# Administration de systèmes UNIX
# Formation ARS 2010 – 2011

**Partie 2**

**Thierry Besançon**




**Formation Permanente de l'Université**
**Pierre et Marie Curie**

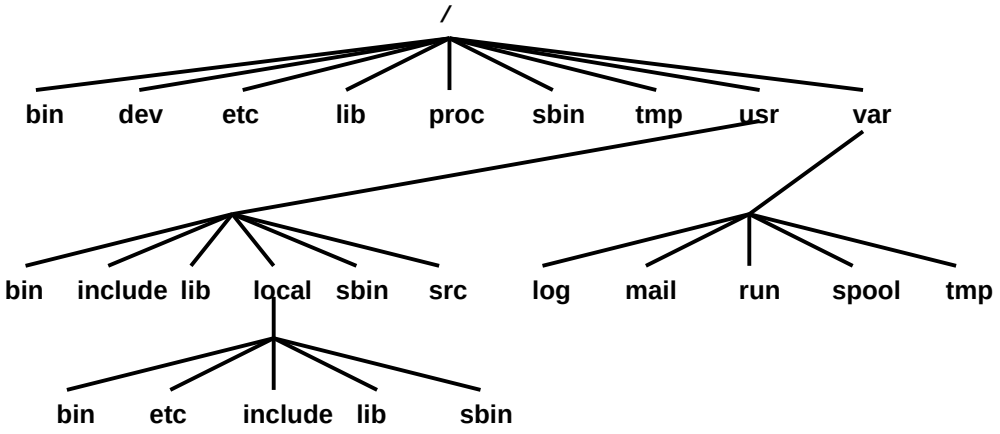
Chapitre 1

# Contenu d'une arborescence type d'un système UNIX





Exemple :




Pourquoi ce chapitre sur l'arborescence ?

1. Pour connaître le contenu des principaux directories
2. Pour connaître l'utilité des principaux directories
3. Pour savoir configurer les PATH (utilisateurs + root)
4. Pour surveiller les directories système pouvant grossir en raison de fichiers log.

Longtemps la hiérarchie a été non standard et très dépendante de chaque OS.
⇒ Difficulté de portage de shell scripts et autres applications.
⇒ Difficulté de gestion en environnement hétérogène.

Exemple :

| | |
|---|---|
| FreeBSD 2.2.x | `/usr/sbin/chown` |
| HP-UX 9.07 | `/bin/chown` |
| SunOS 4.1.4 | `/usr/etc/chown` |

Une certaine convergence apparaît actuellement (effet des différents standards Posix and co ? ou raréfaction des UNIX ?).





**Solution du lien symbolique**
Ne pas oublier la commande "**ln -s fichier1 fichier2**".

| | |
|---|---|
| IRIX 5.3 | `/usr/bsd` |
| SunOS 4.1.4 | `/usr/ucb` |

On fera par exemple sur la machine SGI :

```
% ln -s /usr/bsd /usr/ucb
% ls -l /usr
drwxr-xr-x    2 root     sys        512 Aug 28  1996 bsd/
lrwxr-xr-x    1 root     sys          3 Aug 22  1996 ucb -> bsd/
% echo $PATH
/usr/local/bin:/bin:/usr/ucb:/usr/bin:/usr/sbin:/etc:/usr/etc
% type rlogin
rlogin is /usr/ucb/rlogin
```

Se reporter au site « `http://www.pathname.com/fhs/` »

Document FHS version 2.3 :
« `http://www.pathname.com/fhs/pub/fhs-2.3.pdf` »

**Lecture indispensable !**



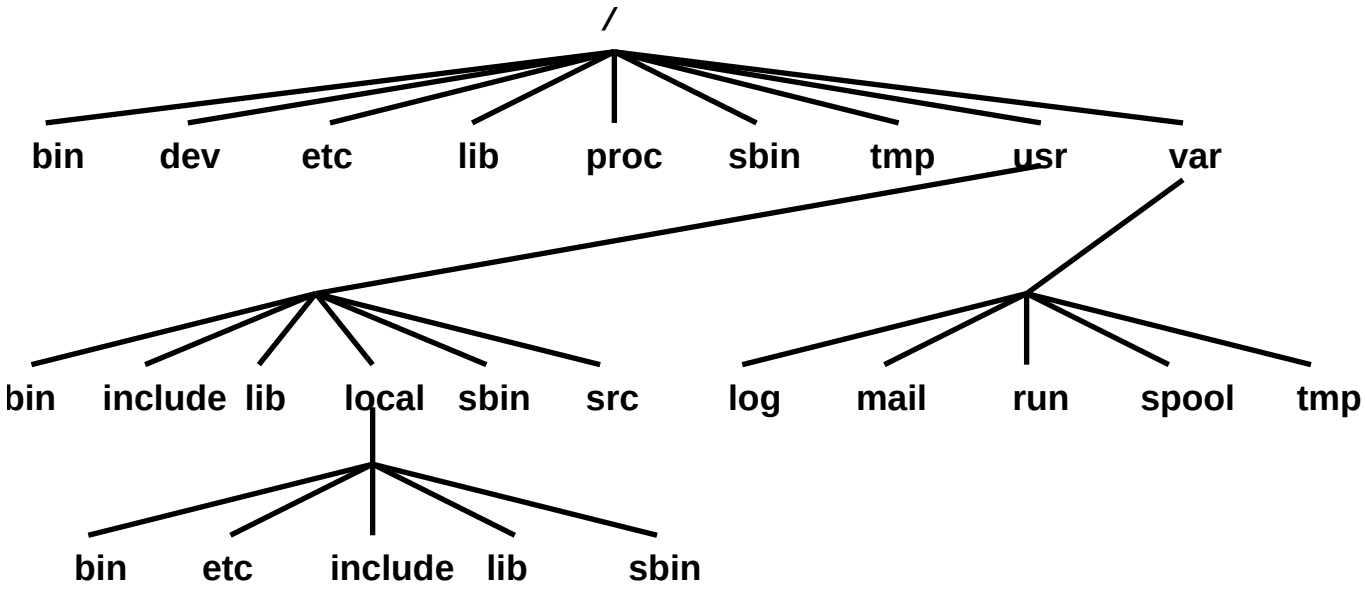


Quelques notions importantes :

- Répertoires système / Répertoires utilisateurs
  ⇒ différences de `PATH`
- Répertoires partageables / Répertoires non partageables
  ⇒ stations diskless
- Répertoires figés / Répertoires évoluant dans le temps
  ⇒ partitions en read-only, partitions sur CDROM

Possibilité de mixer ces caractéristiques.
Exemple :

| | Répertoires partageables | Répertoires non partageables |
|---|---|---|
| Répertoires figés | `/usr`, `/opt` | `/etc`, `/boot` |
| Répertoires évolutifs | `/var/mail`, `/var/spool/lpd` | `/var/run`, `/var/lock` |








**« /sbin » = system binaries**

Il contient les binaires système primordiaux utilisés pendant le démarrage de la machine.
(historiquement, ces binaires étaient dans « /etc »)
Il réside dans la même partition que « / » ce qui garantit sa disponibilité à ce moment-là où d'autres partitions ne sont pas encore disponibles.

Exemple non exhaustif de commandes :
arp, dump, fsck, halt, ifconfig, init, ldconfig, mkfs, restore, shutdown, swapon, etc.

## « /bin » = binaries

Il contient les binaires primordiaux utilisés pendant le démarrage de la machine.
Il réside dans la même partition que « / » ce qui garantit sa disponibilité à ce moment-là où d'autres partitions ne sont pas encore disponibles.

Exemple non exhaustif de commandes :
`bash`, `chmod`, `chown`, `cp`, `date`, `dd`, `df`, `echo`, `hostname`, `kill`, `mkdir`, `mknod`, `more`, `mv`, `ps`, `pwd`, `rm`, `rmdir`, `umount`, etc.





## « /usr/sbin » = user level system binaries

Il contient des binaires système (d'où le « s » dans son nom) non primordiaux et plus utilisés pour la gestion au jour le jour de la machine.
Il réside dans une même partition pouvant ne pas être disponible pendant les premières étapes du démarrage.

Exemple non exhaustif de commandes :
`adduser`, `in.rlogind`, `inetd`, `lpd`, `lpc`, `pwck`, `sa`, `showmount`, `traceroute`, `vipw`, etc.

**« /usr/bin » = user level binaries**

Il contient la plupart des commandes UNIX pour utilisateurs (plusieurs centaines en général).

Exemple non exhaustif de commandes :
`date`, `grep`, `ln`, `perl`, `sed`, `vi`, `who`, etc.





**« /usr/local/sbin » = local user level system binaries**

Il contient les commandes UNIX d'administration propres au site ou propres à la machine.

**« /usr/local/bin » = local user level binaries**

Il contient la plupart des commandes UNIX utilisateurs installées localement.





**« /lib » = libraries**

Il contient en général les librairies dynamiques primordiales pour le fonctionnement du système.

**ATTENTION : ne pas effacer les librairies dynamiques de ce répertoire, sinon AIE !**

Linker avec « -lutilitaires » revient à linker avec « /lib/libutilitaires.a » ou « /usr/lib/libutilitaires.a » (sauf option spécifique à « gcc »)

Exemple non exhaustif de fichiers :
ld.so, libc.so.5, libm.so.5, etc.

En général, on n'y trouve pas de librairie statique « .a ».

**« /usr/lib » = user level libraries**

Il contient les librairies dynamiques non primordiales au fonctionnement du système ainsi que les librairies statiques « .a » de programmation en langage C.

Linker avec « -lutilitaires » revient à linker avec « /lib/libutilitaires.a » ou « /usr/lib/libutilitaires.a » (sauf option spécifique à « gcc »)

Exemple non exhaustif de fichiers :
libm.a, libm.so, libndbm.a, libtermcap.a, etc.





**« /usr/local/lib » = local user level libraries**

Il contient les librairies dynamiques ou statiques installées localement.

**« /usr/include » = user level include files**

Il contient les fichiers « .h » de programmation en langage C.

« <exemple.h> » fait référence à « /usr/include/exemple.h »
(sauf option de compilation explicite à « gcc »)





**« /usr/local/include » = local user level include files**

Il contient les fichiers « .h » installés localement.

« <utilitaires.h> » fera référence à
« /usr/local/include/utilitaires.h » via une option à « gcc » du genre « -I/usr/local/include ».

**« /etc » = system level config files and etc.**

Il contient les fichiers de configuration des divers utilitaires primordiaux assurant le fonctionnement du système.

Exemple non exhaustif de fichiers :
`group`, `passwd`, `motd`, `hosts`, `hosts.allow`, `hosts.deny`, `fstab`, `inittab`, `printcap`, `resolv.conf`, `syslog.conf`, etc.





(en anglais *Message Of The Day*)

Il contient des informations mises en place par l'administrateur pour les utilisateurs. Les informations sont affichées lors du login de l'utilisateur.

Pour ne pas afficher le contenu de « `/etc/motd` », créer un fichier « `$HOME/.hushlogin` » (un fichier vide suffit).

Non disponible sur SOLARIS par exemple.

Ce fichier contient des informations sur le nom de l'UNIX et/ou sa version. Très vrai dans le monde LINUX.

Par exemple :

```
Mandrake Linux release 10.0 (Official) for i586
Kernel 2.6.3-19mdkentreprise on a Dual-processeur i686 / \ 1
```





**« /usr/local/etc » = local system level config files and etc.**

Il contient les fichiers de configuration de divers utilitaires non nécessaires au fonctionnement du système.

**« /usr/man » = system manual**

ATTENTION : on rencontre de plus en plus à la place « /usr/share/man ».
Le répertoire contient les fichiers de la documentation man du constructeur, classée par sous-répertoires : « man1 », « man2 », « man3 », ...

Principe : un fichier par commande dans le man.

Tendance : compression des fichiers de man
Par exemple : « man ssh » → « /usr/share/man/man1/ssh.1.bz2 »

Anciennement : « man ssh » → « /usr/man/man1/ssh.1 »





Fichier de configuration de MAN : « /etc/man.config »

Variable d'environnement de configuration de MAN : « MANPATH »

Les pages de documentation sont au format NROFF de l'ancien traitement de texte « nroff » abandonné sauf pour les pages de manuel (SOLARIS 10 fichiers XML) :

```
.TH SSH 1 "September 25, 1999" ""
.SH NAME
\fBssh\fP
\- OpenSSH SSH client (remote login program)
.SH SYNOPSIS
.br
\fBssh\fP
[\fB\-l\fP \fIlogin_name\fP]
\fIhostname\fP | \fIuser@hostname\fP
[\fIcommand\fP]
...
```

En fait « man ssh » =
« bzcat /usr/share/man/man1/ssh.1.bz2 | nroff | more »
En fait (bis) : « man ssh » =
« bzcat /usr/share/man/man1/ssh.1.bz2 | nroff | $PAGER »





**« /usr/local/man » = local system manual**

ATTENTION : on rencontre de plus en plus à la place
« /usr/local/share/man ».

**« /dev » = system devices**
Il contient les fichiers spéciaux du système permettant d'accèder aux ressources physiques de la machine comme les disques durs, la mémoire RAM etc.

Pas de « /usr/dev » ni « /usr/local/dev » !

**Les droits d'accès aux fichiers spéciaux dans /dev sont très importants.**
Une précaution : conserver une trace écrite d'un « ls -l » de « /dev ».





2 approches historiques :

- ancienne approche : les fichiers spéciaux sont générés manuellement via les scripts « /dev/MAKEDEV » et « /dev/MAKEDEV.local »
- approche moderne : les fichiers spéciaux sont générés automatiquement (faire en plus « boot -r » sur SOLARIS).

Quelle est l'organisation du répertoire `/dev` ?

- organisation en vrac (FreeBSD, LINUX)
- organisation structurée (SOLARIS, HP-UX, etc.) en sous répertoires
  Exemple non exhaustif de SOLARIS :

| **Répertoire** | **Contenu** |
|---|---|
| `/dev/rdsk` | disques en mode caractère |
| `/dev/dsk` | disques en mode bloc |
| `/dev/rmt` | bandes magnétiques |
| `/dev/pty` | pseudo-terminaux esclaves |
| `/dev/ptym` | pseudo-terminaux maîtres |





Plusieurs fichiers communs à tous les UNIX :

| | |
|---|---|
| `/dev/audio` | le device de la carte audio de la machine |
| `/dev/dsp` | le device de la carte audio de la machine (Digital Signal Processor) |
| `/dev/console` | le device de la console texte de la machine |
| `/dev/kmem` | mémoire du noyau UNIX |
| `/dev/log` | canal des messages système |
| `/dev/mem` | mémoire de la machine |
| `/dev/null` | trou noir |
| `/dev/tty` | le device virtuel de votre terminal texte |
| `/dev/zero` | générateur de caractères de code ASCII 0 |

On peut vouloir se débarasser d'une partie de l'affichage.

Solution inefficace :

```
% application > /tmp/temporaire
...
% rm /tmp/temporaire
```

La solution est de rediriger vers `/dev/null` :

```
% application > /dev/null
...
```

C'est un fichier spécial (logique car dans `/dev`...) :

```
crw-rw-rw-   1 root     sys       13,  2 Oct 19 01:13 /dev/null
```

**« `/dev/null` » est indispensable dans la vie de l'administrateur système.**





Exemple :

```
C:\DOCUME~1\besancon\MESDOC~1>application.exe 2>NUL
...
```

La console texte de la machine est accessible via le device `/dev/console`



Classiquement, les messages importants s'affichent sur la console.

(sur un poste LINUX en mode X Windows, faire `Ctr Alt F1` pour passer sur la console en mode texte)





**« `/proc` » = system process**

Il contient une représentation sous forme de fichiers virtuels et directories virtuels d'informations de la machine UNIX.

Utilité ?
⇒ pour le programmeur système qui utilisera les appels standard (et non pas des appels spécifiques comme la `libkvm` ou des `ioctl()` spécifiques)

Implémentation de `/proc` propre à chaque UNIX.

Représentation virtuelle de :

- informations sur les processus
- informations sur le hardware
- informations sur le noyau

## ◇ Exemple : informations sur les processus

```
% echo $$
23247
% cd /proc/23247
% ls -l
total 0
-r--r--r--   1 besancon sysadmin    0 Sep 22 14:30 cmdline
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 cwd -> [0001]:15235153
-r--------   1 besancon sysadmin    0 Sep 22 15:45 environ
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 exe -> [0801]:10251
dr-x------   1 besancon sysadmin    0 Sep 22 15:45 fd
pr--r--r--   1 besancon sysadmin    0 Sep 22 15:45 maps
-rw-------   1 besancon sysadmin    0 Sep 22 15:45 mem
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 root -> [0801]:2
-r--r--r--   1 besancon sysadmin    0 Sep 22 14:29 stat
-r--r--r--   1 besancon sysadmin    0 Sep 22 15:45 statm
-r--r--r--   1 besancon sysadmin    0 Sep 22 15:45 status
```





## ◇ Exemple : informations sur le hardware

```
% ls -l /proc/pci
-r--r--r--    1 root     root            0 Nov  2 18:52 /proc/pci

% cat pci
PCI devices found:
  Bus  0, device   0, function  0:
    Host bridge: VIA Technologies, Inc. VT82C693A/694x [Apollo PRO133x] (rev 196).
      Prefetchable 32 bit memory at 0xd0000000 [0xd3ffffff].
  Bus  0, device   1, function  0:
    PCI bridge: VIA Technologies, Inc. VT82C598/694x [Apollo MVP3/Pro133x AGP] (re
      Master Capable.  No bursts.  Min Gnt=12.
...
```

```
% cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 8
model name      : Pentium III (Coppermine)
stepping        : 10
cpu MHz         : 1002.214
cache size      : 256 KB
Physical processor ID   : 0
Number of siblings      : 1
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pa
bogomips        : 1989.22
...
```





## ◇ Exemple : informations sur le noyau et sa version

```
% cat /proc/version
Linux version 2.4.18-27.7.xsmp (bhcompile@stripples.devel.redhat.com)
(gcc version 2.96 20000731 (Red Hat Linux 7.3 2.96-112)) #1 SMP Fri
Mar 14 05:52:30 EST 2003
```

## ◇ Exemple : informations sur le uptime du noyau

```
% cat /proc/uptime
810311.03 800120.37
```

## ◇ Exemple : informations sur des statistiques du noyau sur l'emploi de la machine

```
% cat /proc/stat
cpu  1276780 2094 1821586 158965456
cpu0 681135 1171 903015 79447636
cpu1 595645 923 918571 79517818
page 19636553 18178406
swap 3965 4959
intr 460335474 414888745 32 0 4 4 3635073 6 0 1 0 35855460 1382047 897324 0 11 367
disk_io: (3,0):(3,3,24,0,0) (22,2):(695266,373120,8363128,322146,6352160) (22,3):(
ctxt 233821067
btime 1066985101
processes 201147
```

## ◇ Exemple : informations sur la table ARP du noyau

```
% cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
134.157.46.254   0x1         0x2         00:02:7E:21:F7:9C     *        eth0
192.168.4.18     0x1         0x2         00:48:54:6B:E5:B0     *        eth3
```

## ◇ Exemple : informations sur les sockets unix du noyau

```
% cat /proc/net/unix
Num       RefCount Protocol Flags    Type St Inode Path
db7f5580: 00000002 00000000 00010000 0001 01  9782 /tmp/.font-unix/fs7100
e9fcc580: 00000002 00000000 00010000 0001 01  9940 /tmp/.X11-unix/X0
d3e23a80: 00000002 00000000 00010000 0001 01  8308 /var/lib/mysql/mysql.sock
ce1dd080: 00000002 00000000 00010000 0001 01  9600 /dev/gpmctl
d75f5080: 00000014 00000000 00000000 0002 01  5050 /dev/log
cb91e080: 00000002 00000000 00000000 0002 01 1088373
d4f46080: 00000002 00000000 00000000 0002 01 1087696
f679a580: 00000002 00000000 00000000 0002 01 1085719
f1204080: 00000002 00000000 00000000 0002 01 1057186
```





## ◇ Exemple : informations sur les utilisations de la mémoire par le noyau

```
% cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  1055293440 1040719872 14573568        0 107249664 692142080
Swap: 4294901760 37588992 4257312768
MemTotal:      1030560 kB
MemFree:         14232 kB
MemShared:           0 kB
Buffers:       104736 kB
Cached:        654592 kB
SwapCached:     21328 kB
Active:        547860 kB
Inact_dirty:   197696 kB
Inact_clean:   119996 kB
Inact_target:  173108 kB
HighTotal:     131008 kB
HighFree:        1024 kB
LowTotal:      899552 kB
LowFree:        13208 kB
SwapTotal:    4194240 kB
SwapFree:     4157532 kB
Committed_AS:  281672 kB
```

(en anglais *temporary*)

Le répertoire `/tmp` sert à stocker des fichiers temporaires.
C'est l'équivalent de `C:\windows\temp` sur Microsoft Windows.

Ses droits d'accès :

```
drwxrwxrwt  12 root   sys    2648 Sep 28 13:02 /tmp/
```

c'est-à-dire 1777 exprimé en octal :

- signification de 777 : tout le monde sur la machine peut créer, modifier, effacer des fichiers
- signification de 1000 : un utilisateur ne peut effacer que les fichiers qui lui appartiennent

En principe, `/tmp` est **vidé au démarrage de la machine UNIX**.





(en anglais *variable log files*)
Il stocke différents fichiers de bord du système.
Par exemple :

```
total 1900
-rw-------   1 root     sys            0 Oct 29  2002 authlog
-rw-r--r--   1 root     other     294141 Nov  2 18:50 ppp.log
-rw-r--r--   1 root     root         844 Aug 28 13:06 skipd.log
-rw-r--r--   1 root     root         844 Aug 28 12:55 skipd.log.0
-rw-r--r--   1 root     root         844 Aug 28 12:52 skipd.log.1
-rw-r--r--   1 root     root         844 Aug 28 12:45 skipd.log.2
-rw-r--r--   1 root     root         844 Aug 28 12:41 skipd.log.3
-rw-r--r--   1 root     other        254 Oct 29  2002 sysidconfig.log
-rw-r--r--   1 root     sys        57297 Nov  2 18:50 syslog
-rw-r--r--   1 root     sys        74519 Oct 31 03:10 syslog.0
-rw-r--r--   1 root     sys        85749 Oct 24 03:10 syslog.1
-rw-r--r--   1 root     sys        79963 Oct 17 03:10 syslog.2
-rw-r--r--   1 root     sys        69391 Oct 10 03:10 syslog.3
-rw-r--r--   1 root     sys        85748 Oct  3 03:10 syslog.4
-rw-r--r--   1 root     sys        69120 Sep 26 03:10 syslog.5
-rw-r--r--   1 root     sys        74101 Sep 19 03:10 syslog.6
-rw-r--r--   1 root     sys        63515 Sep 12 03:10 syslog.7
```

(en anglais *variable mailbox files* )

Il stocke les boites aux lettres de courrier électronique.

Par exemple :

```
% ls -l /var/mail/besancon
-rw-rw---- 1 besancon mail   1311 Dec  4  2002 /var/mail/besancon
```

Parfois, on rencontre encore « /var/spool/mail ».





(en anglais *variable run time data*)

Il contient des informations sur l'état du système après le dernier démarrage.
⇒ nettoyer au moment du démarrage les anciens fichiers

En général, on y trouve des fichiers de la forme « /var/run/démon.pid » qui stocke le PID de l'application « démon ».

Intérêt ?
Plutôt que faire :

```
% ps -ax | grep démon
% kill -TERM PID-de-démon-trouvé-ci-dessus
```

on fait :

```
% kill -TERM `cat /var/run/démon.pid`
```

(en anglais *variable spool data*)

Il stocke des sous répertoires dédiés à des applications consommatrices de nombreux fichiers temporaires comme par exemple le système d'impression ou la messagerie électronique.

Pour le système d'impression, c'est « /var/spool/lpd ».
Pour la messagerie électronique, c'est « /var/spool/mqueue ».





(en anglais *variable temporary*)

Le répertoire « /var/tmp » sert à stocker des fichiers temporaires.

Différence avec « /tmp » : « /var/tmp » n'est pas vidé au démarrage de la machine UNIX.

Ses droits d'accès :

```
drwxrwxrwt  12 root   sys   2648 Sep 28 13:02 /var/tmp
```

c'est-à-dire 1777 exprimé en octal :

- signification de 777 : tout le monde sur la machine peut créer, modifier, effacer des fichiers
- signification de 1000 : un utilisateur ne peut effacer que les fichiers qui lui appartiennent

(en anglais *mount*)

**« /mnt » = mount point**
Point de montage pour rendre disponible provisoirement une partition (locale ou réseau)

**/mnt est un répertoire.**

Tendance LINUX :

- /mnt/floppy
- /mnt/cdrom
- /mnt/removable
- etc.

Nouvelle tendance LINUX :

- /media/cdrom
- /media/cdrecorder
- /media/removable
- etc.





(en anglais *optional*)

**« /usr/local » = local add-ons**
**« /opt » = local add-on applications software**
/usr/local contient les ajouts locaux de commandes.
/opt contient les ajouts locaux de « paquets » complets de commandes (càd nombreuses commandes exécutables et fichiers annexes)

/opt existe par exemple sur SOLARIS
Le logiciel *application* s'installera dans /opt/application qui contiendra :

- /opt/application/bin
- /opt/application/include
- /opt/application/lib
- /opt/application/man

Les fichiers de config vont en principe dans « /etc/opt/ ».

(en anglais *source*)

Quand il existe, ce directory contient en principe les sources du système UNIX sur lequel on le trouve.
Utile au développeur système.

