper une commande et l'exécute. Il existe le Bourne-shell (`sh`), le C-shell (`csh`), le Bourne Again Shell (`bash`), ...

signal un signal est envoyé à un processus pour lui indiquer qu'un événement précis s'est passé.

single user mode spécial de UNIX dans lequel, seul le root peut se connecter. Utilisé pour des opérations de maintenance.

Super User nom par lequel on désigne parfois l'administrateur UNIX.

Sys V désigne ici l'ensemble des OS de type Sys V . cf chapitre 1.

tâche de fond voir *background*

uid (User ID) numéro unique d'un utilisateur.

Index