Présent sur les unix libres.
Non présent sur les UNIX commerciaux du type SOLARIS, AIX, HP-UX, etc.





(en anglais *system files*)

Plus généralement, on trouve `/usr/sys`. Ce directory contient les fichiers modules « .o » nécessaires à recompiler uniquement un noyau UNIX.

Présent sur les UNIX commerciaux type SOLARIS, AIX, HP-UX, etc.
Non présent sur les unix libres.

Ci joint dans la version imprimée de ce cours, le document FHS version 2.3.

Adresse web du document fourni :
« `http://www.pathname.com/fhs/pub/fhs-2.3.pdf` »

# Filesystem Hierarchy Standard

**Filesystem Hierarchy Standard Group**

**Edited by**

**Rusty Russell**

**Daniel Quinlan**

**Christopher Yeoh**

**Filesystem Hierarchy Standard**
by Filesystem Hierarchy Standard Group
Edited by Rusty Russell, Daniel Quinlan, and Christopher Yeoh

Published January 28 2004
Copyright © 1994-2004 Daniel Quinlan
Copyright © 2001-2004 Paul 'Rusty' Russell
Copyright © 2003-2004 Christopher Yeoh

This standard consists of a set of requirements and guidelines for file and directory placement under UNIX-like operating systems. The guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.

All trademarks and copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.
Permission is granted to make and distribute verbatim copies of this standard provided the copyright and this permission notice are preserved on all copies.
Permission is granted to copy and distribute modified versions of this standard under the conditions for verbatim copying, provided also that the title page is labeled as modified including a reference to the original standard, provided that information on retrieving the original standard is included, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.
Permission is granted to copy and distribute translations of this standard into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the copyright holder.

# Table of Contents

**1. Introduction** ..... **1**
1.1. Purpose ..... 1
1.2. Conventions ..... 1
**2. The Filesystem** ..... **2**
**3. The Root Filesystem** ..... **3**
3.1. Purpose ..... 3
3.2. Requirements ..... 4
3.3. Specific Options ..... 4
3.4. /bin : Essential user command binaries (for use by all users) ..... 4
3.4.1. Purpose ..... 4
3.4.2. Requirements ..... 5
3.4.3. Specific Options ..... 6
3.5. /boot : Static files of the boot loader ..... 6
3.5.1. Purpose ..... 6
3.5.2. Specific Options ..... 7
3.6. /dev : Device files ..... 7
3.6.1. Purpose ..... 7
3.6.2. Specific Options ..... 7
3.7. /etc : Host-specific system configuration ..... 7
3.7.1. Purpose ..... 7
3.7.2. Requirements ..... 7
3.7.3. Specific Options ..... 7
3.7.4. /etc/opt : Configuration files for /opt ..... 8
3.7.4.1. Purpose ..... 9
3.7.4.2. Requirements ..... 9
3.7.5. /etc/X11 : Configuration for the X Window System (optional) ..... 9
3.7.5.1. Purpose ..... 9
3.7.5.2. Specific Options ..... 9
3.7.6. /etc/sgml : Configuration files for SGML (optional) ..... 9
3.7.6.1. Purpose ..... 10
3.7.7. /etc/xml : Configuration files for XML (optional) ..... 10
3.7.7.1. Purpose ..... 10
3.8. /home : User home directories (optional) ..... 10
3.8.1. Purpose ..... 10
3.8.2. Requirements ..... 10
3.9. /lib : Essential shared libraries and kernel modules ..... 10
3.9.1. Purpose ..... 10
3.9.2. Requirements ..... 10
3.9.3. Specific Options ..... 11
3.10. /lib<qual> : Alternate format essential shared libraries (optional) ..... 11
3.10.1. Purpose ..... 11
3.10.2. Requirements ..... 11
3.11. /media : Mount point for removeable media ..... 11
3.11.1. Purpose ..... 11
3.11.2. Specific Options ..... 12
3.12. /mnt : Mount point for a temporarily mounted filesystem ..... 12
3.12.1. Purpose ..... 12
3.13. /opt : Add-on application software packages ..... 12
3.13.1. Purpose ..... 12

3.13.2. Requirements ..... 12
3.14. /root : Home directory for the root user (optional) ..... 13
3.14.1. Purpose ..... 14
3.15. /sbin : System binaries ..... 14
3.15.1. Purpose ..... 14
3.15.2. Requirements ..... 14
3.15.3. Specific Options ..... 14
3.16. /srv : Data for services provided by this system ..... 15
3.16.1. Purpose ..... 15
3.17. /tmp : Temporary files ..... 15
3.17.1. Purpose ..... 15
**4. The /usr Hierarchy ..... 18**
4.1. Purpose ..... 18
4.2. Requirements ..... 18
4.3. Specific Options ..... 18
4.4. /usr/X11R6 : X Window System, Version 11 Release 6 (optional) ..... 18
4.4.1. Purpose ..... 19
4.4.2. Specific Options ..... 19
4.5. /usr/bin : Most user commands ..... 19
4.5.1. Purpose ..... 19
4.5.2. Specific Options ..... 19
4.6. /usr/include : Directory for standard include files. ..... 20
4.6.1. Purpose ..... 20
4.6.2. Specific Options ..... 20
4.7. /usr/lib : Libraries for programming and packages ..... 20
4.7.1. Purpose ..... 20
4.7.2. Specific Options ..... 20
4.8. /usr/lib<qual> : Alternate format libraries (optional) ..... 21
4.8.1. Purpose ..... 21
4.8.2. /usr/local : Local hierarchy ..... 21
4.8.2.1. Purpose ..... 21
4.8.2.2. Requirements ..... 21
4.8.2.3. Specific Options ..... 21
4.9. /usr/local/share ..... 22
4.10. /usr/sbin : Non-essential standard system binaries ..... 22
4.10.1. Purpose ..... 22
4.11. /usr/share : Architecture-independent data ..... 22
4.11.1. Purpose ..... 22
4.11.2. Requirements ..... 22
4.11.3. Specific Options ..... 23
4.11.4. /usr/share/dict : Word lists (optional) ..... 23
4.11.4.1. Purpose ..... 23
4.11.4.2. Specific Options ..... 23
4.11.5. /usr/share/man : Manual pages ..... 24
4.11.5.1. Purpose ..... 24
4.11.5.2. Specific Options ..... 25
4.11.6. /usr/share/misc : Miscellaneous architecture-independent data ..... 26
4.11.6.1. Specific Options ..... 26
4.11.7. /usr/share/sgml : SGML data (optional) ..... 27
4.11.7.1. Purpose ..... 27
4.11.7.2. Specific Options ..... 27

4.11.8. /usr/share/xml : XML data (optional) ..... 27
4.11.8.1. Purpose ..... 27
4.11.8.2. Specific Options ..... 27
4.12. /usr/src : Source code (optional) ..... 28
4.12.1. Purpose ..... 28
**5. The /var Hierarchy ..... 30**
5.1. Purpose ..... 30
5.2. Requirements ..... 30
5.3. Specific Options ..... 30
5.4. /var/account : Process accounting logs (optional) ..... 31
5.4.1. Purpose ..... 31
5.5. /var/cache : Application cache data ..... 31
5.5.1. Purpose ..... 31
5.5.2. Specific Options ..... 31
5.5.3. /var/cache/fonts : Locally-generated fonts (optional) ..... 32
5.5.3.1. Purpose ..... 32
5.5.3.2. Specific Options ..... 32
5.5.4. /var/cache/man : Locally-formatted manual pages (optional) ..... 32
5.5.4.1. Purpose ..... 32
5.6. /var/crash : System crash dumps (optional) ..... 33
5.6.1. Purpose ..... 33
5.7. /var/games : Variable game data (optional) ..... 33
5.7.1. Purpose ..... 33
5.8. /var/lib : Variable state information ..... 33
5.8.1. Purpose ..... 33
5.8.2. Requirements ..... 34
5.8.3. Specific Options ..... 34
5.8.4. /var/lib/<editor> : Editor backup files and state (optional) ..... 34
5.8.4.1. Purpose ..... 34
5.8.5. /var/lib/hwclock : State directory for hwclock (optional) ..... 35
5.8.5.1. Purpose ..... 35
5.8.6. /var/lib/misc : Miscellaneous variable data ..... 35
5.8.6.1. Purpose ..... 35
5.9. /var/lock : Lock files ..... 35
5.9.1. Purpose ..... 35
5.10. /var/log : Log files and directories ..... 35
5.10.1. Purpose ..... 36
5.10.2. Specific Options ..... 36
5.11. /var/mail : User mailbox files (optional) ..... 36
5.11.1. Purpose ..... 36
5.12. /var/opt : Variable data for /opt ..... 36
5.12.1. Purpose ..... 36
5.13. /var/run : Run-time variable data ..... 37
5.13.1. Purpose ..... 37
5.13.2. Requirements ..... 37
5.14. /var/spool : Application spool data ..... 37
5.14.1. Purpose ..... 37
5.14.2. Specific Options ..... 37
5.14.3. /var/spool/lpd : Line-printer daemon print queues (optional) ..... 38
5.14.3.1. Purpose ..... 38
5.14.3.2. Specific Options ..... 38

5.14.4. /var/spool/rwho : Rwhod files (optional) ..... 38
5.14.4.1. Purpose ..... 38
5.15. /var/tmp : Temporary files preserved between system reboots ..... 38
5.15.1. Purpose ..... 38
5.16. /var/yp : Network Information Service (NIS) database files (optional) ..... 39
5.16.1. Purpose ..... 39
**6. Operating System Specific Annex ..... 40**
6.1. Linux ..... 40
6.1.1. / : Root directory ..... 40
6.1.2. /bin : Essential user command binaries (for use by all users) ..... 40
6.1.3. /dev : Devices and special files ..... 40
6.1.4. /etc : Host-specific system configuration ..... 40
6.1.5. /lib64 and /lib32 : 64/32-bit libraries (architecture dependent) ..... 41
6.1.6. /proc : Kernel and process information virtual filesystem ..... 41
6.1.7. /sbin : Essential system binaries ..... 41
6.1.8. /usr/include : Header files included by C programs ..... 42
6.1.9. /usr/src : Source code ..... 42
6.1.10. /var/spool/cron : cron and at jobs ..... 43
**7. Appendix ..... 44**
7.1. The FHS mailing list ..... 44
7.2. Background of the FHS ..... 44
7.3. General Guidelines ..... 44
7.4. Scope ..... 44
7.5. Acknowledgments ..... 45
7.6. Contributors ..... 45

# Chapter 1. Introduction

## 1.1. Purpose

This standard enables:

- Software to predict the location of installed files and directories, and
- Users to predict the location of installed files and directories.

We do this by:

- Specifying guiding principles for each area of the filesystem,
- Specifying the minimum files and directories required,
- Enumerating exceptions to the principles, and
- Enumerating specific cases where there has been historical conflict.

The FHS document is used by:

- Independent software suppliers to create applications which are FHS compliant, and work with distributions which are FHS complaint,
- OS creators to provide systems which are FHS compliant, and
- Users to understand and maintain the FHS compliance of a system.

The FHS document has a limited scope:

- Local placement of local files is a local issue, so FHS does not attempt to usurp system administrators.
- FHS addresses issues where file placements need to be coordinated between multiple parties such as local sites, distributions, applications, documentation, etc.

## 1.2. Conventions

We recommend that you read a typeset version of this document rather than the plain text version. In the typeset version, the names of files and directories are displayed in a constant-width font.

Components of filenames that vary are represented by a description of the contents enclosed in "<" and ">" characters, <*thus*>. Electronic mail addresses are also enclosed in "<" and ">" but are shown in the usual typeface.

Optional components of filenames are enclosed in "*[*" and "*]*" characters and may be combined with the "<" and ">" convention. For example, if a filename is allowed to occur either with or without an extension, it might be represented by <*filename*>*[*.<*extension*>*]*.

Variable substrings of directory names and filenames are indicated by "*".

The sections of the text marked as *Rationale* are explanatory and are non-normative.

# Chapter 2. The Filesystem

This standard assumes that the operating system underlying an FHS-compliant file system supports the same basic security features found in most UNIX filesystems.

It is possible to define two independent distinctions among files: shareable vs. unshareable and variable vs. static. In general, files that differ in either of these respects should be located in different directories. This makes it easy to store files with different usage characteristics on different filesystems.

"Shareable" files are those that can be stored on one host and used on others. "Unshareable" files are those that are not shareable. For example, the files in user home directories are shareable whereas device lock files are not.

"Static" files include binaries, libraries, documentation files and other files that do not change without system administrator intervention. "Variable" files are files that are not static.

 **Rationale**

Shareable files can be stored on one host and used on several others. Typically, however, not all files in the filesystem hierarchy are shareable and so each system has local storage containing at least its unshareable files. It is convenient if all the files a system requires that are stored on a foreign host can be made available by mounting one or a few directories from the foreign host.

Static and variable files should be segregated because static files, unlike variable files, can be stored on read-only media and do not need to be backed up on the same schedule as variable files.

Historical UNIX-like filesystem hierarchies contained both static and variable files under both `/usr` and `/etc`. In order to realize the advantages mentioned above, the `/var` hierarchy was created and all variable files were transferred from `/usr` to `/var`. Consequently `/usr` can now be mounted read-only (if it is a separate filesystem). Variable files have been transferred from `/etc` to `/var` over a longer period as technology has permitted.

Here is an example of a FHS-compliant system. (Other FHS-compliant layouts are possible.)

| | **shareable** | **unshareable** |
|---|---|---|
| static | /usr | /etc |
| | /opt | /boot |
| variable | /var/mail | /var/run |
| | /var/spool/news | /var/lock |

# Chapter 3. The Root Filesystem

## 3.1. Purpose

The contents of the root filesystem must be adequate to boot, restore, recover, and/or repair the system.

- To boot a system, enough must be present on the root partition to mount other filesystems. This includes utilities, configuration, boot loader information, and other essential start-up data. `/usr`, `/opt`, and `/var` are designed such that they may be located on other partitions or filesystems.
- To enable recovery and/or repair of a system, those utilities needed by an experienced maintainer to diagnose and reconstruct a damaged system must be present on the root filesystem.
- To restore a system, those utilities needed to restore from system backups (on floppy, tape, etc.) must be present on the root filesystem.



**Rationale**

The primary concern used to balance these considerations, which favor placing many things on the root filesystem, is the goal of keeping root as small as reasonably possible. For several reasons, it is desirable to keep the root filesystem small:

- It is occasionally mounted from very small media.
- The root filesystem contains many system-specific configuration files. Possible examples include a kernel that is specific to the system, a specific hostname, etc. This means that the root filesystem isn't always shareable between networked systems. Keeping it small on servers in networked systems minimizes the amount of lost space for areas of unshareable files. It also allows workstations with smaller local hard drives.
- While you may have the root filesystem on a large partition, and may be able to fill it to your heart's content, there will be people with smaller partitions. If you have more files installed, you may find incompatibilities with other systems using root filesystems on smaller partitions. If you are a developer then you may be turning your assumption into a problem for a large number of users.
- Disk errors that corrupt data on the root filesystem are a greater problem than errors on any other partition. A small root filesystem is less prone to corruption as the result of a system crash.

Applications must never create or require special files or subdirectories in the root directory. Other locations in the FHS hierarchy provide more than enough flexibility for any package.



**Rationale**

There are several reasons why creating a new subdirectory of the root filesystem is prohibited:

- It demands space on a root partition which the system administrator may want kept small and simple for either performance or security reasons.
- It evades whatever discipline the system administrator may have set up for distributing standard file hierarchies across mountable volumes.

Distributions should not create new directories in the root hierarchy without extremely careful consideration of the consequences including for application portability.

## 3.2. Requirements

The following directories, or symbolic links to directories, are required in `/`.

| Directory | Description |
|---|---|
| `bin` | Essential command binaries |
| `boot` | Static files of the boot loader |
| `dev` | Device files |
| `etc` | Host-specific system configuration |
| `lib` | Essential shared libraries and kernel modules |
| `media` | Mount point for removeable media |
| `mnt` | Mount point for mounting a filesystem temporarily |
| `opt` | Add-on application software packages |
| `sbin` | Essential system binaries |
| `srv` | Data for services provided by this system |
| `tmp` | Temporary files |
| `usr` | Secondary hierarchy |
| `var` | Variable data |

Each directory listed above is specified in detail in separate subsections below. `/usr` and `/var` each have a complete section in this document due to the complexity of those directories.

## 3.3. Specific Options

The following directories, or symbolic links to directories, must be in `/`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| `home` | User home directories (optional) |
| `lib<qual>` | Alternate format essential shared libraries (optional) |
| `root` | Home directory for the root user (optional) |

Each directory listed above is specified in detail in separate subsections below.

## 3.4. /bin : Essential user command binaries (for use by all users)

### 3.4.1. Purpose

`/bin` contains commands that may be used by both the system administrator and by users, but which are required when no other filesystems are mounted (e.g. in single user mode). It may also contain commands which

are used indirectly by scripts. [1]

## 3.4.2. Requirements

There must be no subdirectories in `/bin`.

The following commands, or symbolic links to commands, are required in `/bin`.

| Command | Description |
|---|---|
| **cat** | Utility to concatenate files to standard output |
| **chgrp** | Utility to change file group ownership |
| **chmod** | Utility to change file access permissions |
| **chown** | Utility to change file owner and group |
| **cp** | Utility to copy files and directories |
| **date** | Utility to print or set the system data and time |
| **dd** | Utility to convert and copy a file |
| **df** | Utility to report filesystem disk space usage |
| **dmesg** | Utility to print or control the kernel message buffer |
| **echo** | Utility to display a line of text |
| **false** | Utility to do nothing, unsuccessfully |
| **hostname** | Utility to show or set the system's host name |
| **kill** | Utility to send signals to processes |
| **ln** | Utility to make links between files |
| **login** | Utility to begin a session on the system |
| **ls** | Utility to list directory contents |
| **mkdir** | Utility to make directories |
| **mknod** | Utility to make block or character special files |
| **more** | Utility to page through text |
| **mount** | Utility to mount a filesystem |
| **mv** | Utility to move/rename files |
| **ps** | Utility to report process status |
| **pwd** | Utility to print name of current working directory |
| **rm** | Utility to remove files or directories |
| **rmdir** | Utility to remove empty directories |
| **sed** | The 'sed' stream editor |
| **sh** | The Bourne command shell |
| **stty** | Utility to change and print terminal line settings |
| **su** | Utility to change user ID |
| **sync** | Utility to flush filesystem buffers |
| **true** | Utility to do nothing, successfully |
| **umount** | Utility to unmount file systems |
| **uname** | Utility to print system information |

If **/bin/sh** is not a true Bourne shell, it must be a hard or symbolic link to the real shell command.

The **[** and **test** commands must be placed together in either `/bin` or `/usr/bin`.


**Rationale**

For example bash behaves differently when called as **sh** or **bash**. The use of a symbolic link also allows users to easily see that **/bin/sh** is not a true Bourne shell.

The requirement for the **[** and **test** commands to be included as binaries (even if implemented internally by the shell) is shared with the POSIX.2 standard.

### 3.4.3. Specific Options

The following programs, or symbolic links to programs, must be in `/bin` if the corresponding subsystem is installed:

| Command | Description |
|---|---|
| **csh** | The C shell (optional) |
| **ed** | The 'ed' editor (optional) |
| **tar** | The tar archiving utility (optional) |
| **cpio** | The cpio archiving utility (optional) |
| **gzip** | The GNU compression utility (optional) |
| **gunzip** | The GNU uncompression utility (optional) |
| **zcat** | The GNU uncompression utility (optional) |
| **netstat** | The network statistics utility (optional) |
| **ping** | The ICMP network test utility (optional) |

If the **gunzip** and **zcat** programs exist, they must be symbolic or hard links to gzip. **/bin/csh** may be a symbolic link to **/bin/tcsh** or **/usr/bin/tcsh**.


**Rationale**

The tar, gzip and cpio commands have been added to make restoration of a system possible (provided that `/` is intact).

Conversely, if no restoration from the root partition is ever expected, then these binaries might be omitted (e.g., a ROM chip root, mounting `/usr` through NFS). If restoration of a system is planned through the network, then **ftp** or **tftp** (along with everything necessary to get an ftp connection) must be available on the root partition.

## 3.5. /boot : Static files of the boot loader

### 3.5.1. Purpose

This directory contains everything required for the boot process except configuration files not needed at boot time and the map installer. Thus /boot stores data that is used before the kernel begins executing user-mode programs. This may include saved master boot sectors and sector map files. [2]

### 3.5.2. Specific Options

The operating system kernel must be located in either `/` or `/boot`. [3]

## 3.6. /dev : Device files

### 3.6.1. Purpose

The `/dev` directory is the location of special or device files.

### 3.6.2. Specific Options

If it is possible that devices in `/dev` will need to be manually created, `/dev` must contain a command named `MAKEDEV`, which can create devices as needed. It may also contain a `MAKEDEV.local` for any local devices.

If required, `MAKEDEV` must have provisions for creating any device that may be found on the system, not just those that a particular implementation installs.

## 3.7. /etc : Host-specific system configuration

### 3.7.1. Purpose

The `/etc` hierarchy contains configuration files. A "configuration file" is a local file used to control the operation of a program; it must be static and cannot be an executable binary. [4]

### 3.7.2. Requirements

No binaries may be located under `/etc`. [5]

The following directories, or symbolic links to directories are required in `/etc`:

| Directory | Description |
|---|---|
| opt | Configuration for /opt |
| X11 | Configuration for the X Window system (optional) |
| sgml | Configuration for SGML (optional) |
| xml | Configuration for XML (optional) |

### 3.7.3. Specific Options

The following directories, or symbolic links to directories must be in `/etc`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| opt | Configuration for /opt |

The following files, or symbolic links to files, must be in `/etc` if the corresponding subsystem is installed: [6]

| File | Description |
|---|---|
| `csh.login` | Systemwide initialization file for C shell logins (optional) |
| `exports` | NFS filesystem access control list (optional) |
| `fstab` | Static information about filesystems (optional) |
| `ftpusers` | FTP daemon user access control list (optional) |
| `gateways` | File which lists gateways for routed (optional) |
| `gettydefs` | Speed and terminal settings used by getty (optional) |
| `group` | User group file (optional) |
| `host.conf` | Resolver configuration file (optional) |
| `hosts` | Static information about host names (optional) |
| `hosts.allow` | Host access file for TCP wrappers (optional) |
| `hosts.deny` | Host access file for TCP wrappers (optional) |
| `hosts.equiv` | List of trusted hosts for rlogin, rsh, rcp (optional) |
| `hosts.lpd` | List of trusted hosts for lpd (optional) |
| `inetd.conf` | Configuration file for inetd (optional) |
| `inittab` | Configuration file for init (optional) |
| `issue` | Pre-login message and identification file (optional) |
| `ld.so.conf` | List of extra directories to search for shared libraries (optional) |
| `motd` | Post-login message of the day file (optional) |
| `mtab` | Dynamic information about filesystems (optional) |
| `mtools.conf` | Configuration file for mtools (optional) |
| `networks` | Static information about network names (optional) |
| `passwd` | The password file (optional) |
| `printcap` | The lpd printer capability database (optional) |
| `profile` | Systemwide initialization file for sh shell logins (optional) |
| `protocols` | IP protocol listing (optional) |
| `resolv.conf` | Resolver configuration file (optional) |
| `rpc` | RPC protocol listing (optional) |
| `securetty` | TTY access control for root login (optional) |
| `services` | Port names for network services (optional) |
| `shells` | Pathnames of valid login shells (optional) |
| `syslog.conf` | Configuration file for syslogd (optional) |

`mtab` does not fit the static nature of `/etc`: it is excepted for historical reasons. [7]

## 3.7.4. /etc/opt : Configuration files for /opt

### 3.7.4.1. Purpose

Host-specific configuration files for add-on application software packages must be installed within the directory `/etc/opt/<subdir>`, where `<subdir>` is the name of the subtree in `/opt` where the static data from that package is stored.

### 3.7.4.2. Requirements

No structure is imposed on the internal arrangement of `/etc/opt/<subdir>`.

If a configuration file must reside in a different location in order for the package or system to function properly, it may be placed in a location other than `/etc/opt/<subdir>`.

 **Rationale**

Refer to the rationale for `/opt`.

## 3.7.5. /etc/X11 : Configuration for the X Window System (optional)

### 3.7.5.1. Purpose

*/etc/X11* is the location for all X11 host-specific configuration. This directory is necessary to allow local control if */usr* is mounted read only.

### 3.7.5.2. Specific Options

The following files, or symbolic links to files, must be in `/etc/X11` if the corresponding subsystem is installed:

| File | Description |
|---|---|
| `Xconfig` | The configuration file for early versions of XFree86 (optional) |
| `XF86Config` | The configuration file for XFree86 versions 3 and 4 (optional) |
| `Xmodmap` | Global X11 keyboard modification file (optional) |

Subdirectories of `/etc/X11` may include those for `xdm` and for any other programs (some window managers, for example) that need them. [8] We recommend that window managers with only one configuration file which is a default `.*wmrc` file must name it `system.*wmrc` (unless there is a widely-accepted alternative name) and not use a subdirectory. Any window manager subdirectories must be identically named to the actual window manager binary.

### 3.7.6. /etc/sgml : Configuration files for SGML (optional)

#### 3.7.6.1. Purpose

Generic configuration files defining high-level parameters of the SGML systems are installed here. Files with names `*.conf` indicate generic configuration files. File with names `*.cat` are the DTD-specific centralized catalogs, containing references to all other catalogs needed to use the given DTD. The super catalog file `catalog` references all the centralized catalogs.

### 3.7.7. /etc/xml : Configuration files for XML (optional)

#### 3.7.7.1. Purpose

Generic configuration files defining high-level parameters of the XML systems are installed here. Files with names `*.conf` indicate generic configuration files. The super catalog file `catalog` references all the centralized catalogs.

## 3.8. /home : User home directories (optional)

### 3.8.1. Purpose

`/home` is a fairly standard concept, but it is clearly a site-specific filesystem. [9] The setup will differ from host to host. Therefore, no program should rely on this location. [10]

### 3.8.2. Requirements

User specific configuration files for applications are stored in the user’s home directory in a file that starts with the ’.’ character (a "dot file"). If an application needs to create more than one dot file then they should be placed in a subdirectory with a name starting with a ’.’ character, (a "dot directory"). In this case the configuration files should not start with the ’.’ character. [11]

## 3.9. /lib : Essential shared libraries and kernel modules

### 3.9.1. Purpose

The `/lib` directory contains those shared library images needed to boot the system and run the commands in the root filesystem, ie. by binaries in `/bin` and `/sbin`. [12]

### 3.9.2. Requirements

At least one of each of the following filename patterns are required (they may be files, or symbolic links):

| File | Description |
|---|---|
| `libc.so.*` | The dynamically-linked C library (optional) |
| `ld*` | The execution time linker/loader (optional) |

If a C preprocessor is installed, */lib/cpp* must be a reference to it, for historical reasons. [13]

### 3.9.3. Specific Options

The following directories, or symbolic links to directories, must be in `/lib`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| `modules` | Loadable kernel modules (optional) |

## 3.10. /lib<qual> : Alternate format essential shared libraries (optional)

### 3.10.1. Purpose

There may be one or more variants of the `/lib` directory on systems which support more than one binary format requiring separate libraries. [14]

### 3.10.2. Requirements

If one or more of these directories exist, the requirements for their contents are the same as the normal `/lib` directory, except that `/lib<qual>/cpp` is not required. [15]

## 3.11. /media : Mount point for removeable media

### 3.11.1. Purpose

This directory contains subdirectories which are used as mount points for removeable media such as floppy disks, cdroms and zip disks.


**Rationale**

Historically there have been a number of other different places used to mount removeable media such as `/cdrom`, `/mnt` or `/mnt/cdrom`. Placing the mount points for all removeable media directly in the root directory would potentially result in a large number of extra directories in `/`. Although the use of subdirectories in `/mnt` as a mount point has recently been common, it conflicts with a much older tradition of using `/mnt` directly as a temporary mount point.

### 3.11.2. Specific Options

The following directories, or symbolic links to directories, must be in `/media`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| `floppy` | Floppy drive (optional) |
| `cdrom` | CD-ROM drive (optional) |
| `cdrecorder` | CD writer (optional) |
| `zip` | Zip drive (optional) |

On systems where more than one device exists for mounting a certain type of media, mount directories can be created by appending a digit to the name of those available above starting with ’0’, but the unqualified name must also exist. [16]

## 3.12. /mnt : Mount point for a temporarily mounted filesystem

### 3.12.1. Purpose

This directory is provided so that the system administrator may temporarily mount a filesystem as needed. The content of this directory is a local issue and should not affect the manner in which any program is run.

This directory must not be used by installation programs: a suitable temporary directory not in use by the system must be used instead.

## 3.13. /opt : Add-on application software packages

### 3.13.1. Purpose

`/opt` is reserved for the installation of add-on application software packages.

A package to be installed in `/opt` must locate its static files in a separate `/opt/<package>` or `/opt/<provider>` directory tree, where `<package>` is a name that describes the software package and `<provider>` is the provider’s LANANA registered name.

## 3.13.2. Requirements

| Directory | Description |
|---|---|
| <package> | Static package objects |
| <provider> | LANANA registered provider name |

The directories `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib`, and `/opt/man` are reserved for local system administrator use. Packages may provide "front-end" files intended to be placed in (by linking or copying) these reserved directories by the local system administrator, but must function normally in the absence of these reserved directories.

Programs to be invoked by users must be located in the directory `/opt/<package>/bin` or under the `/opt/<provider>` hierarchy. If the package includes UNIX manual pages, they must be located in `/opt/<package>/share/man` or under the `/opt/<provider>` hierarchy, and the same substructure as `/usr/share/man` must be used.

Package files that are variable (change in normal operation) must be installed in `/var/opt`. See the section on `/var/opt` for more information.

Host-specific configuration files must be installed in `/etc/opt`. See the section on `/etc` for more information.

No other package files may exist outside the `/opt`, `/var/opt`, and `/etc/opt` hierarchies except for those package files that must reside in specific locations within the filesystem tree in order to function properly. For example, device lock files must be placed in `/var/lock` and devices must be located in `/dev`.

Distributions may install software in `/opt`, but must not modify or delete software installed by the local system administrator without the assent of the local system administrator.



**Rationale**

The use of `/opt` for add-on software is a well-established practice in the UNIX community. The System V Application Binary Interface [AT&T 1990], based on the System V Interface Definition (Third Edition), provides for an `/opt` structure very similar to the one defined here.

The Intel Binary Compatibility Standard v. 2 (iBCS2) also provides a similar structure for `/opt`.

Generally, all data required to support a package on a system must be present within `/opt/<package>`, including files intended to be copied into `/etc/opt/<package>` and `/var/opt/<package>` as well as reserved directories in `/opt`.

The minor restrictions on distributions using `/opt` are necessary because conflicts are possible between distribution-installed and locally-installed software, especially in the case of fixed pathnames found in some binary software.

The structure of the directories below `/opt/<provider>` is left up to the packager of the software, though it is recommended that packages are installed in `/opt/<provider>/<package>` and follow a similar structure to the guidelines for `/opt/package`. A valid reason for diverging from this structure is for support packages which may have files installed in `/opt/<provider>/lib` or `/opt/<provider>/bin`.

# 3.14. /root : Home directory for the root user (optional)

## 3.14.1. Purpose

The root account's home directory may be determined by developer or local preference, but this is the recommended default location. [17]

# 3.15. /sbin : System binaries

## 3.15.1. Purpose

Utilities used for system administration (and other root-only commands) are stored in `/sbin`, `/usr/sbin`, and `/usr/local/sbin`. `/sbin` contains binaries essential for booting, restoring, recovering, and/or repairing the system in addition to the binaries in `/bin`. [18] Programs executed after `/usr` is known to be mounted (when there are no problems) are generally placed into `/usr/sbin`. Locally-installed system administration programs should be placed into `/usr/local/sbin`. [19]

## 3.15.2. Requirements

The following commands, or symbolic links to commands, are required in `/sbin`.

| Command | Description |
|---|---|
| **shutdown** | Command to bring the system down. |

## 3.15.3. Specific Options

The following files, or symbolic links to files, must be in `/sbin` if the corresponding subsystem is installed:

| Command | Description |
|---|---|
| `fastboot` | Reboot the system without checking the disks (optional) |
| `fasthalt` | Stop the system without checking the disks (optional) |
| `fdisk` | Partition table manipulator (optional) |
| `fsck` | File system check and repair utility (optional) |
| `fsck.*` | File system check and repair utility for a specific filesystem (optional) |
| `getty` | The getty program (optional) |
| `halt` | Command to stop the system (optional) |
| `ifconfig` | Configure a network interface (optional) |
| `init` | Initial process (optional) |
| `mkfs` | Command to build a filesystem (optional) |
| `mkfs.*` | Command to build a specific filesystem (optional) |
| `mkswap` | Command to set up a swap area (optional) |

| Command | Description |
|---|---|
| `reboot` | Command to reboot the system (optional) |
| `route` | IP routing table utility (optional) |
| `swapon` | Enable paging and swapping (optional) |
| `swapoff` | Disable paging and swapping (optional) |
| `update` | Daemon to periodically flush filesystem buffers (optional) |

# 3.16. /srv : Data for services provided by this system

## 3.16.1. Purpose

`/srv` contains site-specific data which is served by this system.



**Rationale**

This main purpose of specifying this is so that users may find the location of the data files for particular service, and so that services which require a single tree for readonly data, writable data and scripts (such as cgi scripts) can be reasonably placed. Data that is only of interest to a specific user should go in that users' home directory.

The methodology used to name subdirectories of `/srv` is unspecified as there is currently no consensus on how this should be done. One method for structuring data under `/srv` is by protocol, eg. `ftp`, `rsync`, `www`, and `cvs`. On large systems it can be useful to structure `/srv` by administrative context, such as `/srv/physics/www`, `/srv/compsci/cvs`, etc. This setup will differ from host to host. Therefore, no program should rely on a specific subdirectory structure of `/srv` existing or data necessarily being stored in `/srv`. However `/srv` should always exist on FHS compliant systems and should be used as the default location for such data.

Distributions must take care not to remove locally placed files in these directories without administrator permission. [20]

# 3.17. /tmp : Temporary files

## 3.17.1. Purpose

The `/tmp` directory must be made available for programs that require temporary files.

Programs must not assume that any files or directories in `/tmp` are preserved between invocations of the program.



**Rationale**

IEEE standard P1003.2 (POSIX, part 2) makes requirements that are similar to the above section.

Although data stored in `/tmp` may be deleted in a site-specific manner, it is recommended that files and directories located in `/tmp` be deleted whenever the system is booted.

FHS added this recommendation on the basis of historical precedent and common practice, but did not make it a requirement because system administration is not within the scope of this standard.

# Notes

1. Command binaries that are not essential enough to place into `/bin` must be placed in `/usr/bin`, instead. Items that are required only by non-root users (the X Window System, `chsh`, etc.) are generally not essential enough to be placed into the root partition.
2. Programs necessary to arrange for the boot loader to be able to boot a file must be placed in `/sbin`. Configuration files for boot loaders must be placed in `/etc`.

   The GRUB bootloader reads its configurations file before booting, so that must be placed in `/boot`. However, it is a configuration file, so should be in `/etc`. The answer here is a symbolic link such as `/etc/grub/menu.lst -> /boot/menu.lst`.
3. On some i386 machines, it may be necessary for `/boot` to be located on a separate partition located completely below cylinder 1024 of the boot device due to hardware constraints.

   Certain MIPS systems require a `/boot` partition that is a mounted MS-DOS filesystem or whatever other filesystem type is accessible for the firmware. This may result in restrictions with respect to usable filenames within `/boot` (only for affected systems).
4. The setup of command scripts invoked at boot time may resemble System V, BSD or other models. Further specification in this area may be added to a future version of this standard.
5. It is recommended that files be stored in subdirectories of `/etc` rather than directly in `/etc`.
6. Systems that use the shadow password suite will have additional configuration files in `/etc` (`/etc/shadow` and others) and programs in `/usr/sbin` (**useradd**, **usermod**, and others).
7. On some Linux systems, this may be a symbolic link to `/proc/mounts`, in which case this exception is not required.
8. `/etc/X11/xdm` holds the configuration files for `xdm`. These are most of the files previously found in `/usr/lib/X11/xdm`. Some local variable data for `xdm` is stored in `/var/lib/xdm`.
9. Different people prefer to place user accounts in a variety of places. This section describes only a suggested placement for user home directories; nevertheless we recommend that all FHS-compliant distributions use this as the default location for home directories.

   On small systems, each user's directory is typically one of the many subdirectories of `/home` such as `/home/smith`, `/home/torvalds`, `/home/operator`, etc. On large systems (especially when the `/home` directories are shared amongst many hosts using NFS) it is useful to subdivide user home directories. Subdivision may be accomplished by using subdirectories such as `/home/staff`, `/home/guests`, `/home/students`, etc.
10. If you want to find out a user's home directory, you should use the `getpwent(3)` library function rather than relying on `/etc/passwd` because user information may be stored remotely using systems such as NIS.
11. It is recommended that apart from autosave and lock files programs should refrain from creating non dot files or directories in a home directory without user intervention.

12. Shared libraries that are only necessary for binaries in `/usr` (such as any X Window binaries) must not be in `/lib`. Only the shared libraries required to run binaries in `/bin` and `/sbin` may be here. In particular, the library `libm.so.*` may also be placed in `/usr/lib` if it is not required by anything in `/bin` or `/sbin`.
13. The usual placement of this binary is `/usr/bin/cpp`.
14. This is commonly used for 64-bit or 32-bit support on systems which support multiple binary formats, but require libraries of the same name. In this case, `/lib32` and `/lib64` might be the library directories, and `/lib` a symlink to one of them.
15. `/lib<qual>/cpp` is still permitted: this allows the case where `/lib` and `/lib<qual>` are the same (one is a symbolic link to the other).
16. A compliant implementation with two CDROM drives might have `/media/cdrom0` and `/media/cdrom1` with `/media/cdrom` a symlink to either of these.
17. If the home directory of the root account is not stored on the root partition it will be necessary to make certain it will default to `/` if it can not be located.

    We recommend against using the root account for tasks that can be performed as an unprivileged user, and that it be used solely for system administration. For this reason, we recommend that subdirectories for mail and other applications not appear in the root account's home directory, and that mail for administration roles such as root, postmaster, and webmaster be forwarded to an appropriate user.
18. Originally, `/sbin` binaries were kept in `/etc`.
19. Deciding what things go into *"sbin"* directories is simple: if a normal (not a system administrator) user will ever run it directly, then it must be placed in one of the *"bin"* directories. Ordinary users should not have to place any of the `sbin` directories in their path.

    For example, files such as **chfn** which users only occasionally use must still be placed in `/usr/bin`. **ping**, although it is absolutely necessary for root (network recovery and diagnosis) is often used by users and must live in `/bin` for that reason.

    We recommend that users have read and execute permission for everything in `/sbin` except, perhaps, certain setuid and setgid programs. The division between `/bin` and `/sbin` was not created for security reasons or to prevent users from seeing the operating system, but to provide a good partition between binaries that everyone uses and ones that are primarily used for administration tasks. There is no inherent security advantage in making `/sbin` off-limits for users.
20. This is particularly important as these areas will often contain both files initially installed by the distributor, and those added by the administrator.

# Chapter 4. The /usr Hierarchy

## 4.1. Purpose

`/usr` is the second major section of the filesystem. `/usr` is shareable, read-only data. That means that `/usr` should be shareable between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.

Large software packages must not use a direct subdirectory under the `/usr` hierarchy.

## 4.2. Requirements

The following directories, or symbolic links to directories, are required in `/usr`.

| Directory | Description |
|---|---|
| `bin` | Most user commands |
| `include` | Header files included by C programs |
| `lib` | Libraries |
| `local` | Local hierarchy (empty after main installation) |
| `sbin` | Non-vital system binaries |
| `share` | Architecture-independent data |

## 4.3. Specific Options

| Directory | Description |
|---|---|
| `X11R6` | XWindow System, version 11 release 6 (optional) |
| `games` | Games and educational binaries (optional) |
| `lib<qual>` | Alternate Format Libraries (optional) |
| `src` | Source code (optional) |

An exception is made for the X Window System because of considerable precedent and widely-accepted practice.

The following symbolic links to directories may be present. This possibility is based on the need to preserve compatibility with older systems until all implementations can be assumed to use the `/var` hierarchy.

```
/usr/spool -> /var/spool
/usr/tmp -> /var/tmp
/usr/spool/locks -> /var/lock
```

Once a system no longer requires any one of the above symbolic links, the link may be removed, if desired.

## 4.4. /usr/X11R6 : X Window System, Version 11 Release 6

# (optional)

## 4.4.1. Purpose

This hierarchy is reserved for the X Window System, version 11 release 6, and related files.

To simplify matters and make XFree86 more compatible with the X Window System on other systems, the following symbolic links must be present if `/usr/X11R6` exists:

```
/usr/bin/X11 -> /usr/X11R6/bin
/usr/lib/X11 -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

In general, software must not be installed or managed via the above symbolic links. They are intended for utilization by users only. The difficulty is related to the release version of the X Window System — in transitional periods, it is impossible to know what release of X11 is in use.

## 4.4.2. Specific Options

Host-specific data in `/usr/X11R6/lib/X11` should be interpreted as a demonstration file. Applications requiring information about the current host must reference a configuration file in `/etc/X11`, which may be linked to a file in `/usr/X11R6/lib`. [1]

# 4.5. /usr/bin : Most user commands

## 4.5.1. Purpose

This is the primary directory of executable commands on the system.

## 4.5.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/bin`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| `mh` | Commands for the MH mail handling system (optional) |

`/usr/bin/X11` must be a symlink to `/usr/X11R6/bin` if the latter exists.

The following files, or symbolic links to files, must be in `/usr/bin`, if the corresponding subsystem is installed:

| Command | Description |
|---|---|
| **perl** | The Practical Extraction and Report Language (optional) |
| **python** | The Python interpreted language (optional) |
| **tclsh** | Simple shell containing Tcl interpreter (optional) |
| **wish** | Simple Tcl/Tk windowing shell (optional) |

| Command | Description |
|---|---|
| **expect** | Program for interactive dialog (optional) |



**Rationale**

Because shell script interpreters (invoked with `#!<path>` on the first line of a shell script) cannot rely on a path, it is advantageous to standardize their locations. The Bourne shell and C-shell interpreters are already fixed in `/bin`, but Perl, Python, and Tcl are often found in many different places. They may be symlinks to the physical location of the shell interpreters.

# 4.6. /usr/include : Directory for standard include files.

## 4.6.1. Purpose

This is where all of the system's general-use include files for the C programming language should be placed.

## 4.6.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/include`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| bsd | BSD compatibility include files (optional) |

The symbolic link `/usr/include/X11` must link to `/usr/X11R6/include/X11` if the latter exists.

# 4.7. /usr/lib : Libraries for programming and packages

## 4.7.1. Purpose

`/usr/lib` includes object files, libraries, and internal binaries that are not intended to be executed directly by users or shell scripts. [2]

Applications may use a single subdirectory under `/usr/lib`. If an application uses a subdirectory, all architecture-dependent data exclusively used by the application must be placed within that subdirectory. [3]

## 4.7.2. Specific Options

For historical reasons, **/usr/lib/sendmail** must be a symbolic link to **/usr/sbin/sendmail** if the latter exists. [4]

If `/lib/X11` exists, `/usr/lib/X11` must be a symbolic link to `/lib/X11`, or to whatever `/lib/X11` is a symbolic link to. [5]

# 4.8. /usr/lib<qual> : Alternate format libraries (optional)

## 4.8.1. Purpose

`/usr/lib<qual>` performs the same role as `/usr/lib` for an alternate binary format, except that the symbolic links `/usr/lib<qual>/sendmail` and `/usr/lib<qual>/X11` are not required. [6]

## 4.8.2. /usr/local : Local hierarchy

### 4.8.2.1. Purpose

The `/usr/local` hierarchy is for use by the system administrator when installing software locally. It needs to be safe from being overwritten when the system software is updated. It may be used for programs and data that are shareable amongst a group of hosts, but not found in `/usr`.

Locally installed software must be placed within `/usr/local` rather than `/usr` unless it is being installed to replace or upgrade software in `/usr`. [7]

### 4.8.2.2. Requirements

The following directories, or symbolic links to directories, must be in `/usr/local`

| Directory | Description |
|---|---|
| `bin` | Local binaries |
| `etc` | Host-specific system configuration for local binaries |
| `games` | Local game binaries |
| `include` | Local C header files |
| `lib` | Local libraries |
| `man` | Local online manuals |
| `sbin` | Local system binaries |
| `share` | Local architecture-independent hierarchy |
| `src` | Local source code |

No other directories, except those listed below, may be in `/usr/local` after first installing a FHS-compliant system.

### 4.8.2.3. Specific Options

If directories `/lib<qual>` or `/usr/lib<qual>` exist, the equivalent directories must also exist in `/usr/local`.

`/usr/local/etc` may be a symbolic link to `/etc/local`.



**Rationale**

The consistency of `/usr/local/etc` is beneficial to installers, and is already used in other systems. As all of `/usr/local` needs to be backed up to reproduce a system, it introduces no additional maintenance overhead, but a symlink to `/etc/local` is suitable if systems want alltheir configuration under one hierarchy.

Note that `/usr/etc` is still not allowed: programs in `/usr` should place configuration files in `/etc`.

# 4.9. /usr/local/share

The requirements for the contents of this directory are the same as `/usr/share`. The only additional constraint is that `/usr/local/share/man` and `/usr/local/man` directories must be synonomous (usually this means that one of them must be a symbolic link). [8]

# 4.10. /usr/sbin : Non-essential standard system binaries

## 4.10.1. Purpose

This directory contains any non-essential binaries used exclusively by the system administrator. System administration programs that are required for system repair, system recovery, mounting `/usr`, or other essential functions must be placed in `/sbin` instead. [9]

# 4.11. /usr/share : Architecture-independent data

## 4.11.1. Purpose

The `/usr/share` hierarchy is for all read-only architecture independent data files. [10]

This hierarchy is intended to be shareable among all architecture platforms of a given OS; thus, for example, a site with i386, Alpha, and PPC platforms might maintain a single `/usr/share` directory that is centrally-mounted. Note, however, that `/usr/share` is generally not intended to be shared by different OSes or by different releases of the same OS.

Any program or package which contains or requires data that doesn't need to be modified should store that data in `/usr/share` (or `/usr/local/share`, if installed locally). It is recommended that a subdirectory be used in `/usr/share` for this purpose.

Game data stored in `/usr/share/games` must be purely static data. Any modifiable files, such as score files, game play logs, and so forth, should be placed in `/var/games`.

## 4.11.2. Requirements

The following directories, or symbolic links to directories, must be in `/usr/share`

| Directory | Description |
|---|---|
| man | Online manuals |
| misc | Miscellaneous architecture-independent data |

## 4.11.3. Specific Options

The following directories, or symbolic links to directories, must be in /usr/share, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| dict | Word lists (optional) |
| doc | Miscellaneous documentation (optional) |
| games | Static data files for /usr/games (optional) |
| info | GNU Info system s primary directory (optional) |
| locale | Locale information (optional) |
| nls | Message catalogs for Native language support (optional) |
| sgml | SGML data (optional) |
| terminfo | Directories for terminfo database (optional) |
| tmac | troff macros not distributed with groff (optional) |
| xml | XML data (optional) |
| zoneinfo | Timezone information and configuration (optional) |

It is recommended that application-specific, architecture-independent directories be placed here. Such directories include **groff**, **perl**, **ghostscript**, **texmf**, and **kbd** (Linux) or **syscons** (BSD). They may, however, be placed in /usr/lib for backwards compatibility, at the distributor's discretion. Similarly, a /usr/lib/games hierarchy may be used in addition to the /usr/share/games hierarchy if the distributor wishes to place some game data there.

## 4.11.4. /usr/share/dict : Word lists (optional)

### 4.11.4.1. Purpose

This directory is the home for word lists on the system; Traditionally this directory contains only the English words file, which is used by **look(1)** and various spelling programs. words may use either American or British spelling.



**Rationale**

The reason that only word lists are located here is that they are the only files common to all spell checkers.

### 4.11.4.2. Specific Options

The following files, or symbolic links to files, must be in `/usr/share/dict`, if the corresponding subsystem is installed:

| File | Description |
|---|---|
| `words` | List of English words (optional) |

Sites that require both American and British spelling may link `words` to `/usr/share/dict/american-english` or `/usr/share/dict/british-english`.

Word lists for other languages may be added using the English name for that language, e.g., `/usr/share/dict/french`, `/usr/share/dict/danish`, etc. These should, if possible, use an ISO 8859 character set which is appropriate for the language in question; if possible the Latin1 (ISO 8859-1) character set should be used (this is often not possible).

Other word lists must be included here, if present.

## 4.11.5. /usr/share/man : Manual pages

### 4.11.5.1. Purpose

This section details the organization for manual pages throughout the system, including `/usr/share/man`. Also refer to the section on `/var/cache/man`.

The primary `<mandir>` of the system is `/usr/share/man`. `/usr/share/man` contains manual information for commands and data under the `/` and `/usr` filesystems. [11]

Manual pages are stored in `<mandir>/<locale>/man<section>/<arch>`. An explanation of `<mandir>`, `<locale>`, `<section>`, and `<arch>` is given below.

A description of each section follows:

- `man1`: User programs Manual pages that describe publicly accessible commands are contained in this chapter. Most program documentation that a user will need to use is located here.
- `man2`: System calls This section describes all of the system calls (requests for the kernel to perform operations).
- `man3`: Library functions and subroutines Section 3 describes program library routines that are not direct calls to kernel services. This and chapter 2 are only really of interest to programmers.
- `man4`: Special files Section 4 describes the special files, related driver functions, and networking support available in the system. Typically, this includes the device files found in `/dev` and the kernel interface to networking protocol support.
- `man5`: File formats The formats for many data files are documented in the section 5. This includes various include files, program output files, and system files.
- `man6`: Games This chapter documents games, demos, and generally trivial programs. Different people have various notions about how essential this is.
- `man7`: Miscellaneous Manual pages that are difficult to classify are designated as being section 7. The troff and other text processing macro packages are found here.
- `man8`: System administration Programs used by system administrators for system operation and maintenance are documented here. Some of these programs are also occasionally useful for normal users.

### 4.11.5.2. Specific Options

The following directories, or symbolic links to directories, must be in `/usr/share/<mandir>/<locale>`, unless they are empty: [12]

| Directory | Description |
|---|---|
| `man1` | User programs (optional) |
| `man2` | System calls (optional) |
| `man3` | Library calls (optional) |
| `man4` | Special files (optional) |
| `man5` | File formats (optional) |
| `man6` | Games (optional) |
| `man7` | Miscellaneous (optional) |
| `man8` | System administration (optional) |

The component `<section>` describes the manual section.

Provisions must be made in the structure of `/usr/share/man` to support manual pages which are written in different (or multiple) languages. These provisions must take into account the storage and reference of these manual pages. Relevant factors include language (including geographical-based differences), and character code set.

This naming of language subdirectories of `/usr/share/man` is based on Appendix E of the POSIX 1003.1 standard which describes the locale identification string — the most well-accepted method to describe a cultural environment. The `<locale>` string is:

`<language>[_<territory>][.<character-set>][,<version>]`

The `<language>` field must be taken from ISO 639 (a code for the representation of names of languages). It must be two characters wide and specified with lowercase letters only.

The `<territory>` field must be the two-letter code of ISO 3166 (a specification of representations of countries), if possible. (Most people are familiar with the two-letter codes used for the country codes in email addresses.) It must be two characters wide and specified with uppercase letters only. [13]

The `<character-set>` field must represent the standard describing the character set. If the `<character-set>` field is just a numeric specification, the number represents the number of the international standard describing the character set. It is recommended that this be a numeric representation if possible (ISO standards, especially), not include additional punctuation symbols, and that any letters be in lowercase.

A parameter specifying a `<version>` of the profile may be placed after the `<character-set>` field, delimited by a comma. This may be used to discriminate between different cultural needs; for instance, dictionary order versus a more systems-oriented collating order. This standard recommends not using the `<version>` field, unless it is necessary.

Systems which use a unique language and code set for all manual pages may omit the `<locale>` substring and store all manual pages in `<mandir>`. For example, systems which only have English manual pages coded with ASCII, may store manual pages (the `man<section>` directories) directly in `/usr/share/man`. (That is the traditional circumstance and arrangement, in fact.)

Countries for which there is a well-accepted standard character code set may omit the `<character-set>` field, but it is strongly recommended that it be included, especially for countries with several competing standards.

Various examples:

| Language | Territory | Character Set | Directory |
|---|---|---|---|
| English | — | ASCII | /usr/share/man/en |

| Language | Territory | Character Set | Directory |
|---|---|---|---|
| English | United Kingdom | ISO 8859-15 | /usr/share/man/en_GB |
| English | United States | ASCII | /usr/share/man/en_US |
| French | Canada | ISO 8859-1 | /usr/share/man/fr_CA |
| French | France | ISO 8859-1 | /usr/share/man/fr_FR |
| German | Germany | ISO 646 | /usr/share/man/de_DE.646 |
| German | Germany | ISO 6937 | /usr/share/man/de_DE.6937 |
| German | Germany | ISO 8859-1 | /usr/share/man/de_DE.88591 |
| German | Switzerland | ISO 646 | /usr/share/man/de_CH.646 |
| Japanese | Japan | JIS | /usr/share/man/ja_JP.jis |
| Japanese | Japan | SJIS | /usr/share/man/ja_JP.sjis |
| Japanese | Japan | UJIS (or EUC-J) | /usr/share/man/ja_JP.ujis |

Similarly, provision must be made for manual pages which are architecture-dependent, such as documentation on device-drivers or low-level system administration commands. These must be placed under an `<arch>` directory in the appropriate `man<section>` directory; for example, a man page for the i386 ctrlaltdel(8) command might be placed in `/usr/share/man/<locale>/man8/i386/ctrlaltdel.8`.

Manual pages for commands and data under `/usr/local` are stored in `/usr/local/man`. Manual pages for X11R6 are stored in `/usr/X11R6/man`. It follows that all manual page hierarchies in the system must have the same structure as `/usr/share/man`.

The cat page sections (`cat<section>`) containing formatted manual page entries are also found within subdirectories of `<mandir>/<locale>`, but are not required nor may they be distributed in lieu of nroff source manual pages.

The numbered sections "1" through "8" are traditionally defined. In general, the file name for manual pages located within a particular section end with `.<section>`.

In addition, some large sets of application-specific manual pages have an additional suffix appended to the manual page filename. For example, the MH mail handling system manual pages must have `mh` appended to all MH manuals. All X Window System manual pages must have an `x` appended to the filename.

The practice of placing various language manual pages in appropriate subdirectories of `/usr/share/man` also applies to the other manual page hierarchies, such as `/usr/local/man` and `/usr/X11R6/man`. (This portion of the standard also applies later in the section on the optional `/var/cache/man` structure.)

## 4.11.6. /usr/share/misc : Miscellaneous architecture-independent data

This directory contains miscellaneous architecture-independent files which don’t require a separate subdirectory under `/usr/share`.

### 4.11.6.1. Specific Options

The following files, or symbolic links to files, must be in `/usr/share/misc`, if the corresponding subsystem is installed:

| File | Description |
|---|---|
| `ascii` | ASCII character set table (optional) |

| File | Description |
|---|---|
| magic | Default list of magic numbers for the file command (optional) |
| termcap | Terminal capability database (optional) |
| termcap.db | Terminal capability database (optional) |

Other (application-specific) files may appear here, but a distributor may place them in /usr/lib at their discretion. [14]

## 4.11.7. /usr/share/sgml : SGML data (optional)

### 4.11.7.1. Purpose

/usr/share/sgml contains architecture-independent files used by SGML applications, such as ordinary catalogs (not the centralized ones, see /etc/sgml), DTDs, entities, or style sheets.

### 4.11.7.2. Specific Options

The following directories, or symbolic links to directories, must be in /usr/share/sgml, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| docbook | docbook DTD (optional) |
| tei | tei DTD (optional) |
| html | html DTD (optional) |
| mathml | mathml DTD (optional) |

Other files that are not specific to a given DTD may reside in their own subdirectory.

## 4.11.8. /usr/share/xml : XML data (optional)

### 4.11.8.1. Purpose

/usr/share/xml contains architecture-independent files used by XML applications, such as ordinary catalogs (not the centralized ones, see /etc/sgml), DTDs, entities, or style sheets.

### 4.11.8.2. Specific Options

The following directories, or symbolic links to directories, must be in /usr/share/xml, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| docbook | docbook XML DTD (optional) |
| xhtml | XHTML DTD (optional) |

| Directory | Description |
| --- | --- |
| `mathml` | MathML DTD (optional) |

## 4.12. /usr/src : Source code (optional)

### 4.12.1. Purpose

Source code may be place placed in this subdirectory, only for reference purposes. [15]

# Notes

1. Examples of such configuration files include `Xconfig`, `XF86Config`, or `system.twmrc`)
2. Miscellaneous architecture-independent application-specific static files and subdirectories must be placed in `/usr/share`.
3. For example, the `perl5` subdirectory for Perl 5 modules and libraries.
4. Some executable commands such as **makewhatis** and **sendmail** have also been traditionally placed in `/usr/lib`. **makewhatis** is an internal binary and must be placed in a binary directory; users access only **catman**. Newer **sendmail** binaries are now placed by default in `/usr/sbin`. Additionally, systems using a *sendmail*-compatible mail transfer agent must provide **/usr/sbin/sendmail** as a symbolic link to the appropriate executable.
5. Host-specific data for the X Window System must not be stored in `/usr/lib/X11`. Host-specific configuration files such as `Xconfig` or `XF86Config` must be stored in `/etc/X11`. This includes configuration data such as `system.twmrc` even if it is only made a symbolic link to a more global configuration file (probably in `/usr/X11R6/lib/X11`).
6. The case where `/usr/lib` and `/usr/lib<qual>` are the same (one is a symbolic link to the other) these files and the per-application subdirectories will exist.
7. Software placed in `/` or `/usr` may be overwritten by system upgrades (though we recommend that distributions do not overwrite data in `/etc` under these circumstances). For this reason, local software must not be placed outside of `/usr/local` without good reason.
8. `/usr/local/man` may be deprecated in future FHS releases, so if all else is equal, making that one a symlink seems sensible.
9. Locally installed system administration programs should be placed in `/usr/local/sbin`.
10. Much of this data originally lived in `/usr` (`man`, `doc`) or `/usr/lib` (`dict`, `terminfo`, `zoneinfo`).
11. Obviously, there are no manual pages in `/` because they are not required at boot time nor are they required in emergencies. Really.
12. For example, if `/usr/local/man` has no manual pages in section 4 (Devices), then `/usr/local/man/man4` may be omitted.
13. A major exception to this rule is the United Kingdom, which is ‘GB’ in the ISO 3166, but ‘UK’ for most email addresses.

14. Some such files include: `airport`, `birthtoken`, `eqnchar`, `getopt`, `gprof.callg`, `gprof.flat`, `inter.phone`, `ipfw.samp.filters`, `ipfw.samp.scripts`, `keycap.pcvt`, `mail.help`, `mail.tildehelp`, `man.template`, `map3270`, `mdoc.template`, `more.help`, `na.phone`, `nslookup.help`, `operator`, `scsi_modes`, `sendmail.hf`, `style`, `units.lib`, `vgrindefs`, `vgrindefs.db`, `zipcodes`
15. Generally, source should not be built within this hierarchy.

# Chapter 5. The /var Hierarchy

## 5.1. Purpose

`/var` contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files.

Some portions of `/var` are not shareable between different systems. For instance, `/var/log`, `/var/lock`, and `/var/run`. Other portions may be shared, notably `/var/mail`, `/var/cache/man`, `/var/cache/fonts`, and `/var/spool/news`.

`/var` is specified here in order to make it possible to mount `/usr` read-only. Everything that once went into `/usr` that is written to during system operation (as opposed to installation and software maintenance) must be in `/var`.

If `/var` cannot be made a separate partition, it is often preferable to move `/var` out of the root partition and into the `/usr` partition. (This is sometimes done to reduce the size of the root partition or when space runs low in the root partition.) However, `/var` must not be linked to `/usr` because this makes separation of `/usr` and `/var` more difficult and is likely to create a naming conflict. Instead, link `/var` to `/usr/var`.

Applications must generally not add directories to the top level of `/var`. Such directories should only be added if they have some system-wide implication, and in consultation with the FHS mailing list.

## 5.2. Requirements

The following directories, or symbolic links to directories, are required in `/var`.

| Directory | Description |
|---|---|
| `cache` | Application cache data |
| `lib` | Variable state information |
| `local` | Variable data for /usr/local |
| `lock` | Lock files |
| `log` | Log files and directories |
| `opt` | Variable data for /opt |
| `run` | Data relevant to running processes |
| `spool` | Application spool data |
| `tmp` | Temporary files preserved between system reboots |

Several directories are ‘reserved’ in the sense that they must not be used arbitrarily by some new application, since they would conflict with historical and/or local practice. They are:

```
/var/backups
/var/cron
/var/msgs
/var/preserve
```

## 5.3. Specific Options

The following directories, or symbolic links to directories, must be in `/var`, if the corresponding subsystem is

installed:

| Directory | Description |
|---|---|
| `account` | Process accounting logs (optional) |
| `crash` | System crash dumps (optional) |
| `games` | Variable game data (optional) |
| `mail` | User mailbox files (optional) |
| `yp` | Network Information Service (NIS) database files (optional) |

# 5.4. /var/account : Process accounting logs (optional)

## 5.4.1. Purpose

This directory holds the current active process accounting log and the composite process usage data (as used in some UNIX-like systems by **lastcomm** and **sa**).

# 5.5. /var/cache : Application cache data

## 5.5.1. Purpose

`/var/cache` is intended for cached data from applications. Such data is locally generated as a result of time-consuming I/O or calculation. The application must be able to regenerate or restore the data. Unlike `/var/spool`, the cached files can be deleted without data loss. The data must remain valid between invocations of the application and rebooting the system.

Files located under `/var/cache` may be expired in an application specific manner, by the system administrator, or both. The application must always be able to recover from manual deletion of these files (generally because of a disk space shortage). No other requirements are made on the data format of the cache directories.



**Rationale**

The existence of a separate directory for cached data allows system administrators to set different disk and backup policies from other directories in `/var`.

## 5.5.2. Specific Options

| Directory | Description |
|---|---|
| `fonts` | Locally-generated fonts (optional) |
| `man` | Locally-formatted manual pages (optional) |

| Directory | Description |
| --- | --- |
| www | WWW proxy or cache data (optional) |
| <package> | Package specific cache data (optional) |

## 5.5.3. /var/cache/fonts : Locally-generated fonts (optional)

### 5.5.3.1. Purpose

The directory /var/cache/fonts should be used to store any dynamically-created fonts. In particular, all of the fonts which are automatically generated by **mktexpk** must be located in appropriately-named subdirectories of /var/cache/fonts. [1]

### 5.5.3.2. Specific Options

Other dynamically created fonts may also be placed in this tree, under appropriately-named subdirectories of /var/cache/fonts.

## 5.5.4. /var/cache/man : Locally-formatted manual pages (optional)

### 5.5.4.1. Purpose

This directory provides a standard location for sites that provide a read-only /usr partition, but wish to allow caching of locally-formatted man pages. Sites that mount /usr as writable (e.g., single-user installations) may choose not to use /var/cache/man and may write formatted man pages into the cat<section> directories in /usr/share/man directly. We recommend that most sites use one of the following options instead:

- Preformat all manual pages alongside the unformatted versions.
- Allow no caching of formatted man pages, and require formatting to be done each time a man page is brought up.
- Allow local caching of formatted man pages in /var/cache/man.

The structure of /var/cache/man needs to reflect both the fact of multiple man page hierarchies and the possibility of multiple language support.

Given an unformatted manual page that normally appears in <path>/man/<locale>/man<section>, the directory to place formatted man pages in is /var/cache/man/<catpath>/<locale>/cat<section>, where <catpath> is derived from <path> by removing any leading usr and/or trailing share pathname components. (Note that the <locale> component may be missing.) [2]

Man pages written to /var/cache/man may eventually be transferred to the appropriate preformatted directories in the source man hierarchy or expired; likewise formatted man pages in the source man hierarchy may be expired if they are not accessed for a period of time.

If preformatted manual pages come with a system on read-only media (a CD-ROM, for instance), they must be installed in the source man hierarchy (e.g. /usr/share/man/cat<section>). /var/cache/man is reserved as a writable cache for formatted manual pages.



**Rationale**

Release 1.2 of the standard specified `/var/catman` for this hierarchy. The path has been moved under `/var/cache` to better reflect the dynamic nature of the formatted man pages. The directory name has been changed to `man` to allow for enhancing the hierarchy to include post-processed formats other than "cat", such as PostScript, HTML, or DVI.

# 5.6. /var/crash : System crash dumps (optional)

## 5.6.1. Purpose

This directory holds system crash dumps. As of the date of this release of the standard, system crash dumps were not supported under Linux but may be supported by other systems which may comply with the FHS.

# 5.7. /var/games : Variable game data (optional)

## 5.7.1. Purpose

Any variable data relating to games in `/usr` should be placed here. `/var/games` should hold the variable data previously found in `/usr`; static data, such as help text, level descriptions, and so on, must remain elsewhere, such as `/usr/share/games`.



**Rationale**

`/var/games` has been given a hierarchy of its own, rather than leaving it merged in with the old `/var/lib` as in release 1.2. The separation allows local control of backup strategies, permissions, and disk usage, as well as allowing inter-host sharing and reducing clutter in `/var/lib`. Additionally, `/var/games` is the path traditionally used by BSD.

# 5.8. /var/lib : Variable state information

## 5.8.1. Purpose

This hierarchy holds state information pertaining to an application or the system. State information is data that programs modify while they run, and that pertains to one specific host. Users must never need to modify files in `/var/lib` to configure a package’s operation.

State information is generally used to preserve the condition of an application (or a group of inter-related applications) between invocations and between different instances of the same application. State information should generally remain valid after a reboot, should not be logging output, and should not be spooled data.

An application (or a group of inter-related applications) must use a subdirectory of `/var/lib` for its data. There is one required subdirectory, `/var/lib/misc`, which is intended for state files that don't need a subdirectory; the other subdirectories should only be present if the application in question is included in the distribution. [3]

`/var/lib/<name>` is the location that must be used for all distribution packaging support. Different distributions may use different names, of course.

## 5.8.2. Requirements

The following directories, or symbolic links to directories, are required in `/var/lib`:

| Directory | Description |
|---|---|
| `misc` | Miscellaneous state data |

## 5.8.3. Specific Options

The following directories, or symbolic links to directories, must be in `/var/lib`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| <editor> | Editor backup files and state (optional) |
| <pkgtool> | Packaging support files (optional) |
| <package> | State data for packages and subsystems (optional) |
| **hwclock** | State directory for hwclock (optional) |
| **xdm** | X display manager variable data (optional) |

## 5.8.4. /var/lib/<editor> : Editor backup files and state (optional)

### 5.8.4.1. Purpose

These directories contain saved files generated by any unexpected termination of an editor (e.g., **elvis**, **jove**, **nvi**).

Other editors may not require a directory for crash-recovery files, but may require a well-defined place to store other information while the editor is running. This information should be stored in a subdirectory under `/var/lib` (for example, GNU Emacs would place lock files in `/var/lib/emacs/lock`).

Future editors may require additional state information beyond crash-recovery files and lock files — this information should also be placed under `/var/lib/<editor>`.



**Rationale**

Previous Linux releases, as well as all commercial vendors, use `/var/preserve` for vi or its clones. However, each editor uses its own format for these crash-recovery files, so a separate directory is needed for each editor.

Editor-specific lock files are usually quite different from the device or resource lock files that are stored in `/var/lock` and, hence, are stored under `/var/lib`.

## 5.8.5. /var/lib/hwclock : State directory for hwclock (optional)

### 5.8.5.1. Purpose

This directory contains the file `/var/lib/hwclock/adjtime`.

 **Rationale**

In FHS 2.1, this file was `/etc/adjtime`, but as **hwclock** updates it, that was obviously incorrect.

## 5.8.6. /var/lib/misc : Miscellaneous variable data

### 5.8.6.1. Purpose

This directory contains variable data not placed in a subdirectory in `/var/lib`. An attempt should be made to use relatively unique names in this directory to avoid namespace conflicts. [4]

# 5.9. /var/lock : Lock files

## 5.9.1. Purpose

Lock files should be stored within the `/var/lock` directory structure.

Lock files for devices and other resources shared by multiple applications, such as the serial device lock files that were originally found in either `/usr/spool/locks` or `/usr/spool/uucp`, must now be stored in `/var/lock`. The naming convention which must be used is "LCK.." followed by the base name of the device. For example, to lock /dev/ttyS0 the file "LCK..ttyS0" would be created. [5]

The format used for the contents of such lock files must be the HDB UUCP lock file format. The HDB format is to store the process identifier (PID) as a ten byte ASCII decimal number, with a trailing newline. For example, if process 1230 holds a lock file, it would contain the eleven characters: space, space, space, space, space, space, one, two, three, zero, and newline.

# 5.10. /var/log : Log files and directories

## 5.10.1. Purpose

This directory contains miscellaneous log files. Most logs must be written to this directory or an appropriate subdirectory.

## 5.10.2. Specific Options

The following files, or symbolic links to files, must be in `/var/log`, if the corresponding subsystem is installed:

| File | Description |
|---|---|
| `lastlog` | record of last login of each user |
| `messages` | system messages from **syslogd** |
| `wtmp` | record of all logins and logouts |

# 5.11. /var/mail : User mailbox files (optional)

## 5.11.1. Purpose

The mail spool must be accessible through `/var/mail` and the mail spool files must take the form `<username>`. [6]

User mailbox files in this location must be stored in the standard UNIX mailbox format.



**Rationale**

The logical location for this directory was changed from `/var/spool/mail` in order to bring FHS in-line with nearly every UNIX implementation. This change is important for inter-operability since a single `/var/mail` is often shared between multiple hosts and multiple UNIX implementations (despite NFS locking issues).

It is important to note that there is no requirement to physically move the mail spool to this location. However, programs and header files must be changed to use `/var/mail`.

# 5.12. /var/opt : Variable data for /opt

## 5.12.1. Purpose

Variable data of the packages in `/opt` must be installed in `/var/opt/<subdir>`, where `<subdir>` is the name of the subtree in `/opt` where the static data from an add-on software package is stored, except where superseded by another file in `/etc`. No structure is imposed on the internal arrangement of `/var/opt/<subdir>`.


**Rationale**

Refer to the rationale for `/opt`.

# 5.13. /var/run : Run-time variable data

## 5.13.1. Purpose

This directory contains system information data describing the system since it was booted. Files under this directory must be cleared (removed or truncated as appropriate) at the beginning of the boot process. Programs may have a subdirectory of `/var/run`; this is encouraged for programs that use more than one run-time file. [7] Process identifier (PID) files, which were originally placed in `/etc`, must be placed in `/var/run`. The naming convention for PID files is `<program-name>.pid`. For example, the **crond** PID file is named `/var/run/crond.pid`.

## 5.13.2. Requirements

The internal format of PID files remains unchanged. The file must consist of the process identifier in ASCII-encoded decimal, followed by a newline character. For example, if **crond** was process number 25, `/var/run/crond.pid` would contain three characters: two, five, and newline.

Programs that read PID files should be somewhat flexible in what they accept; i.e., they should ignore extra whitespace, leading zeroes, absence of the trailing newline, or additional lines in the PID file. Programs that create PID files should use the simple specification located in the above paragraph.

The `utmp` file, which stores information about who is currently using the system, is located in this directory.

System programs that maintain transient UNIX-domain sockets must place them in this directory.

# 5.14. /var/spool : Application spool data

## 5.14.1. Purpose

`/var/spool` contains data which is awaiting some kind of later processing. Data in `/var/spool` represents work to be done in the future (by a program, user, or administrator); often data is deleted after it has been processed. [8]

## 5.14.2. Specific Options

The following directories, or symbolic links to directories, must be in `/var/spool`, if the corresponding subsystem is installed:

| Directory | Description |
|---|---|
| lpd | Printer spool directory (optional) |
| mqueue | Outgoing mail queue (optional) |
| news | News spool directory (optional) |
| rwho | Rwhod files (optional) |
| uucp | Spool directory for UUCP (optional) |

## 5.14.3. /var/spool/lpd : Line-printer daemon print queues (optional)

### 5.14.3.1. Purpose

The lock file for **lpd**, `lpd.lock`, must be placed in `/var/spool/lpd`. It is suggested that the lock file for each printer be placed in the spool directory for that specific printer and named `lock`.

### 5.14.3.2. Specific Options

| Directory | Description |
|---|---|
| printer | Spools for a specific printer (optional) |

## 5.14.4. /var/spool/rwho : Rwhod files (optional)

### 5.14.4.1. Purpose

This directory holds the **rwhod** information for other systems on the local net.



**Rationale**

Some BSD releases use `/var/rwho` for this data; given its historical location in `/var/spool` on other systems and its approximate fit to the definition of ‘spooled’ data, this location was deemed more appropriate.

# 5.15. /var/tmp : Temporary files preserved between system reboots

## 5.15.1. Purpose

The `/var/tmp` directory is made available for programs that require temporary files or directories that are

preserved between system reboots. Therefore, data stored in `/var/tmp` is more persistent than data in `/tmp`.

Files and directories located in `/var/tmp` must not be deleted when the system is booted. Although data stored in `/var/tmp` is typically deleted in a site-specific manner, it is recommended that deletions occur at a less frequent interval than `/tmp`.

# 5.16. /var/yp : Network Information Service (NIS) database files (optional)

## 5.16.1. Purpose

Variable data for the Network Information Service (NIS), formerly known as the Sun Yellow Pages (YP), must be placed in this directory.

 **Rationale**

`/var/yp` is the standard directory for NIS (YP) data and is almost exclusively used in NIS documentation and systems. [9]

# Notes

1. This standard does not currently incorporate the TeX Directory Structure (a document that describes the layout TeX files and directories), but it may be useful reading. It is located at ftp://ctan.tug.org/tex/
2. For example, `/usr/share/man/man1/ls.1` is formatted into `/var/cache/man/cat1/ls.1`, and `/usr/X11R6/man/<locale>/man3/XtClass.3x` into `/var/cache/man/X11R6/<locale>/cat3/XtClass.3x`.
3. An important difference between this version of this standard and previous ones is that applications are now required to use a subdirectory of `/var/lib`.
4. This hierarchy should contain files stored in `/var/db` in current BSD releases. These include `locate.database` and `mountdtab`, and the kernel symbol database(s).
5. Then, anything wishing to use `/dev/ttyS0` can read the lock file and act accordingly (all locks in `/var/lock` should be world-readable).
6. Note that `/var/mail` may be a symbolic link to another directory.
7. `/var/run` should be unwritable for unprivileged users (root or users running daemons); it is a major security problem if any user can write in this directory.
8. UUCP lock files must be placed in `/var/lock`. See the above section on `/var/lock`.
9. NIS should not be confused with Sun NIS+, which uses a different directory, `/var/nis`.

# Chapter 6. Operating System Specific Annex

This section is for additional requirements and recommendations that only apply to a specific operating system. The material in this section should never conflict with the base standard.

## 6.1. Linux

This is the annex for the Linux operating system.

### 6.1.1. / : Root directory

On Linux systems, if the kernel is located in `/`, we recommend using the names `vmlinux` or `vmlinuz`, which have been used in recent Linux kernel source packages.

### 6.1.2. /bin : Essential user command binaries (for use by all users)

Linux systems which require them place these additional files into `/bin`:

- **setserial**

### 6.1.3. /dev : Devices and special files

The following devices must exist under /dev.

`/dev/null`

All data written to this device is discarded. A read from this device will return an EOF condition.

`/dev/zero`

This device is a source of zeroed out data. All data written to this device is discarded. A read from this device will return as many bytes containing the value zero as was requested.

`/dev/tty`

This device is a synonym for the controlling terminal of a process. Once this device is opened, all reads and writes will behave as if the actual controlling terminal device had been opened.

 **Rationale**

Previous versions of the FHS had stricter requirements for `/dev`. Other devices may also exist in /dev. Device names may exist as symbolic links to other device nodes located in /dev or subdirectories of /dev. There is no requirement concerning major/minor number values.

### 6.1.4. /etc : Host-specific system configuration

Linux systems which require them place these additional files into `/etc`.

- `lilo.conf`

### 6.1.5. /lib64 and /lib32 : 64/32-bit libraries (architecture dependent)

The 64-bit architectures PPC64, s390x, sparc64 and AMD64 must place 64-bit libraries in `/lib64`, and 32-bit (or 31-bit on s390) libraries in `/lib`.

The 64-bit architecture IA64 must place 64-bit libraries in `/lib`.



**Rationale**

This is a refinement of the general rules for `/lib<qual>` and `/usr/lib<qual>`. The architectures PPC64, s390x, sparc64 and AMD64 support support both 32-bit (for s390 more precise 31-bit) and 64-bit programs. Using `lib` for 32-bit binaries allows existing binaries from the 32-bit systems to work without any changes: such binaries are expected to be numerous. IA-64 uses a different scheme, reflecting the deprecation of 32-bit binaries (and hence libraries) on that architecture.

### 6.1.6. /proc : Kernel and process information virtual filesystem

The `proc` filesystem is the de-facto standard Linux method for handling process and system information, rather than `/dev/kmem` and other similar methods. We strongly encourage this for the storage and retrieval of process information as well as other kernel and memory information.

### 6.1.7. /sbin : Essential system binaries

Linux systems place these additional files into `/sbin`.

- Second extended filesystem commands (optional):
  - **badblocks**
  - **dumpe2fs**
  - **e2fsck**
  - **mke2fs**
  - **mklost+found**
  - **tune2fs**

- Boot-loader map installer (optional):
  - **lilo**

Optional files for /sbin:

- Static binaries:
  - **ldconfig**
  - **sln**
  - **ssync**

  Static **ln** (**sln**) and static **sync** (**ssync**) are useful when things go wrong. The primary use of **sln** (to repair incorrect symlinks in `/lib` after a poorly orchestrated upgrade) is no longer a major concern now that the **ldconfig** program (usually located in `/usr/sbin`) exists and can act as a guiding hand in upgrading the dynamic libraries. Static **sync** is useful in some emergency situations. Note that these need not be statically linked versions of the standard **ln** and **sync**, but may be.

  The **ldconfig** binary is optional for `/sbin` since a site may choose to run **ldconfig** at boot time, rather than only when upgrading the shared libraries. (It's not clear whether or not it is advantageous to run **ldconfig** on each boot.) Even so, some people like **ldconfig** around for the following (all too common) situation:

  1. I've just removed `/lib/<file>`.
  2. I can't find out the name of the library because **ls** is dynamically linked, I'm using a shell that doesn't have **ls** built-in, and I don't know about using "**echo ***" as a replacement.
  3. I have a static **sln**, but I don't know what to call the link.

- Miscellaneous:
  - **ctrlaltdel**
  - **kbdrate**

  So as to cope with the fact that some keyboards come up with such a high repeat rate as to be unusable, **kbdrate** may be installed in `/sbin` on some systems.

  Since the default action in the kernel for the Ctrl-Alt-Del key combination is an instant hard reboot, it is generally advisable to disable the behavior before mounting the root filesystem in read-write mode. Some **init** suites are able to disable Ctrl-Alt-Del, but others may require the **ctrlaltdel** program, which may be installed in `/sbin` on those systems.

## 6.1.8. /usr/include : Header files included by C programs

These symbolic links are required if a C or C++ compiler is installed and only for systems not based on glibc.

```
/usr/include/asm -> /usr/src/linux/include/asm-<arch>
/usr/include/linux -> /usr/src/linux/include/linux
```

## 6.1.9. /usr/src : Source code

For systems based on glibc, there are no specific guidelines for this directory. For systems based on Linux libc revisions prior to glibc, the following guidelines and rationale apply:

The only source code that should be placed in a specific location is the Linux kernel source code. It is located in `/usr/src/linux`.

If a C or C++ compiler is installed, but the complete Linux kernel source code is not installed, then the include files from the kernel source code must be located in these directories:

```
/usr/src/linux/include/asm-<arch>
/usr/src/linux/include/linux
```

`<arch>` is the name of the system architecture.

 **Note**

`/usr/src/linux` may be a symbolic link to a kernel source code tree.

 **Rationale**

It is important that the kernel include files be located in `/usr/src/linux` and not in `/usr/include` so there are no problems when system administrators upgrade their kernel version for the first time.

## 6.1.10. /var/spool/cron : cron and at jobs

This directory contains the variable data for the **cron** and **at** programs.

# Chapter 7. Appendix

## 7.1. The FHS mailing list

The FHS mailing list is located at <freestandards-fhs-discuss@lists.sourceforge.net>. You can subscribe to the mailing list at this page http://sourceforge.net/projects/freestandards/.

Thanks to Network Operations at the University of California at San Diego who allowed us to use their excellent mailing list server.

As noted in the introduction, please do not send mail to the mailing list without first contacting the FHS editor or a listed contributor.

## 7.2. Background of the FHS

The process of developing a standard filesystem hierarchy began in August 1993 with an effort to restructure the file and directory structure of Linux. The FSSTND, a filesystem hierarchy standard specific to the Linux operating system, was released on February 14, 1994. Subsequent revisions were released on October 9, 1994 and March 28, 1995.

In early 1995, the goal of developing a more comprehensive version of FSSTND to address not only Linux, but other UNIX-like systems was adopted with the help of members of the BSD development community. As a result, a concerted effort was made to focus on issues that were general to UNIX-like systems. In recognition of this widening of scope, the name of the standard was changed to Filesystem Hierarchy Standard or FHS for short.

Volunteers who have contributed extensively to this standard are listed at the end of this document. This standard represents a consensus view of those and other contributors.

## 7.3. General Guidelines

Here are some of the guidelines that have been used in the development of this standard:

- Solve technical problems while limiting transitional difficulties.
- Make the specification reasonably stable.
- Gain the approval of distributors, developers, and other decision-makers in relevant development groups and encourage their participation.
- Provide a standard that is attractive to the implementors of different UNIX-like systems.

## 7.4. Scope

This document specifies a standard filesystem hierarchy for FHS filesystems by specifying the location of files and directories, and the contents of some system files.

This standard has been designed to be used by system integrators, package developers, and system administrators in the construction and maintenance of FHS compliant filesystems. It is primarily intended to be a reference and is not a tutorial on how to manage a conforming filesystem hierarchy.

The FHS grew out of earlier work on FSSTND, a filesystem organization standard for the Linux operating system. It builds on FSSTND to address interoperability issues not just in the Linux community but in a wider arena including 4.4BSD-based operating systems. It incorporates lessons learned in the BSD world and elsewhere about multi-architecture support and the demands of heterogeneous networking.

Although this standard is more comprehensive than previous attempts at filesystem hierarchy standardization, periodic updates may become necessary as requirements change in relation to emerging technology. It is also possible that better solutions to the problems addressed here will be discovered so that our solutions will no longer be the best possible solutions. Supplementary drafts may be released in addition to periodic updates to this document. However, a specific goal is backwards compatibility from one release of this document to the next.

Comments related to this standard are welcome. Any comments or suggestions for changes may be directed to the FHS editor (Daniel Quinlan <quinlan@pathname.com>) or the FHS mailing list. Typographical or grammatical comments should be directed to the FHS editor.

Before sending mail to the mailing list it is requested that you first contact the FHS editor in order to avoid excessive re-discussion of old topics.

Questions about how to interpret items in this document may occasionally arise. If you have need for a clarification, please contact the FHS editor. Since this standard represents a consensus of many participants, it is important to make certain that any interpretation also represents their collective opinion. For this reason it may not be possible to provide an immediate response unless the inquiry has been the subject of previous discussion.

## 7.5. Acknowledgments

The developers of the FHS wish to thank the developers, system administrators, and users whose input was essential to this standard. We wish to thank each of the contributors who helped to write, compile, and compose this standard.

The FHS Group also wishes to thank those Linux developers who supported the FSSTND, the predecessor to this standard. If they hadn’t demonstrated that the FSSTND was beneficial, the FHS could never have evolved.

## 7.6. Contributors

| | |
|---|---|
| Brandon S. Allbery | <bsa@kf8nh.wariat.org> |
| Keith Bostic | <bostic@cs.berkeley.edu> |
| Drew Eckhardt | <drew@colorado.edu> |
| Rik Faith | <faith@cs.unc.edu> |
| Stephen Harris | <sweh@spuddy.mew.co.uk> |
| Ian Jackson | <ijackson@cus.cam.ac.uk> |
| Andreas Jaeger | <aj@suse.de> |
| John A. Martin | <jmartin@acm.org> |
| Ian McCloghrie | <ian@ucsd.edu> |
| Chris Metcalf | <metcalf@lcs.mit.edu> |
| Ian Murdock | <imurdock@debian.org> |
| David C. Niemi | <niemidc@clark.net> |
| Daniel Quinlan | <quinlan@pathname.com> |
| Eric S. Raymond | <esr@thyrsus.com> |
| Rusty Russell | <rusty@rustcorp.com.au> |

| | |
|---|---|
| Mike Sangrey | <mike@sojurn.lns.pa.us> |
| David H. Silber | <dhs@glowworm.firefly.com> |
| Thomas Sippel-Dau | <t.sippel-dau@ic.ac.uk> |
| Theodore Ts’o | <tytso@athena.mit.edu> |
| Stephen Tweedie | <sct@dcs.ed.ac.uk> |
| Fred N. van Kempen | <waltje@infomagic.com> |
| Bernd Warken | <bwarken@mayn.de> |
| Christopher Yeoh | <cyeoh@samba.org> |

Chapitre 2

# *Préprocesseur C*

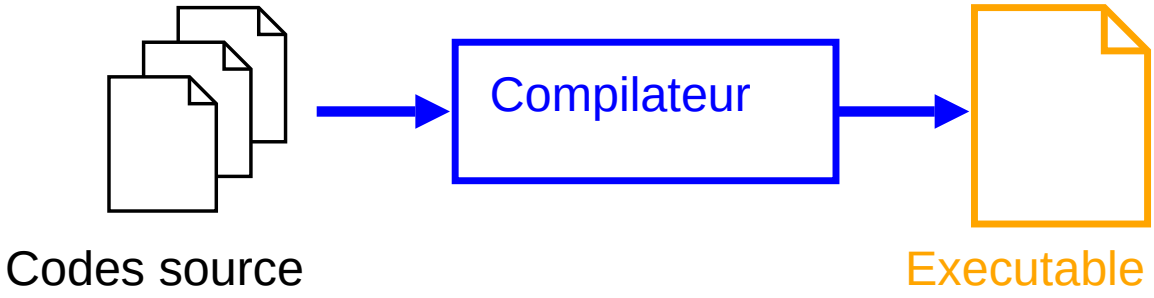




Voir le cours de programmation en langage C pour plus de détails.

On utilisera souvent le sigle CPP pour « C Pre Processor » dans ce cours (et non pas pour « C Plus Plus » plutôt codé « C++ »).

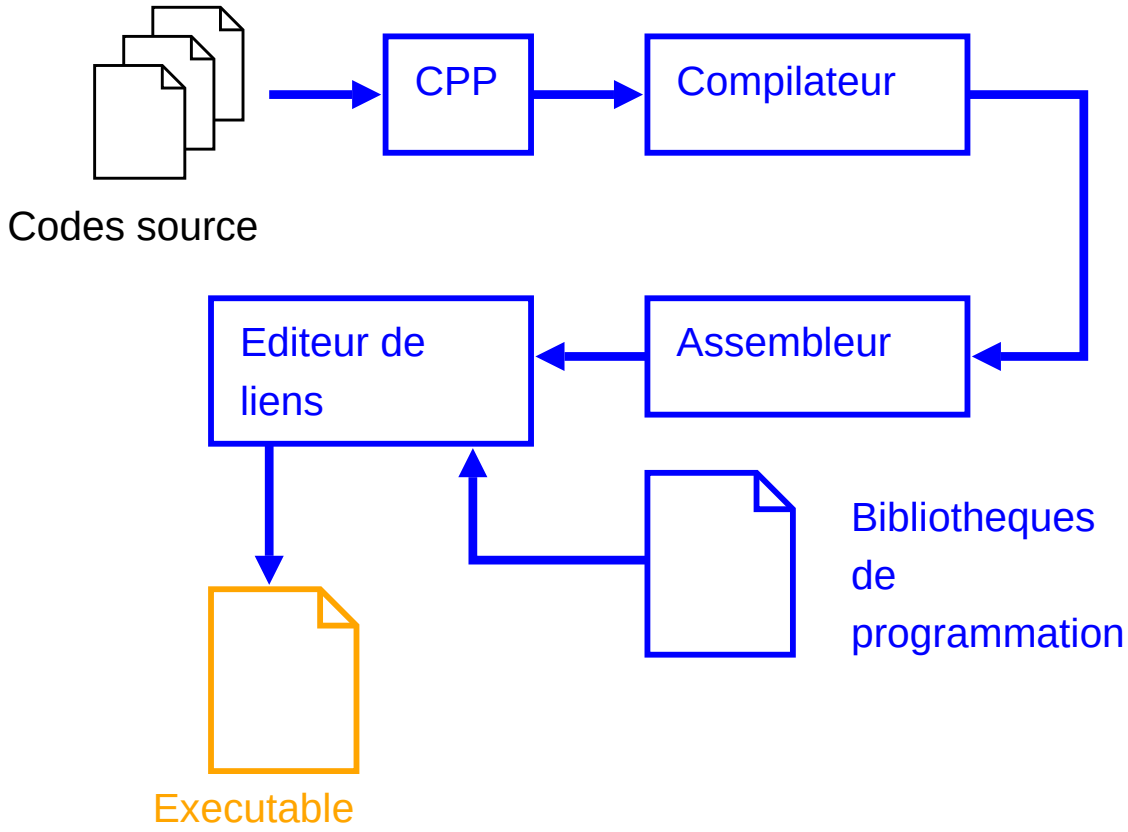
En apparence :








En réalité :



Sur les machines UNIX traditionnelles, le compilateur C est la commande « cc ». Caractéristiques :

- compilateurs spécifiques d'une machine
- compilateurs souvent performants
- compilateurs souvent payants

Sur les distributions LINUX, le compilateur C est la commande « gcc ». Caractéristiques :

- compilateur universel
- compilateur considéré comme peu performant
- problèmes lors des changements de versions (1.x à 2.x, 3.x à 4.x)
- compilateur gratuit
- compile les langages C, C++, Objective-C, Java, Fortran, etc.

Dans ce cours, on parlera indifféremment de « cc » ou « gcc ».





Le préprocesseur C (CPP) intervient tout au début de la chaîne de compilation.

Il existe deux méthodes de l'appeler :

1. commande du type :
```
% /usr/lib/cpp prog.c
```
2. le plus pratique (mais cela ne marche que si le fichier a une extension «.c») :
```
% cc -E prog.c
```

CPP a pour utilités :

- de fournir des macros dans le code C
- de fournir les descriptions de structures système, les prototypes des fonctions système
- de décider du déroulement du code C selon certains critères de compilation

Consultons la page de manuel de la fonction C « stat » :

```
STAT(2)                    FreeBSD System Calls Manual

NAME
     stat, lstat, fstat - get file status

SYNOPSIS
     #include <sys/types.h>
     #include <sys/stat.h>
...
     struct stat {
         dev_t     st_dev;     /* inode's device */
         ino_t     st_ino;     /* inode's number */
         mode_t    st_mode;    /* inode protection mode */
         nlink_t   st_nlink;   /* number of hard links */
         uid_t     st_uid;     /* user ID of the file's owner */
...
```





```
% cat prog.c
#include <sys/stat.h>
/* fin du fichier */

% cc -E prog.c
# 1 "prog.c"
# 1 "/usr/include/sys/stat.h" 1 3
...
# 96 "/usr/include/sys/types.h" 2 3

typedef unsigned long   clock_t;
...
# 51 "/usr/include/sys/stat.h" 2 3

struct stat {
        dev_t     st_dev;
        ino_t     st_ino;
        mode_t    st_mode;
        nlink_t   st_nlink;
        uid_t     st_uid;
...
```

Au passage, on notera les lignes du type :

```
# 96 "/usr/include/sys/types.h" 2 3
```

La commande « `cpp -P` » permet de supprimer ces lignes lors de l'appel direct au préprocesseur.





Il existe deux applications possibles de la directive « `#include` » :

- « `#include<...>` » : On cherche le fichier spécifié entre <> dans la liste des répertoires système : `/usr/include`, `/usr/local/include` ...
- « `#include "..."` » : On cherche le fichier spécifié entre "" relativement au directory courant de compilation.

Par exemple :

```
#include "hack.h"
#include <signal.h>
```

La recherche des fichiers include peut être modifiée en spécifiant de nouveaux répertoires à sonder. La syntaxe est :

```
% cc -Irépertoire prog.c
```

Pour porter un programme sur une nouvelle architecture, on utilisera parfois « `-I` ».





On peut préciser l'existence d'un symbole, la non existence d'un symbole via les directives « `#define` » ou « `#undef` » :

```
#define DEBUG
#define TAILLE 32
#undef  VERBOSE
```

On peut aussi préciser l'existence d'un symbole, la non existence d'un symbole via la ligne de commande via les options « -D » ou « -U » :

En C :

```
#define DEBUG
#define TAILLE 32
#undef  VERBOSE
```

Via la ligne de commande :

```
% cc -DDEBUG prog.c
% cc -DTAILLE=32 prog.c
% cc -UVERBOSE prog.c
```





Voir le cours de C pour les macros en général.

Rappel : exemple d'une macro codant une division :

```
#define DIV(x,y)  x / y
```

Cette macro est cependant **fausse**.

« `DIV(a+1, b)` » donne littéralement « `a + 1 / b` »
ou encore « $a + \frac{1}{b}$ » ce qui n'est pas « $\frac{a+1}{b}$ ».

Façon correcte de coder la macro :

```
#define DIV(x,y)  (x) / (y)
```

```
#ifdef symbole
...
#else
...
#endif
```

```
#if defined(symbole)
...
#else
...
#endif
```





Par exemple :

```
#ifdef CHDIR
static void chdirx();
#endif /* CHDIR */
```

Par exemple :

```
#if defined(__sun)
extern struct passwd *FDECL(getpwuid,(uid_t));
#else
extern struct passwd *FDECL(getpwuid,(int));
#endif
```

```
#ifndef symbole
...
#else
...
#endif
```

```
#if ! defined(symbole)
...
#else
...
#endif
```

Par exemple :

```
#ifndef O_RDONLY
#include <fcntl.h>
#endif
```

```
#if !defined(O_RDONLY)
#include <fcntl.h>
#endif
```





On peut tester la valeur d'un symbole au moyen des opérateurs classiques du C : « == », « != », etc. :

```
#if symbole operateur valeur
...
#else
...
#endif
```

Par exemple :

```
#if STACK_DIRECTION == 0
  if (STACK_DIR == 0) /* Unknown growth direction. */
    find_stack_direction ();
#endif
```

On peut composer des tests sur les symboles par les opérateurs :

- « `&&` » (ET),
- « `||` » (OU),
- « `(` » « `)` » (regroupement),
- « `!` » (négation) :

Par exemple :

```
#if !defined(_ANSI_SOURCE) && !defined(__STRICT_ANSI__)
#include <sys/types.h>
#endif
```





Extrait de `<stdio.h>` :

```
#ifndef _STDIO_H_
#define _STDIO_H_
...
#include <sys/cdefs.h>
...
#endif /* _STDIO_H_ */
```

Extrait de `<sys/cdefs.h>` :

```
#ifndef _SYS_CDEFS_H_
#define _SYS_CDEFS_H_

...

#endif /* _SYS_CDEFS_H_ */
```

Explication : permet l'inclusion multiple sans générer d'erreur de multi-déclarations :

Par exemple :

```
#include <stdio.h>
#include <sys/cdefs.h>
```

L'utilisation de l'option « -v » du compilateur montre les symboles définis lors de l'appel de CPP.

Voir les exemples dans les pages suivantes.





◇ Linux

```
% cc -v prog.c
Reading specs from /usr/lib/gcc-lib/i386-linux/2.7.2.1/specs
gcc version 2.7.2.1
 /usr/lib/gcc-lib/i386-linux/2.7.2.1/cpp -lang-c -v -undef
-D__GNUC__=2 -D__GNUC_MINOR__=7 -D__ELF__ -Dunix -Di386 -Dlinux
-D__ELF__ -D__unix__ -D__i386__ -D__linux__ -D__unix -D__i386
-D__linux -Asystem(unix) -Asystem(posix) -Acpu(i386)
-Amachine(i386) prog.c /tmp/cca29785.i
GNU CPP version 2.7.2.1 (i386 Linux/ELF)
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /usr/i386-linux/include
 /usr/lib/gcc-lib/i386-linux/2.7.2.1/include
 /usr/include
End of search list.
...
```

## ◇ AIX 4

(compilateur C payant)

```
% cc -v prog.c
exec: /usr/lpp/xlC/exe/xlcentry(xlcentry,-D_AIX,-D_AIX32,
-D_AIX41,-D_IBMR2,-D_POWER,-qtbtable=full,-qlanglvl=extended,
-qnoro,-qnoroconst,-oprog.o,prog.c,/tmp/xlcrdt2ia,/tmp/xlcrdt2ib,
/dev/null,prog.lst,prog,/tmp/xlcrdt2ic,NULL)
...
```





## ◇ Digital UNIX 4

```
% cc -v prog.c
/usr/lib/cmplrs/cc/gemc_cc -D__LANGUAGE_C__ -D__unix__
-D__osf__ -D__alpha -D_SYSTYPE_BSD -D_LONGLONG -DLANGUAGE_C
-Dunix -DSYSTYPE_BSD -I/usr/include -preempt_symbol -g0
-O2 -std0 -o prog.o prog.c
/usr/lib/cmplrs/cc/gemc_cc:
0.00u 0.04s 0:02 1% 0+7k 467+4io 39pf+0w 7stk+920mem
...
```

## ◇ HP-UX 9.07

```
% cc -v prog.c
cc: CCOPTS is not set.
/lib/cpp prog.c /tmp/ctmAAAa19847 -$ -D__hp9000s700
-D__hp9000s800 -D__hppa -D__hpux -D__unix -D_PA_RISC1_1 -j
cc: Entering Preprocessor.
...
```





## ◇ Solaris 2.5

(compilateur C payant)

```
% /opt/SUNWspro/bin/cc -# prog.c
/opt/SUNWspro/bin/../SC4.2/bin/acomp -i prog.c -y-fbe
-y/opt/SUNWspro/bin/../SC4.2/bin/fbe -y-xarch=v7 -y-o -yprog.o
-y-s -y-verbose -Qy -D__SunOS_5_5 -D__SUNPRO_C=0x420
-D__SVR4 -D__unix -D__sun -D__sparc -D__BUILTIN_VA_ARG_INCR
-Xa -Dunix -Dsun -Dsparc -D__RESTRICT
-I/opt/SUNWspro/SC4.2/include/cc -g/opt/SUNWspro/bin/../SC4.2/bin/
...
```

## ◇ SunOS

```
% cc -v prog.c
/lib/cpp -undef -Dunix -Dsun -Dsparc prog.c
>/tmp/cpp.00308.0.i
...
```

Chapitre 3

# Compilation automatisée





Si par exemple les fichiers `source1.c`, `source2.c` et `source3.c` constituent un programme, il faut passer (par exemple) par les étapes suivantes pour arriver à l'exécutable :

1. `% gcc -c source1.c`
2. `% gcc -c source2.c`
3. `% gcc -c source3.c`
4. `% gcc source1.o source2.o source3.o -o prog.exe -lm`

En cas de modification de `source1.c`, il faut exécuter à nouveau les étapes 1 et 4.

Dans le cas d'un exécutable avec de nombreux fichiers sources, c'est vite lassant !

Le rôle de l'utilitaire « `make` » est d'automatiser la compilation une fois qu'on lui a décrit la logique de compilation du programme via le fichier « `Makefile` ».

L'emploi de « `make` » permettra de ne plus refaire les étapes 1 et 4. On lancera simplement « `make` » qui déclenchera ce qu'il faut.





Fichier « `Makefile` » de l'exemple :

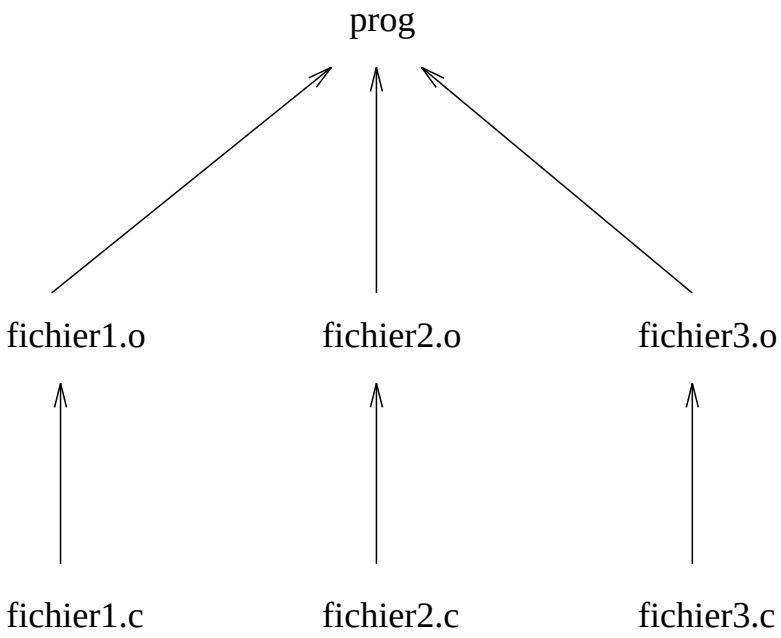
```
prog.exe: fichier1.o fichier2.o fichier3.o
        gcc -o prog.exe fichier1.o fichier2.o fichier3.o -lm

fichier1.o: fichier1.c
        gcc -c fichier1.c -o fichier1.o

fichier2.o: fichier2.c
        gcc -c fichier2.c -o fichier2.o

fichier3.o: fichier3.c
        gcc -c fichier3.c -o fichier3.o
```

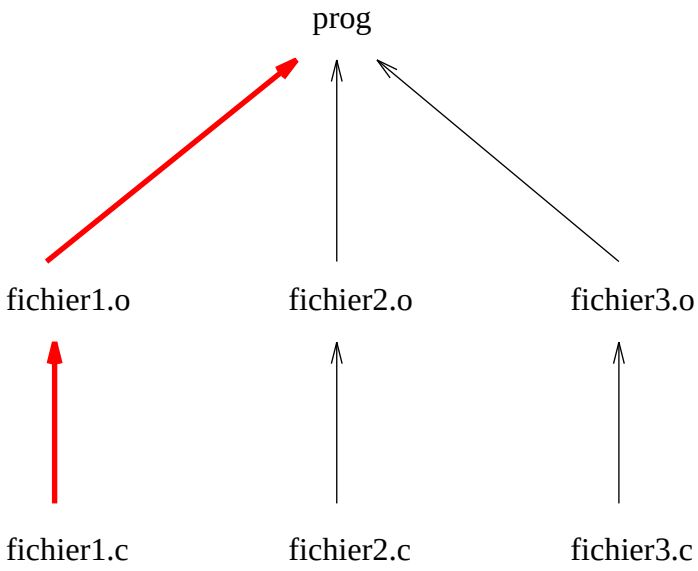
Les dépendances du fichier « `Makefile` » peuvent être traduites sous forme d'un arbre :








La modification de « `fichier1.c` » se traduit par une date de modification plus récente que la date du module « `fichier1.o` » et la date de l'exécutable « `prog.exe` ».




On recompile donc.

Syntaxe de la commande : `make [options] [cibles]`

Le fichier associé est « `Makefile` » ou « `makefile` ».

Quelques options intéressantes :

- option « `-n` » : affiche ce qui serait réalisé sans exécuter les commandes
- option « `-f fichier` » : utilise le nom de fichier mentionné plutôt que celui de « `Makefile` » ou « `makefile` »
- options « `-p` » : affiche la liste des variables ainsi que des règles de compilation prédéfinies

GNU make : `ftp://ftp.lip6.fr/pub/gnu/make/`





Syntaxe (partielle) du fichier :

```
cible: dépendances
\t commande à exécuter
```

Le «`\t`» est à remplacer par une tabulation. **Il faut respecter cette syntaxe.**

Le fichier « `Makefile` » peut utiliser des variables plutôt que de tout expliciter partout.
Forme : `VARIABLE = VALEUR`
Il y a également des noms de cibles prédéfinis ou conventionnels.
Par exemple : « `all` », « `install` », « `clean` »

En fait, MAKE dispose de tout un tas d'options et de comportements de compilation prédéfinis.
Cf « `make -p` ».

Par exemple :

```
% gmake -p
...
%.sh:

%: %.sh
        cat $< >$@
        chmod a+x $@
...
```

```
% ls
prog.sh
% make prog
cat prog.sh >prog
chmod a+x prog
% ls
prog prog.sh
% make prog
'prog' is up to date.
```





Exemple :

```
CC = gcc
CFLAGS = -O

all : prog.exe

prog.exe: fichier1.o fichier2.o fichier3.o
        $(CC) $(CFLAGS) -o prog.exe fichier1.o fichier2.o
        fichier3.o -lm

fichier1.o: fichier1.c
        $(CC) $(CFLAGS) -c fichier1.c -o fichier1.o

fichier2.o: fichier2.c
        $(CC) $(CFLAGS) -c fichier2.c -o fichier2.o

fichier3.o: fichier3.c
        $(CC) $(CFLAGS) -c fichier3.c -o fichier3.o

clean:
        rm -f *.o prog.exe core *~
```

L'exemple précédent peut être réécrit en utilisant les macros de MAKE et les règles de compilation par défaut :

```
CC = gcc
CFLAGS = -O
LIBS = -lm

all: prog.exe

prog.exe: fichier1.o fichier2.o fichier3.o
        $(CC) -o $@ $^ $(LIBS)

clean:
        rm -f *.o prog.exe core
```





ce qui donne :

```
% gmake clean
rm -f *.o prog.exe core
% gmake all
gcc -O   -c -o fichier1.o fichier1.c
gcc -O   -c -o fichier2.o fichier2.c
gcc -O   -c -o fichier3.o fichier3.c
gcc -o prog.exe fichier1.o fichier2.o fichier3.o -lm
```

L'utilitaire « `make` » peut être utilisé à autre chose que de faire de la compilation : rien ne fait mention à l'obligation de n'utiliser que les commandes de compilation au niveau du fichier `Makefile`. Par exemple :

```
all: cours.dvi

SRC = \
        cours.tex \
        introduction.tex \
        chapitre1.tex \
        chapitre2.tex \
        macros.tex

cours.dvi: $(SRC)
        latex cours.tex
```

Chapitre 4

# *Logiciels en sources versus logiciels en packages*





Deux philosophies :

- installer des logiciels en en compilant les sources
- récupérer les logiciels sous forme déjà compilée et les installer tels que ; on parle de *packages* (paquetages)

La tendance est de plus en plus à la seconde philosophie suite à l'apparition des distributions grand public de LINUX.
Les puristes préféreront compiler les sources.

Avantages de compiler les sources d'un logiciel :

- totale maitrise de la configuration du logiciel
- possibilité d'adapter le logiciel à une version légérement différente d'un logiciel associé

Inconvénients de compiler les sources d'un logiciel :

- nécessité de maitriser la compilation de logiciels (cf chapitres à venir)
- difficulté des mises à jour des logiciels compilés
- difficulté de la désinstallation des logiciels compilés





Avantages des packages :

- installation relativement aisée même pour un néophyte
- gestion des dépendances entre packages
- rapidité de disponibilité des packages via les sociétés distributrices de LINUX
- couplage des RPM avec d'autres logiciels (par exemple logiciels d'inventaire comme OCS-NG)

Inconvénients des packages :

- impossibilité d'installer deux versions du même logiciel
- un package d'une distribution LINUX A ne fonctionne pas sur une distribution LINUX B
- confiance à donner à des binaires compilés par d'autres personnes ; chevaux de Troie ?
- options de compilation adaptées à votre système ?
- dépendance entre packages parfois trop fortes ; la version 3.2 du logiciel XYZ fonctionne comme la 3.1 mais le package que l'on installe n'en démord pas et veut la 3.1 !
- trop grande généralité de certains packages ; par exemple un package d'impression va installer un traitement de textes



# Chapitre 4 • Logiciels en sources versus logiciels en packages

## §4.2 • Packages Red Hat, `rpm`

C'est le format des logiciels livrés par la version commerciale Red Hat de LINUX (mais le programme `rpm` est disponible sur d'autres UNIX, par exemple sur Solaris).

On trouvera de la documentation sur ce format à ces quelques endroits :

- Certainement pas dans la page de manuel qui est incompréhensible
- sur le site `http://www.rpm.org`
- dans le livre *Maximum RPM* disponible sous forme postscript à l'URL `http://www.rpm.org`

Un logiciel au format RPM se présente sous la forme d'un fichier dont le nom suit la syntaxe :

```
nom du package-version du package-version RPM-architecture du processeur.rpm
```

comme dans « `rpm-3.0.2-6.0.src.rpm` ».

Voici un tableau mentionnant quelques options utiles :

| Commande | Fonction |
|---|---|
| `rpm -ivh package.rpm` | Install `package.rpm`, be verbose, show hash marks |
| `rpm -Uvh package.rpm` | Upgrade `package.rpm`, be verbose, show hash marks |
| `rpm -qf fichier` | Check which package owns a file |
| `rpm -qpi package.rpm` | Queries `package.rpm`, lists info |
| `rpm -qpl package.rpm` | Queries `package.rpm`, lists all files |
| `rpm -qa` | Queries RPM database lists all packages installed |
| `rpm -e package-name` | Removes `packag-name` from the system (as listed by `rpm -qa`) |





◇ Installation, mise à jour et rafraichissement

Syntaxes :

- « **`rpm -i package.rpm`** »
- « **`rpm -U package.rpm`** »
- « **`rpm -F package.rpm`** »

| | package non installé | package déjà installé |
|---|---|---|
| « `-i` » | installation du package | rien |
| « `-U` » | installation du package | mise à jour du package |
| « `-F` » | rien | mise à jour du package |

Les fichiers de configuration déjà présents sont conservés lors d'une mise à jour de package. Les nouveaux fichiers de configuration qui devraient être installés sont mis en place avec l'extension « `.rpmnew` ».

◇ Désinstallation

Syntaxes :

- « **rpm -e package** »





◇ Interrogation de la base RPM

Syntaxes :

- « **rpm -q -a** » : liste des packages installés
- « **rpm -q -l package** » : liste des fichiers installés par le package
- « **rpm -q -c package** » : liste les fichiers de configuration du package installé
- « **rpm -q -f /chemin/vers/objet** » : renvoie le nom du package ayant installé cet objet
- « **rpm -q --scripts package** » : liste les scripts de pré-installation et de post-installation du package installé

Option « `-p package.rpm` » : agit sur le package indiqué sans qu'il soit installé

### ◇ Vérification d'un package

La base RPM contient diverses informations :

- checksum MD5 des composants du package
- droits des composants du package
- propriétaires des composants du package
- groupes des composants du package

Syntaxes :

- « **`rpm -V -a`** » : vérifie tous les packages installés
- « **`rpm -V -f /chemin/vers/objet`** » : vérifie l'objet installé





### ◇ Autres opérations

Syntaxes :

- « **`rpm -q -a --last`** » : liste les packages installés par date d'installation
- « **`rpm -rebuildb`** » : reconstruit la base RPM

◇ Commande « yum »

YUM = Yellow dog Updater Modified
Version améliorée de RPM.
Syntaxes :

- « **yum install package** »
- « **yum remove package** »
- « **yum update package** »
- « **yum check-update** »
- « **yum list all** »
- « **yum list available** »
- « **yum list updates** »
- « **yum list installed** »
- « **yum provides /chemin/vers/objet** »
- « **yum search toto** »





**A completer...**

Chapitre 5

# GNU configure





`ftp://ftp.lip6.fr/pub/gnu/configure`

GNU configure : logiciel « d'assistance à la compilation de programme »

En vogue chez plus en plus de programmeurs car simplification de la vie pour tout le monde (programmeur comme utilisateur final qui compilera) :

- plus besoin de connaître les détails de l'API C de l'UNIX
- plus besoin de savoir modifier un `Makefile`
- plus besoin de savoir utiliser les options `-I`, `-L`, `-l`
- plus besoin de savoir patcher des sources C avec `#ifdef`, `#define`

La documentation du logiciel à compiler dira de faire appel à « configure ».

Puis comment utiliser « configure » ?

1. « ./configure --help » puis regarder ce qui est affiché pour en déduire les options disponibles.
2. lancer « configure » avec les réglages souhaités
3. compiler par « make »

Fichiers liés à configure repérables :

```
% ls
Changes                 README.win32            lib
Fonts                   ac-tools                parse_afm
LGPL                    configure               type1afm
LICENSE                 configure.com           win32
Makefile.in             configure.in            xglyph
README.t1lib-1.3.1      doc
README.t1python         examples
```





Exemple de sortie de l'aide :

```
% ./configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE
  --help                  print this message
  --no-create             do not create output files
  --quiet, --silent       do not print `checking...' messages
  --version               print the version of autoconf that created conf
Directory and file names:
  --prefix=PREFIX         install architecture-independent files in PREFI
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [same as prefix]
  --bindir=DIR            user executables in DIR [EPREFIX/bin]
  --sbindir=DIR           system admin executables in DIR [EPREFIX/sbin]
  --libexecdir=DIR        program executables in DIR [EPREFIX/libexec]
```

```
  --datadir=DIR           read-only architecture-independent data in DIR
                          [PREFIX/share]
  --sysconfdir=DIR        read-only single-machine data in DIR [PREFIX/et
  --sharedstatedir=DIR    modifiable architecture-independent data in DIR
                          [PREFIX/com]
  --localstatedir=DIR     modifiable single-machine data in DIR [PREFIX/v
  --libdir=DIR            object code libraries in DIR [EPREFIX/lib]
  --includedir=DIR        C header files in DIR [PREFIX/include]
  --oldincludedir=DIR     C header files for non-gcc in DIR [/usr/include
  --infodir=DIR           info documentation in DIR [PREFIX/info]
  --mandir=DIR            man documentation in DIR [PREFIX/man]
  --srcdir=DIR            find the sources in DIR [configure dir or ..]
  --program-prefix=PREFIX prepend PREFIX to installed program names
  --program-suffix=SUFFIX append SUFFIX to installed program names
  --program-transform-name=PROGRAM
                          run sed PROGRAM on installed program names
Host type:
  --build=BUILD           configure for building on BUILD [BUILD=HOST]
  --host=HOST             configure for HOST [guessed]
  --target=TARGET         configure for TARGET [TARGET=HOST]
```





```
Features and packages:
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATUR
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --x-includes=DIR        X include files are in DIR
  --x-libraries=DIR       X library files are in DIR
--enable and --with options recognized:
  --with-x                use the X Window System
  --without-athena        do not use Athena widgets
  --with-athena-includes=DIR   Athena include files are in DIR
  --with-athena-libraries=DIR  Athena libraries are in DIR
  --enable-shared[=PKGS]  build shared libraries [default=yes]
  --enable-static[=PKGS]  build static libraries [default=yes]
  --enable-fast-install[=PKGS]  optimize for fast installation [default=y
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --disable-libtool-lock  avoid locking (might break parallel builds)
```

## Exemple de sortie de l'exécution de configure :

```
% ./configure --prefix=/tmp/toto
creating cache ./config.cache
checking whether make sets $MAKE... yes
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking for X... libraries /usr/X11R6/lib, headers /usr/X11R6/include
checking for dnet_ntoa in -ldnet... no
checking for dnet_ntoa in -ldnet_stub... no
checking for gethostbyname... yes
checking for connect... yes
checking for remove... yes
checking for shmat... yes
checking for IceConnectionNumber in -lICE... yes
```





```
checking for Athena... libraries in default path, headers in default path
checking host system type... i386-unknown-freebsd4.6.1
checking build system type... i386-unknown-freebsd4.6.1
checking for ranlib... ranlib
checking for ld used by GCC... /usr/libexec/elf/ld
checking if the linker (/usr/libexec/elf/ld) is GNU ld... yes
checking for BSD-compatible nm... /usr/bin/nm -B
checking whether ln -s works... yes
updating cache ./config.cache
loading cache ./config.cache within ltconfig
checking for object suffix... o
checking for executable suffix... no
checking for gcc option to produce PIC... -fPIC
checking if gcc PIC flag -fPIC works... yes
checking if gcc supports -c -o file.o... yes
checking if gcc supports -c -o file.lo... yes
checking if gcc supports -fno-rtti -fno-exceptions ... yes
checking if gcc static flag -static works... -static
checking if the linker (/usr/libexec/elf/ld) is GNU ld... yes
checking whether the linker (/usr/libexec/elf/ld) supports shared librari
checking command to parse /usr/bin/nm -B output... ok
```

```
checking how to hardcode library paths into programs... immediate
checking for /usr/libexec/elf/ld option to reload object files... -r
checking dynamic linker characteristics... freebsd4.6.1 ld.so
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... yes
checking for objdir... .libs
creating libtool
updating cache ./config.cache
loading cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether char is unsigned... no
checking size of char... 1
checking size of short... 2
checking size of int... 4
checking size of long... 4
checking size of long long... 8
checking size of float... 4
checking size of double... 8
checking size of long double... 12
checking size of void *... 4
```





```
checking for gcc strength-reduce bug... no
checking whether external symbols need an underscore prefix... no
checking which ANSI integer type is 16 bit... short
checking which ANSI integer type is 32 bit... int
checking which ANSI integer type is 64 bit... <none>
checking whether stat file-mode macros are broken... no
checking for working const... yes
checking for ANSI C header files... yes
checking for size_t... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating lib/Makefile
creating lib/type1/Makefile
creating lib/t1lib/Makefile
creating xglyph/Makefile
creating type1afm/Makefile
creating examples/Makefile
creating doc/Makefile
creating lib/t1lib/sysconf.h
lib/t1lib/sysconf.h is unchanged
```

```
Configure finished!
        Do:  'make'                to compile the t1lib-package.
        Do:  'make without_doc'    to compile if you do not have LaTeX.
        Do:  'make install'        to install the t1lib-package.
```





On notera l'apparition de nouveaux fichiers :

```
% ls
Changes                 README.win32            doc
Fonts                   ac-tools                examples
LGPL                    config.cache            lib
LICENSE                 config.log              libtool
Makefile                config.status           parse_afm
Makefile.in             configure               type1afm
README.t1lib-1.3.1      configure.com           win32
README.t1python         configure.in            xglyph
```

Chapitre 6

# Gestion des comptes utilisateurs





◇ Humour :



◇ Humour :




"Yesterday I changed everyone's password to 'password'. I sent it to everyone in a memo, put it on a big sign on the wall and printed it on all of the coffee cups. Guess how many people called me this morning because they forgot the password."





Un compte utilisateur :

- définition de son identification pour le système
- mais aussi définition d'un environnement utilisateur :
  - home directory
  - shell
  - fichiers d'initialisation du shell
  - environnement graphique
  - quotas
  - etc.

Opérations courantes sur un compte d'utilisateur :

- création d'un compte
- modifications d'un compte
- destruction d'un compte
- archivage d'un compte

**Le fichier « /etc/passwd » contient les définitions des utilisateurs.**

Exemple :

```
root:U/oeCQo.cmnyc:0:1:Operator:/root:/bin/csh
nobody:*:65534:65534::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
besancon:vSHntkfqz8MEk:4332:1000::/users/adm/besancon:/bin/bash
...
```





Caractéristiques :

- le fichier « /etc/passwd » existe sur tous les UNIX !
- fichier texte ⇒ manipulable par les commandes UNIX classiques par exemple « vi »
- attention : ajouter un compte dans le fichier texte ne fait rien d'autre : **il faut compléter la création du compte pour qu'il soit fonctionnel**

Le format est composé de 7 champs séparés par le caractère « : » :

```
login : mot de passe : UID : GID : gecos : homedir : shell
```

Pour la définition exacte sur sa machine, se reporter à « <pwd.h> ».

Par exemple sur Linux :

```
/* The passwd structure.  */
struct passwd
{
  char *pw_name;                /* Username.  */
  char *pw_passwd;              /* Password.  */
  __uid_t pw_uid;               /* User ID.  */
  __gid_t pw_gid;               /* Group ID.  */
  char *pw_gecos;               /* Real name.  */
  char *pw_dir;                 /* Home directory.  */
  char *pw_shell;               /* Shell program.  */
};
```





- champ 1 : **login**
  Nom sous lequel un ordinateur connait un individu.
  **8 caractères en général**. Pour plus, vérifier
  Sur Linux, on peut avoir des noms plus longs que 8.
  Conseils : pas de majuscules, éviter les caractères accentués.
  Bref : lettres minuscules, chiffres et caractère « - », caractère « _ »

- champ 2 : **mot de passe**
  Mot de passe de l'utilisateur stocké **sous forme chiffrée non déchiffrable**.
  Nécessité d'éduquer les utilisateurs pour choisir un bon mot de passe.
  Les logiciels de crack de mots de passe ne déchiffrent pas les mots de passe : ils font des essais à base de dictionnaires.
  « crack » : ftp://ftp.lip6.fr/pub/unix/security/crack5.0.tar.gz
  « john » : ftp://ftp.false.com/pub/security/john/john-1.6.tar.gz

- champ 3 : **UID**
  Identificateur numérique compris entre 0 et 32767 (short int a priori).
  Sur Linux, c'est un « __uid_t » (équivalent à un « unsigned int »)
  Cette valeur doit être unique au sein des utilisateurs.

- champ 4 : **GID**
  Identificateur numérique compris entre 0 et 32767 (short int a priori).
  Sur Linux, c'est un « __gid_t » (équivalent à un « unsigned int »)
  Cette valeur doit être unique au sein des groupes.

```
% ls -l
-rw-r--r--   1 besancon software  5928 Sep 15 22:49 fichier1.txt
-rw-r--r--   1 besancon software   940 Sep 15 22:27 fichier2.txt

% ls -ln
-rw-r--r--   1 4332     1000      5544 Sep 15 22:47 fichier1.txt
-rw-r--r--   1 4332     1000       940 Sep 15 22:27 fichier2.txt
```





- champ 5 : **gecos**
  Identité en clair de l'utilisateur.
  Les systèmes BSD y ont stocké d'autres informations comme le numéro de téléphone, le numéro de bureau etc.
  La commande « **chfn** » (change finger) permet de modifier ce champs.
  GECOS = *General Electric Comprehensive Operating System*
  Champs ajouté par GE pour des machines UNIX de service aux Bell labs.
  Champs conservé au cours du temps.

- champ 6 : **homedir**
  Répertoire par défaut.

- champ 7 : **shell**
  Shell par défaut.
  Si le champ est vide, on prend « /bin/sh » par défaut.
  Le shell déterminera les fichiers de configuration à installer chez l'utilisateur.
  La commande « **chsh** » (change shell) permet de choisir parmi les shells mentionnés dans « /etc/shells ».





Quelques constatations :

- les UIDs ne sont pas nécessairement consécutifs ; il peut y avoir des trous dans la numérotation
- une ligne dans « /etc/passwd » n'implique pas que le homedirectory existe ; on peut avoir un utilisateur défini sans que son homedir existe
- des objets peuvent appartenir à un UID qui n'est pas utilisé dans « /etc/passwd »
- on peut avoir plusieurs utilisateurs définis ayant tous le même UID ; au niveau de la commande « ps », c'est le premier nom de login du fichier qui apparaitra.

Quelques conseils :

- Il faut toujours trier les lignes de « /etc/passwd » selon l'ordre numérique des UID :

```
# sort -t : +2n -3 /etc/passwd
```

- Il ne faut jamais laisser de comptes sans mot de passe.





Les mots de passe sont stockés sur une forme chiffrée, **mathématiquement non réversible**.

Ce chiffrement non réversible est appelé **hash** (le mot de passe chiffré est dit *hashed*).

Plusieurs algorithmes de chiffrement au cours du temps :

- **DES** : standard fédéral de l'administration américaine
- **MD5**

Le chiffrement DES est disponible sur UNIX via la fonction C « `crypt()` » :

- premier paramètre : la chaine à chiffrer
- second paramètre : la graine (dite *salt* en anglais) qui introduit une part d'aléatoire dans le chiffrement. Le salt est constitué de **deux caractères aléatoires**.
- résultat renvoyé : une chaine dont les premiers caractères reprennent le salt

Exemple de chiffrement :
« `crypt("QWERTY", "NU")` » renvoie « **`NUtw3cPnOYjcs`** »

Pourquoi retrouve-t-on le salt dans le résultat chiffré ?





Le chiffrement MD5 est disponible sur UNIX via la fonction C « `crypt()` ». Même fonction que pour DES mais forme différente des paramètres :

- premier paramètre : la chaine à chiffrer
- second paramètre : la graine (dite *salt* en anglais) qui introduit une part d'aléatoire dans le chiffrement. Le salt est constitué de **au plus huit caractères aléatoires écrits sous la forme « `$1$salt` »**
- résultat renvoyé : une chaine dont les premiers caractères reprennent le salt

Exemple de chiffrement MD5 :
« `crypt("QWERTY", "$1$cerisier")` » renvoie
« `$1$cerisier$XuezDDSxGjcKrvGFeUuV5.` »

Pourquoi retrouve-t-on le salt dans le résultat chiffré ?

Exemple de programme C complet (quick and dirty) :

```
#include<stdio.h>
main(int argc, char *argv[])
  {
    printf("=> %s\n", crypt(getpass("Chaine a chiffrer :"), argv[1]));
  }
```

Compiler sur LINUX par :

```
% gcc crypt.c -o crypt.exe -lcrypt
```

Une fois compilé, le programme montre :

```
% ./crypt.exe NU
Chaine a chiffrer : XXXXXXXX  <-- QWERTY pour l'exemple
=> NUtw3cPnOYjcs

% ./crypt.exe '$1$cerisier'
Chaine a chiffrer : XXXXXXXX  <-- QWERTY pour l'exemple
=> $1$cerisier$XuezDDSxGjcKrvGFeUuV5.
```





◇ Comportement lorsqu'un utilisateur change un mot de passe

```
% passwd
passwd:  Changing password for besancon
Enter login password: XXXXXXXX
New password: ZZZZZZZZ
Re-enter new password: ZZZZZZZZ
passwd (SYSTEM): passwd successfully changed for besancon
```

On vérifie l'identité de la personne en demandant le mot de passe initial.

◇ Comportement lorsque l'administrateur root change un mot de passe

```
# passwd besancon
New password: XXXXXXXX
Re-enter new password: XXXXXXXX
passwd (SYSTEM): passwd successfully changed for besancon
```

Root n'a jamais besoin de connaître le mot de passe d'un utilisateur pour en changer le mot de passe.

Lorsqu'un utilisateur « jean » a perdu son mot de passe :

- root ne peut pas déchiffrer la chaine chiffrée dans « /etc/passwd »
- root peut changer le mot de passe par « passwd jean »





La commande « passwd » peut comporter plus ou moins de règles de vérification de la sûreté du mot de passe :

```
% passwd besancon
Enter login password: XXXXXXXX
New password: XXXXXXXX   <-- on entre exprès le même mot de passe
passwd(SYSTEM): Passwords must differ by at least 3 positions
New password: YYYYYYYY   <-- mélange de lettres minuscules et majuscules
passwd(SYSTEM): The first 6 characters of the password
must contain at least two alphabetic characters and at least
one numeric or special character.
passwd(SYSTEM): Too many failures - try later.
Permission denied
```

La commande « `login` » est lancée par le système pour se connecter. Elle va devoir vérifier si les mots de passe entré et stocké sont identiques, sachant que le mot de passe est stocké sous une forme hashée.

Si le mot de passe hashé est « `saeLydiaFuF5o` », on réalisera donc au login la comparaison C :

```
strcmp("saeLydiaFuF5o", crypt("XXXXXXXX", "sa"))
```

avec « `XXXXXXXX` » le mot de passe en clair.





L'idée part de la constatation que, si beaucoup de programmes accèdent au contenu de « `/etc/passwd` » pour les informations concernant UID, homedir, shell, peu en revanche l'accèdent pour le mot de passe.

**On supprime donc le mot de passe chiffré du fichier « `/etc/passwd` » pour le stocker dans un fichier à accès plus restreint.**

**L'ancien mot de passe chiffré est remplacé par exemple par un caractère comme « `*` » ou « `x` » pour conserver la structure du fichier « `/etc/passwd` » initial de façon à être compatible avec d'anciennes applications.**

Plusieurs formats de fichiers shadow sont utilisés par les constructeurs qui ne se sont pas mis d'accord.

- documentation de la commande :
  ```
  % man passwd
  ```
- documentation des fichiers LINUX :
  ```
  % man -s 4 passwd
  % man -s 4 shadow
  ```
- documentation des fichiers SOLARIS :
  ```
  % man -s 4 passwd
  % man -s 4 shadow
  ```





### ◇ Description du format de shadow sur LINUX

- champ « `username` » : nom d'utilisateur
- champ « `password` » : mot de passe chiffré
- champ « `lastchg` » : date du dernier changement de mot de passe (compté en jours depuis le 1er janvier 1970)
- champ « `min` » : nombre minimal de jours entre deux changement de mot de passe
- champ « `max` » : nombre maximal de jours pendant lesquels le mot de passe est valide
- champ « `warn` » : nombre de jours avant l'expiration du mot de passe pour prévenir l'utilisateur
- champ « `inactive` » : nombre de jours d'inactivité autorisé
- champ « `expire` » : date d'expiration (compté en jours depuis le 1er janvier 1970)
- champ « `flag` » : nombre de logins échoués (sur les 4 bits de poids faible)

```
-rw-r--r--  root wheel 10557 Sep 15 22:51 /etc/passwd
-rw-------  root wheel 13318 Sep 15 22:50 /etc/shadow
```

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
...
```

```
# cat /etc/shadow
root:fUQZjKzpwayTc:11237:0:99999:7:-1:-1:134540316
bin:*:11237:0:99999:7:::
daemon:*:11237:0:99999:7:::
adm:*:11237:0:99999:7:::
...
```





### ◇ Description du format de shadow sur SOLARIS

`username:password:lastchg:min:max:warn:inactive:expire:flag`

- champ « `username` » : nom d'utilisateur
- champ « `password` » : mot de passe chiffré
- champ « `lastchg` » : date du dernier changement de mot de passe (compté en jours depuis le 1er janvier 1970)
- champ « `min` » : nombre minimal de jours entre deux changement de mot de passe
- champ « `max` » : nombre maximal de jours pendant lesquels le mot de passe est valide
- champ « `warn` » : nombre de jours avant l'expiration du mot de passe pour prévenir l'utilisateur
- champ « `inactive` » : nombre de jours d'inactivité autorisé
- champ « `expire` » : date d'expiration (compté en jours depuis le 1er janvier 1970)
- champ « `flag` » : nombre de logins échoués (sur les 4 bits de poids faible)

```
-rw-r--r--   1 root     sys         1032 Jul 11 23:21 /etc/passwd
-rw-------   1 root     sys          536 Jul 11 23:14 /etc/shadow
```

```
# cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
...
boumaiza:x:1001:1000:Lyazid Boumaiza:/net/serveur/home/ars/boumai
corsini:x:1002:1000:Franck Corsini:/net/serveur/home/ars/corsini:,
...
```

```
# cat /etc/shadow
root:y8fPbbq414TBU:6445::::::
daemon:NP:6445::::::
bin:NP:6445::::::
...
boumaiza:aalJCbuvX/xEM:11255::::::
corsini:5Ag56r/M.KD9A:10912::::::
...
```





### ◇ Hash « x »

Si dans « /etc/passwd », on trouve le mot de passe chiffré « x », alors on doit chercher le vrai mot de passe chiffré dans « /etc/shadow ».

### ◇ Hash « * »

Ce hash ne peut pas être le résultat d'un hashage DES ou MD5. En pratique il empêche un utilisateur de se connecter.

### ◇ Hash « !! »

Ce hash signifie que le mot de passe n'a pas encore été défini.

Le format est composé de 4 champs séparés par le caractère « **:** » :

```
group : mot de passe : GID : membres
```

Pour la définition exacte sur sa machine, se reporter à « <grp.h> ».

Par exemple, sur une machine Linux :

```
/* The group structure.  */
struct group
  {
    char *gr_name;              /* Group name.  */
    char *gr_passwd;            /* Password.    */
    __gid_t gr_gid;             /* Group ID.    */
    char **gr_mem;              /* Member list. */
  };
```





- champ 1 : **group**
  Nom du groupe apparaissant par « `ls -lg` ».

- champ 2 : **mot de passe**
  Mot de passe du groupe.
  **Je n'ai jamais vu autre chose que « x » ou « * », c'est-à-dire aucun mot de passe utilisable.**

- champ 3 : **GID**
  Identificateur numérique compris entre 0 et 32767.
  Sur Linux, c'est un « `__gid_t` » (équivalent à un « unsigned int »)
  Cette valeur doit être unique au sein des groupes.

- champ 4 : **membres**
  Liste de noms de login séparés par des virgules.

A savoir :

L'ajout dans un groupe n'est pas dynamique :

1. un utilisateur est ajouté dans un groupe dans « `/etc/group` »
2. déconnexion de l'utilisateur
3. reconnexion de l'utilisateur
4. l'utilisateur fait partie effectivement du groupe

Un groupe UNIX ne contient que des noms d'utilisateurs.
Un groupe UNIX ne contient pas d'autres groupes.

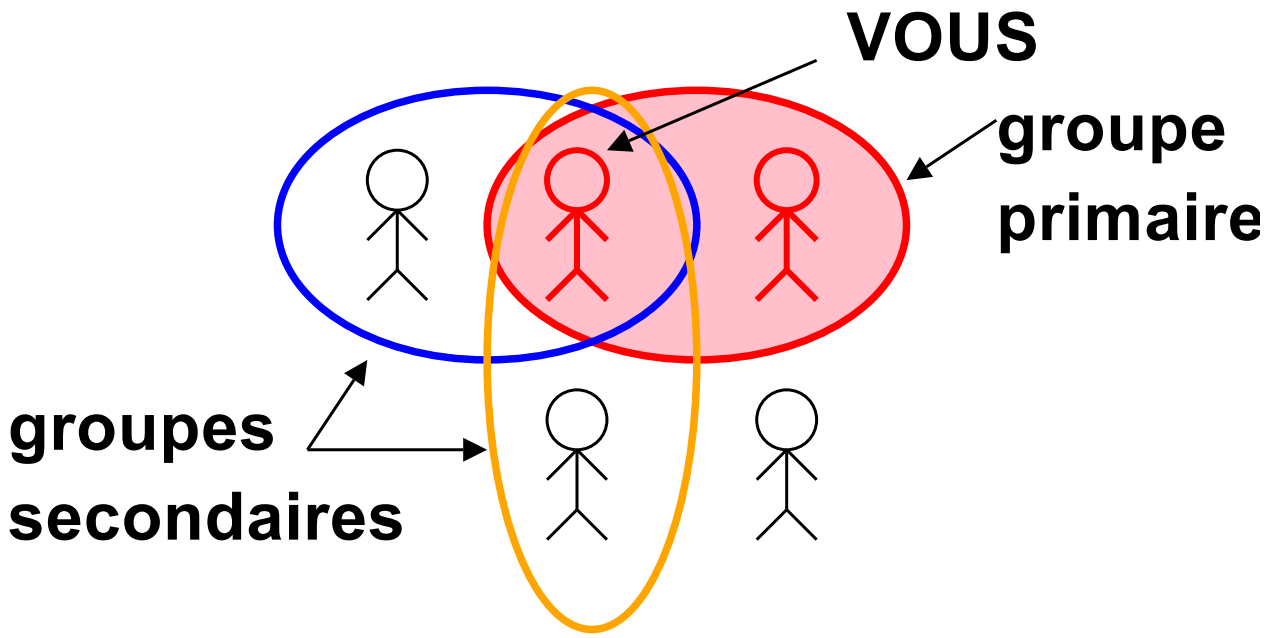
Un utilisateur peut appartenir à plusieurs groupes mais un bug dans NFS v2 limite en pratique le nombre de groupes à 8.



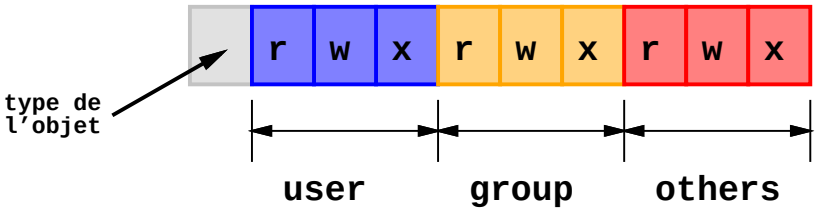


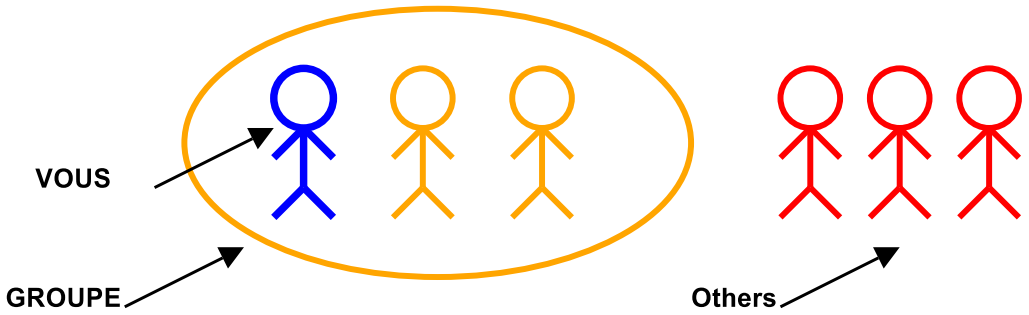
Principe des groupes :

- Un utilisateur appartient à un groupe principal
- Un utilisateur peut appartenir à d'autres groupes
- Un objet a un propriétaire utilisateur et un propriétaire groupe



Rappel sur les droits :

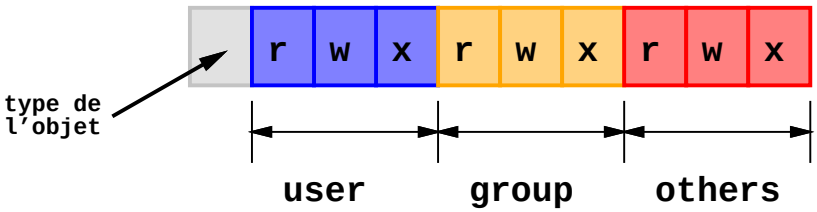


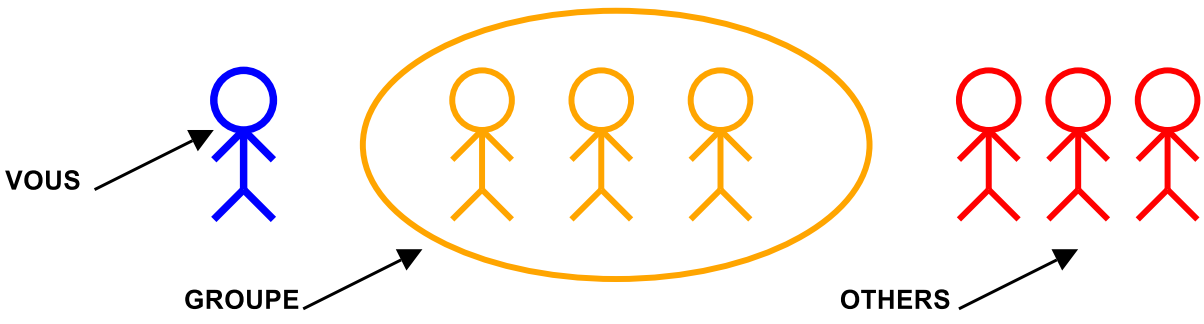








Rappel sur les droits :



Editer manuellement les fichiers « /etc/passwd » et « /etc/shadow » n'est pas conseillé.

On évite donc de faire « vi /etc/passwd /etc/shadow » pour modifier les fichiers.

Solution : commande « vipw »

Syntaxe : « vipw »

La syntaxe n'est pas : « vipw /etc/passwd » ni « vipw /etc/shadow ».





Principe de « vipw » :

- on fait une copie des fichiers « /etc/passwd » et « /etc/shadow » dans des fichiers temporaires
- on édite les fichiers temporaires avec « vi »
- pendant l'édition on positionne un verrou sur les fichiers « /etc/passwd » et « /etc/shadow » pour éviter une autre édition
- on termine l'édition sous «vi » par « :wq »
- la commande « vipw » vérifie (parfois) la syntaxe des fichiers temporaires
- on écrase les fichiers « /etc/passwd » et « /etc/shadow » par les fichiers temporaires
- on supprime le verrou

Démonstration du verrou : cas de deux administrateurs voulant modifier le fichier en même temps :

- Administrateur 1 :

```
# vipw
... ok ... cet administrateur peut modifier le fichier
```

- Administrateur 2 :

```
# vipw
vipw: /etc/passwd file busy
```





Actions schématiques à accomplir lors de la création d'un compte :

1. choix de l'UID et du GID en fonction du service d'appartenance de la personne
2. choix du homedir (en fonction du service ?)
3. choix du nom de login selon la politique locale
4. choix du shell de login
5. incorporation de ces informations dans la base de données des comptes ( « `/etc/passwd` » et « `/etc/shadow` » ) ; « `*` » comme mot de passe par défaut

6. ajout de l'utilisateur dans « `/etc/group` »
7. création du homedir
8. copie des fichiers de configuration de l'environnement (« `.profile` », « `.cshrc` », « `.xsession` » etc.)
9. attribuer le homedir créé à l'utilisateur par « **chown** » + « **chgrp** »
10. initialisation du mot de passe si l'utilisateur est présent

L'étape délicate dans une automatisation de création de compte est bien sûr l'incorporation d'une nouvelle ligne dans la base de données (accès concurrents, ne pas perdre le contenu du fichier, ...).

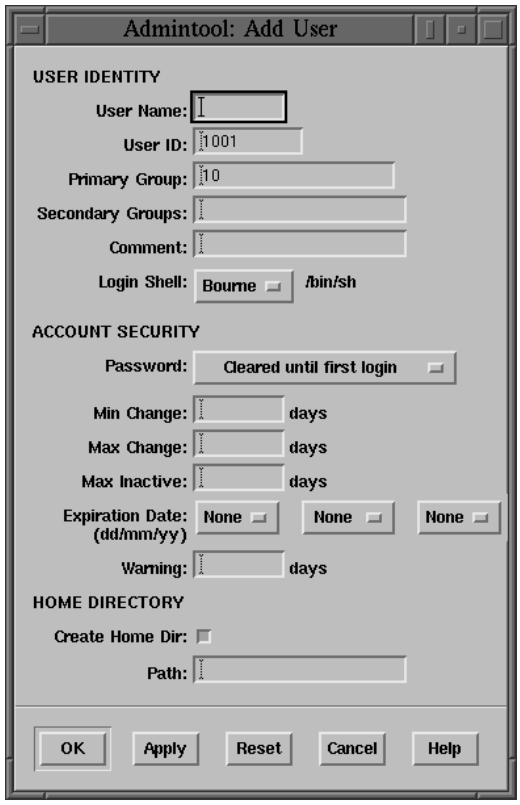




Il existe sur la plupart des systèmes des commandes automatisant le processus de création d'un compte :

| UNIX | Programme |
|---|---|
| AIX | « `smit` » |
| HP-UX | « `sam` » |
| Solaris | « `admintool` » |
| FreeBSD | « `useradd` » |
| LINUX | « `useradd` » |

Exemple d'interface graphique de création de compte :




Problème : comment créer 3000 comptes en une après-midi avec cette interface ?

Problème : comment personnaliser cette interface ?



## Chapitre 6 • Gestion des comptes utilisateurs
### §6.15 • Opérations sur les utilisateurs : `useradd`, etc.

Sur LINUX, il existe plusieurs commandes de manipulation des comptes :

- « `useradd` » : ajout d'un utilisateur
- « `userdel` » : suppression d'un utilisateur
- « `usermod` » : modification d'un utilisateur
- « `su` » : changement d'identité
- « `pwck` » : vérification des fichiers « `/etc/passwd` » et « `/etc/shadow` »

La commande « `useradd` » peut copier des fichiers d'initialisation (les fichiers avec un nom commencant par « . ») à partir du dossier « `/etc/skel` ».

Sur LINUX, il existe plusieurs commandes de manipulation des groupes :

- « `groupadd` » : ajout d'un groupe
- « `groupdel` » : suppression d'un groupe
- « `groupmod` » : modification d'un groupe
- « `newgrp` » : changement de groupe
- « `grpck` » : vérification des fichiers « `/etc/group` » et « `/etc/gshadow` »





« `root` » est le nom du login UNIX de l'administrateur.

Les droits d'accès sur les objets ne s'appliquent pas à l'utilisateur « `root` ».

C'est pour cela qu'on l'appelle aussi « **super-utilisateur** ».

La propriété d'outrepasser tout droit d'accès provient de ce que l'UID vaut 0.

**NE PAS ESSAYER DE CHANGER le nom « `root` » !**
Il est codé profondément dans tous les programmes UNIX !
IMPOSSIBLE DE LE CHANGER !

Le répertoire HOME de « `root` » par le passé était « `/` ».
Maintenant sur les UNIX modernes, c'est « `/root` ».
**NE PAS ESSAYER DE CHANGER l'emplacement « `/root` » !**
Il est codé profondément dans tous les programmes UNIX !
Le répertoire doit aussi être local (pas réseau) pour pouvoir se connecter en cas de problème !

Un utilisateur normal a en général un homedir du type « `/home/jean` ».
Le répertoire peut être réseau par contre parce que les conditions de travail d'un utilisateur normal sont en principe normales (le réseau fonctionne normalement).





Il est déconseillé de travailler en permanence sous le nom de « `root` ».

Cf fond d'écran de « `root` » sous LINUX Mandrake :



Quelques règles :

1. L'utilisateur « `root` » n'a pas « `.` » dans son « `PATH` » (précédence de la commande locale par rapport à la commande système).
2. L'utilisateur « `root` » a 022 pour « `umask` » (accessibilité indispensable de certains fichiers par les utilisateurs normaux).
3. Le shell de « `root` » a traditionnellement pour prompt « `#` » (voir variable d'environnement « `PS1` »).





Exemple de Linux :

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
...
```

Ces utilisateurs fictifs sont essentiellement des propriétaires de fichiers.

Par exemple « **bin** » est le propriétaire de la plupart des exécutables :

```
-rwxr-xr-x   1 bin      bin        40960 Aug  9  1994 stty*
-rws--x--x   1 root     bin        24576 Aug  9  1994 su*
-rwxr-xr-x   1 bin      bin        24576 Aug  9  1994 sum*
```

A noter l'emploi du bit s sur la commande « `su` ».
⇒ son exécution se fait au nom du propriétaire du fichier, ici « `root` ».





(en anglais *identity*)
Syntaxe : « `id [user]` »

La commande « `id` » renvoie l'UID et les GID primaire et secondaires de l'utilisateur.

```
% id
uid=1000(besancon) gid=4(adm) groups=4(adm),0(root),3(sys),
12(daemon)
```

```
% id apache
uid=3000(apache) gid=3000(apache)
```

(en anglais *groups*)
Syntaxe : « `groups [user]` »

La commande « `groups` » renvoie les GID d'appartenance de l'utilisateur.

```
% groups
adm root sys daemon www
```

```
% groups besancon
adm mail pgsql www
```





(en anglais *substitute user*)
Syntaxe : « `su [-] utilisateur2` »

La commande « `su` » permet de changer d'identité.

La commande réussira après la demande du mot de passe de l'utilisateur cible.

**Pour hériter complètement de l'identité, utiliser l'option « - » de la commande « `su` ».**

## ◇ Quand on se trompe sur le mot de passe

```
% id
uid=4332(besancon) gid=1000(software) groups=1000(software)
% su asterix
Password: XXXXXXXX
Sorry
```

## ◇ Quand on donne le bon mot de passe

```
% id
uid=4332(besancon) gid=1000(software) groups=1000(software)
% su asterix
Password: XXXXXXXX
% id
uid=8520(asterix) gid=8506(gaulois) groups=8506(gaulois)
```





## ◇ Exemple de transmission de données parasites de la première identité à la seconde identité

```
% id
uid=4332(besancon) gid=1000(software) groups=1000(software)
% pwd
/users/adm/besancon
% su asterix
Password: XXXXXXXX
% id
uid=8520(asterix) gid=8506(gaulois) groups=8506(gaulois)
% pwd
/users/adm/besancon
% printenv
...
MAIL=/var/mail/besancon
...
```

Certaines variables d'environnement font encore référence à l'identité de départ !

◇ Exemple de non transmission de données parasites de la première identité à la seconde identité

```
% id
uid=4332(besancon) gid=1000(software) groups=1000(software)
% pwd
/users/adm/besancon
% su - asterix
Password: XXXXXXXX
% id
uid=8520(asterix) gid=8506(gaulois) groups=8506(gaulois)
% pwd
/users/stat/asterix
% printenv
...
MAIL=/var/mail/asterix
...
```

Les variables d'environnement font référence maintenant au second utilisateur exclusivement.

**Moralité : il faut toujours faire « `su - utilisateur2` »**



## Chapitre 6 • Gestion des comptes utilisateurs
### §6.22 • Passer du mode utilisateur au mode administrateur

Le passage se fait via la commande « `su -` ».

```
% su -
Password: XXXXXXXX
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
#
```

Le shell de « `root` » a un prompt traditionnellement « `#` » (voir variable d'environnement « `PS1` »).

Quitter le shell root via « `exit` ».

◇ Pour devenir root

Pour devenir root, la commande est :

```
% su - root
```

ce qui peut se simplifier encore en :

```
% su -
```





Problèmatiques :

- lancer au boot une commande au nom d'un autre utilisateur (serveur web, base de données, etc.)
- lancer via la crontab de root une commande au nom d'un autre utilisateur (serveur web, base de données, etc.)

Solution :

```
# su pierre -c commande
...

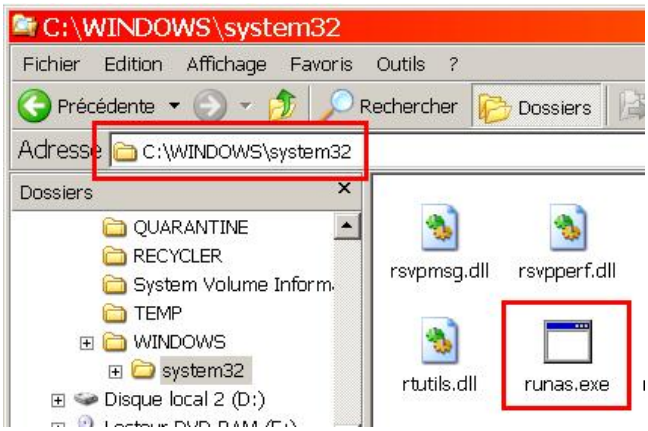
# su - pierre -c commande
...
```

Equivalent WINDOWS de la commande UNIX
« su pierre -c commande » : « runas.exe »














Problématiques :

- déléguer les priviléges de l'administrateur à des personnes **mais** pour quelques opérations seulement
- impossible d'utiliser les bits setuid et setgid avec des shell scripts

Une solution : commande « sudo », http://www.sudo.ws
http://www.courtesan.com/sudo/

**A completer...**

Fourni de base sur MacOS X.

◇ Principe

Un fichier de configuration indique qui a le droit de lancer quoi avec les privilèges de l'administrateur.

Exemple :

```
XXXXX
```

**A completer...**





Pour permettre à l'utilisateur « `apache` » de lancer le script « `update.sh` », faire :

```
# chmod u+s update.sh
# chown root update.sh
% id
apache
% ./update.sh
-> id apache
```

**A completer...**

Exemple :

```
XXXXX
% id
apache
% sudo
XXXXX
```

Sur MacOS X, pas de commande « su ».

Uniquement la commande « su » : Exemple :

```
# find / -name su -print
<-- rien
```

**A completer...**





(en anglais *change owner*)
Syntaxe : « **chown [-Rh] utilisateur objets** »

**Commande réservée à l'administrateur car sinon plus aucune protection de la machine :**

```
% ls -l
-rw-r--r--   1 besancon adm       49 Nov 23 21:05 cerise
% chown root cerise
chown: cerise: Not owner
```

## Très important :

```
% ls -l
-rw-r--r--   1 besancon adm        49 Nov 23 20:54 cerise
lrwxrwxrwx   1 besancon adm         6 Nov 23 20:54 fruit -> cerise
% su -
Password: XXXXXXXX
# chown mysql fruit
# ls -l
total 4
-rw-r--r--   1 mysql    adm        49 Nov 23 20:54 cerise
lrwxrwxrwx   1 besancon adm         6 Nov 23 20:54 fruit -> cerise
```





Options utiles :

- option « `-R` » : changement récursif dans une arborescence
- option « `-h` » : si l'objet est un lien symbolique, alors l'option « `-h` » change le propriétaire du lien et non pas le propriétaire de l'objet pointé par le lien :

```
% ls -l
-rw-r--r--   1 besancon adm        49 Nov 23 20:54 cerise
lrwxrwxrwx   1 besancon adm         6 Nov 23 20:54 fruit -> cerise
% su -
Password: XXXXXXXX
# chown -h mysql fruit
# ls -l
total 4
-rw-r--r--   1 besancon adm        49 Nov 23 20:54 cerise
lrwxrwxrwx   1 mysql    adm         6 Nov 23 20:54 fruit -> cerise
```

(en anglais *change group*)

Syntaxe : « **chgrp [-Rh] groupe objets** »

Fonctionnement identique à celui de « chown ».

A noter la possibilité de combiner un « chown » avec un « chgrp » :

- syntaxe 1 : « **chown [-Rh] utilisateur :groupe objets** »
- syntaxe 2 : « **chown [-Rh] utilisateur.groupe objets** »





L'algorithme DES sert à hasher les mots de passe.

Si l'on a accès au contenu de « /etc/shadow » (ou « /etc/passwd » en mode non protégé), on peut utiliser certains outils pour vérifier la complexité des mots de passe en appliquant des dictionnaires et des règles d'applications :

- John the ripper, disponible à « http://www.openwall.com/john/ »
- Crack, disponible à « http://www.users.dircon.co.uk/~crypto/ »

Commande de base pour détruire des comptes :
« net user jean /del »

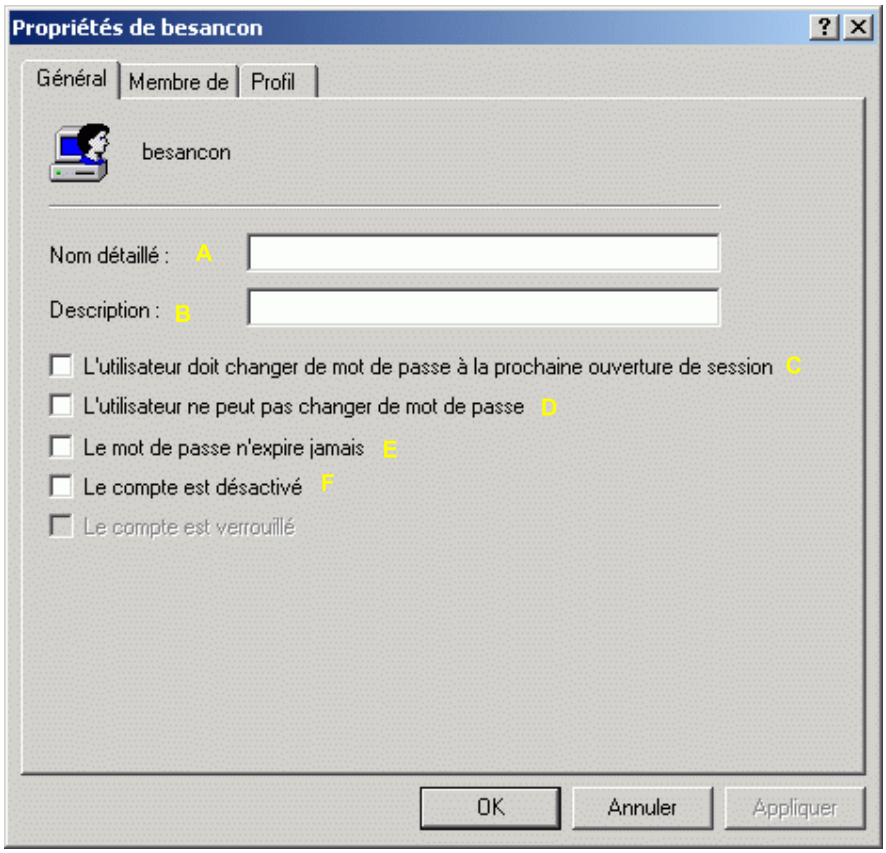
Commande de base pour créer des comptes :
« net user jean mot-de-passe /add [options] »

Les options permettent de définir les caractéristiques du compte :

| Option | Signification | Illustration |
|---|---|---|
| /ACTIVE:{YES \| NO} | compte actif ou désactivé | lettre F |
| /COMMENT:"texte" | champ « description » du compte | lettre B |
| /EXPIRES:{date \| NEVER} | expiration ou pas du compte | lettre E |
| /FULLNAME:"nom" | champ « nom détaillé » du compte | lettre A |
| /PASSWORDCHG:{YES \| NO} | l'utilisateur peut-il changer son mot de passe ? | lettre D |
| /PASSWORDREQ:{YES \| NO} | un mot de passe doit-il être défini ? | |
| /TIMES:{horaires \| ALL} | horaire de fonctionnement du compte | |
| /USERCOMMENT:"texte" | commentaire du compte | |







(en anglais *console user manager*)

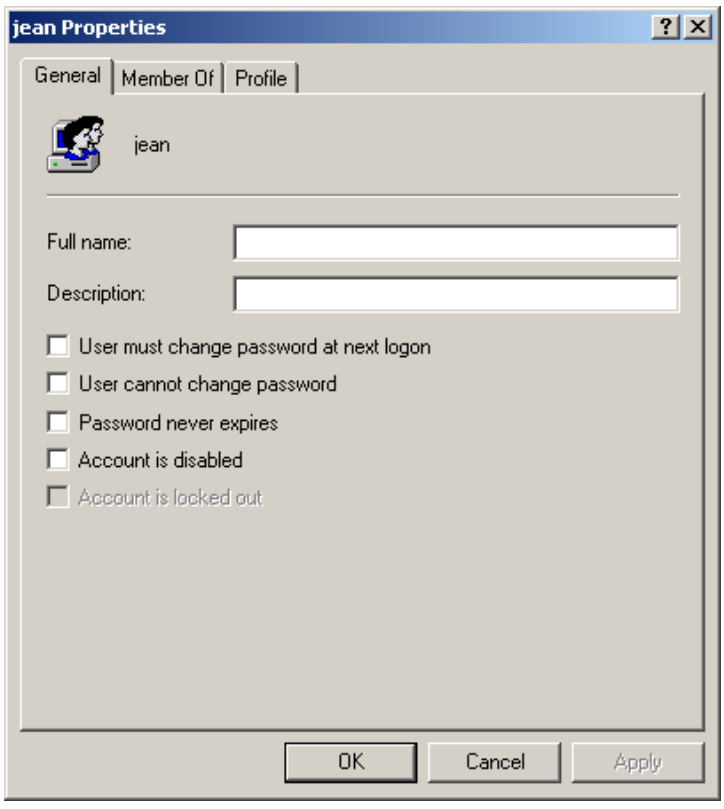
Dans le Resource Kit Windows 2000, on trouve la commande « `cusrmgr.exe` » qui permet de changer les attributs d'un compte.

Forme générale de la commande : « `cusrmgr.exe -u jean option` »



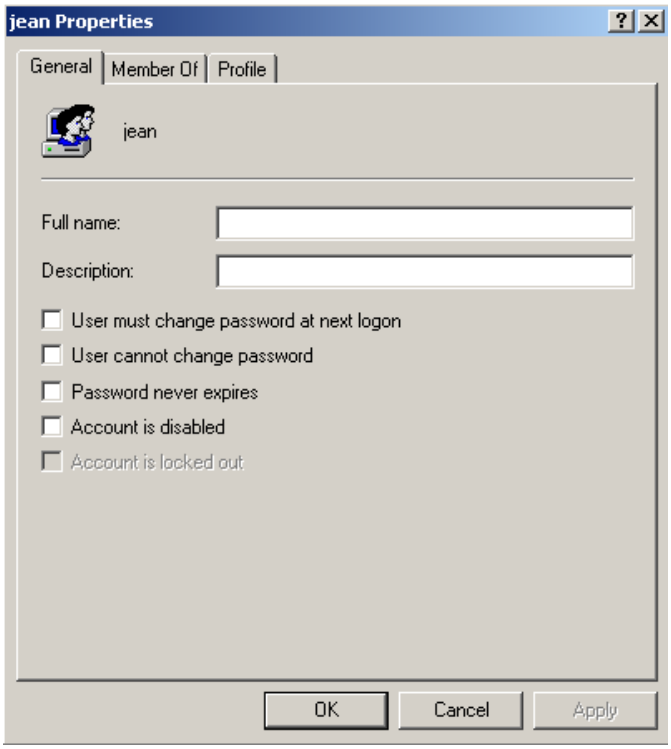


```
net user jean sesame /add
```



```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean -P "sesame"
```

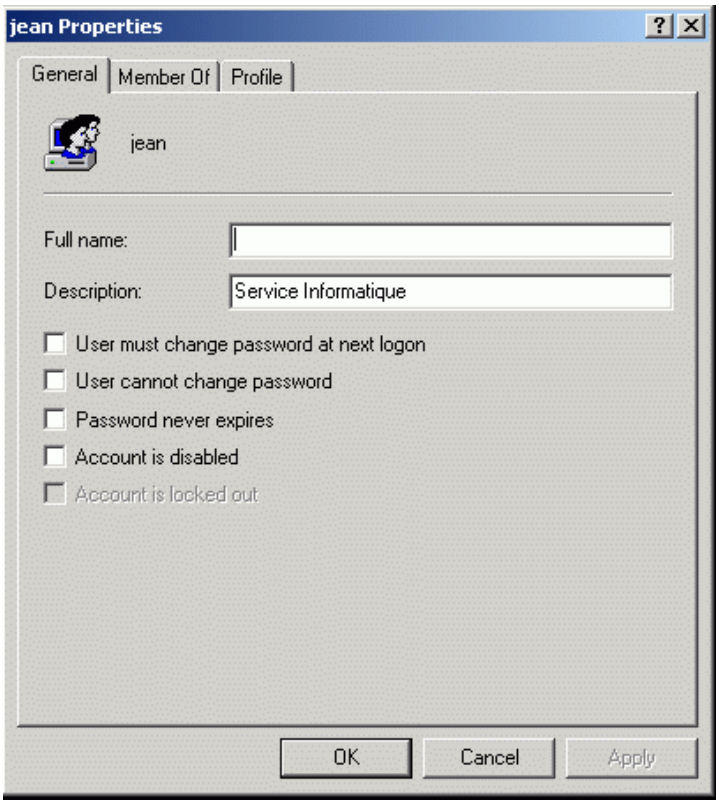
Le réglage demandé ici est non visible dans la fenêtre...



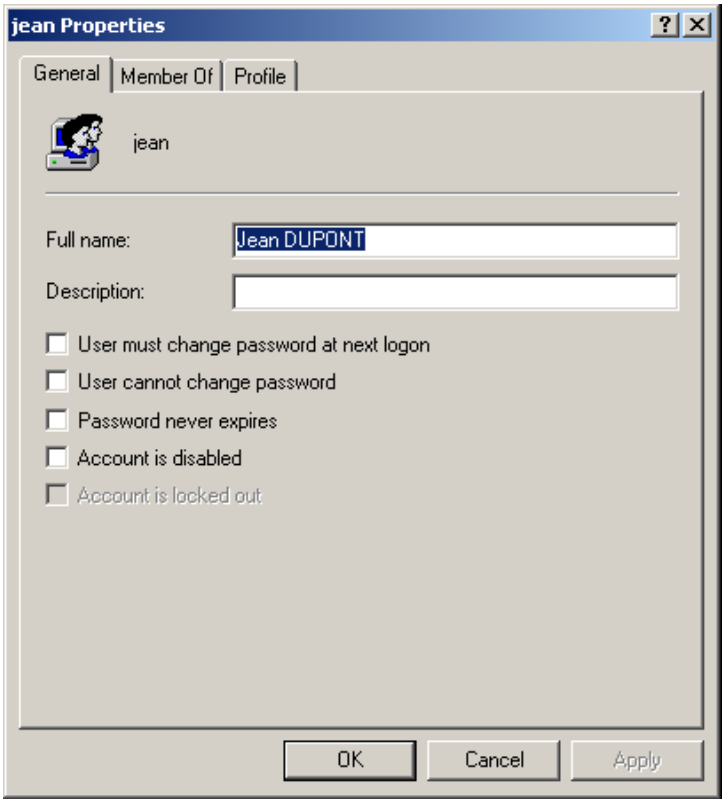





```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
-c "Service Informatique"
```



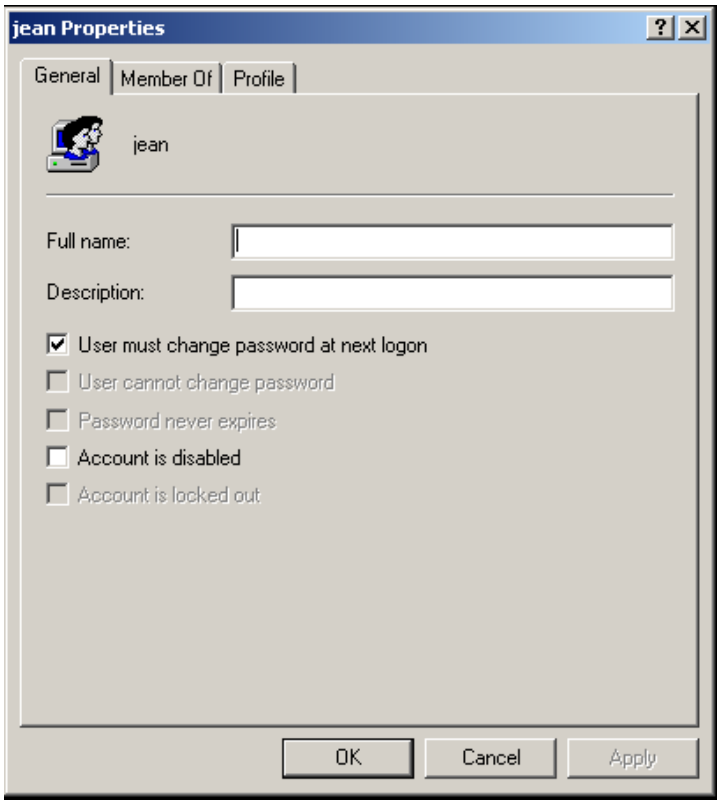
```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
-f "Jean DUPONT"
```



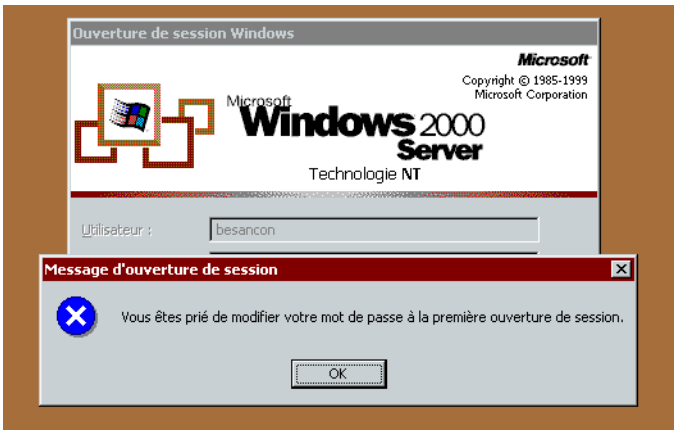





```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
+s MustChangePassword
```



L'option de changer le mot de passe se traduit ainsi au moment de la connexion initiale :



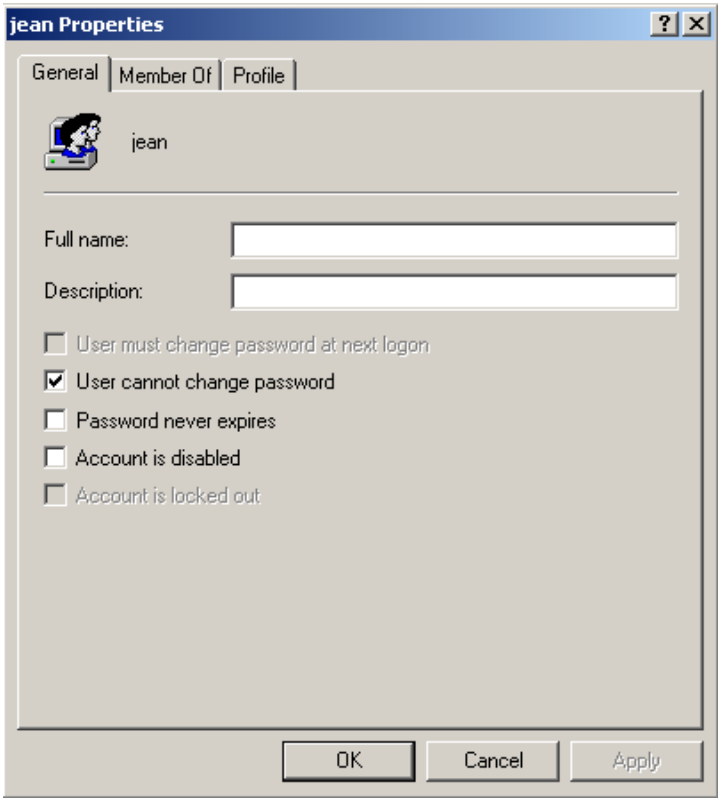




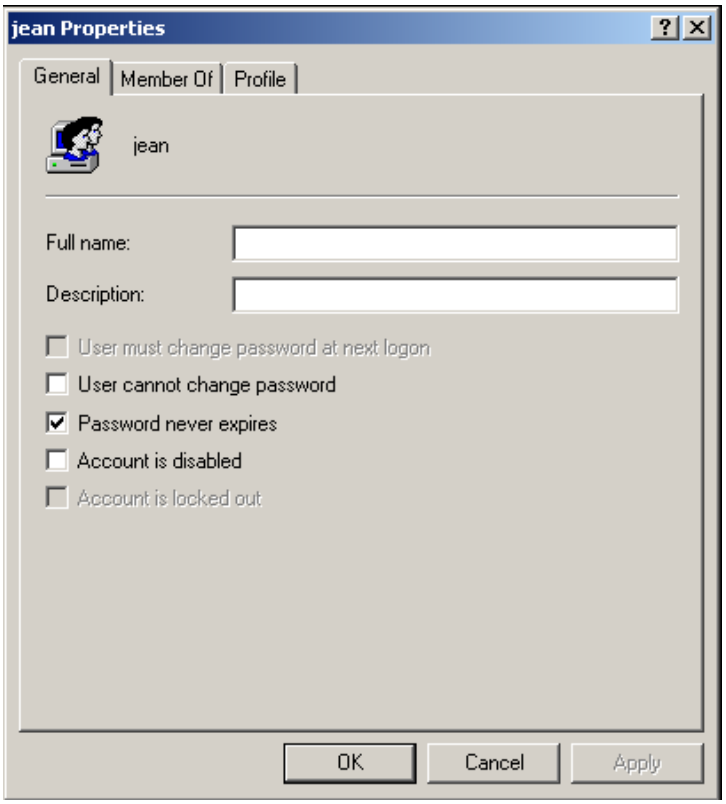




```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
+s CanNotChangePassword
```



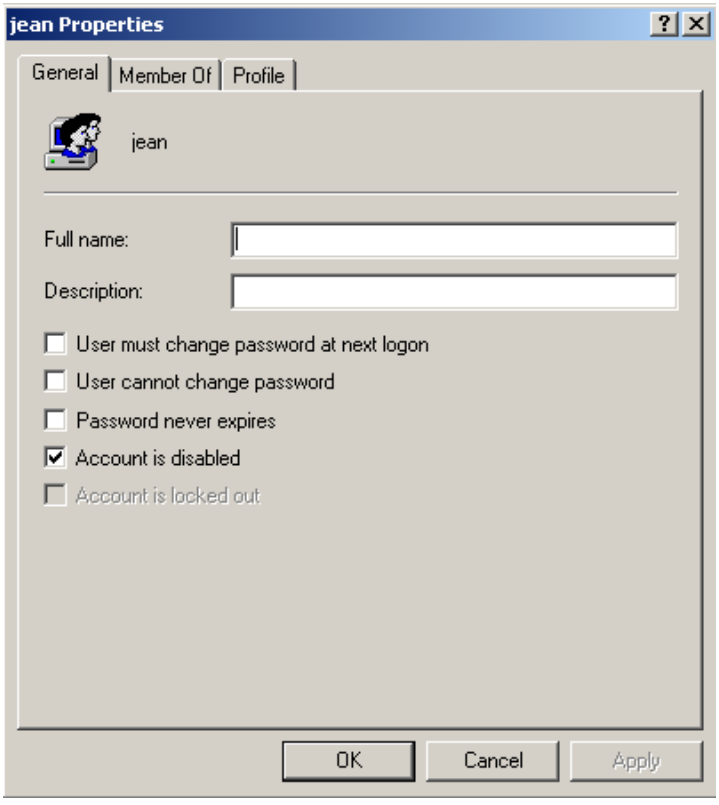
```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
+s PasswordNeverExpires
```








```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
+s AccountDisabled
```



L'option de compte expiré se traduit ainsi au moment de la connexion :

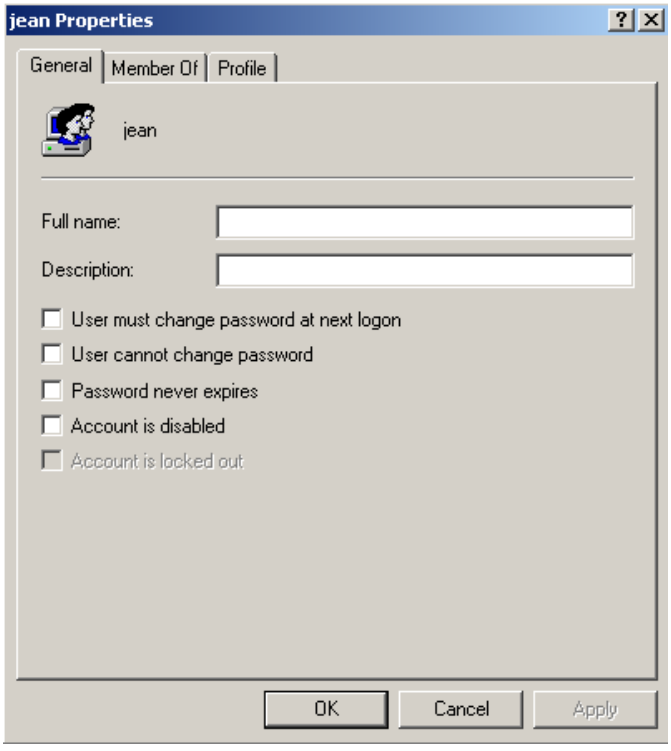




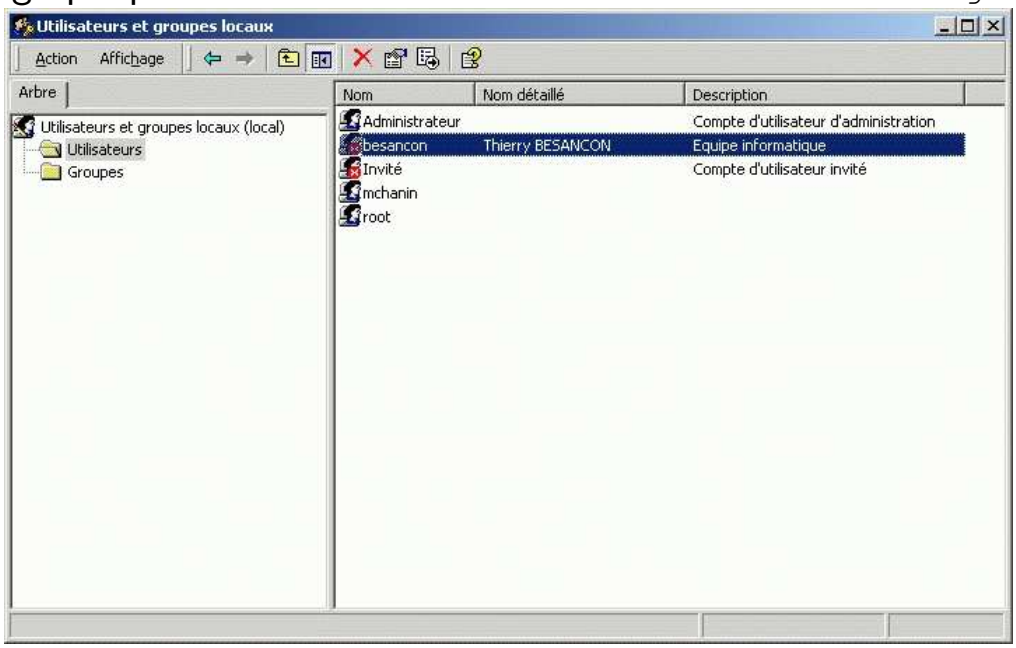



```
"C:\Program Files\Resource Kit\cusrmgr.exe" -u jean
+s AccountLockout
```

Non visible...



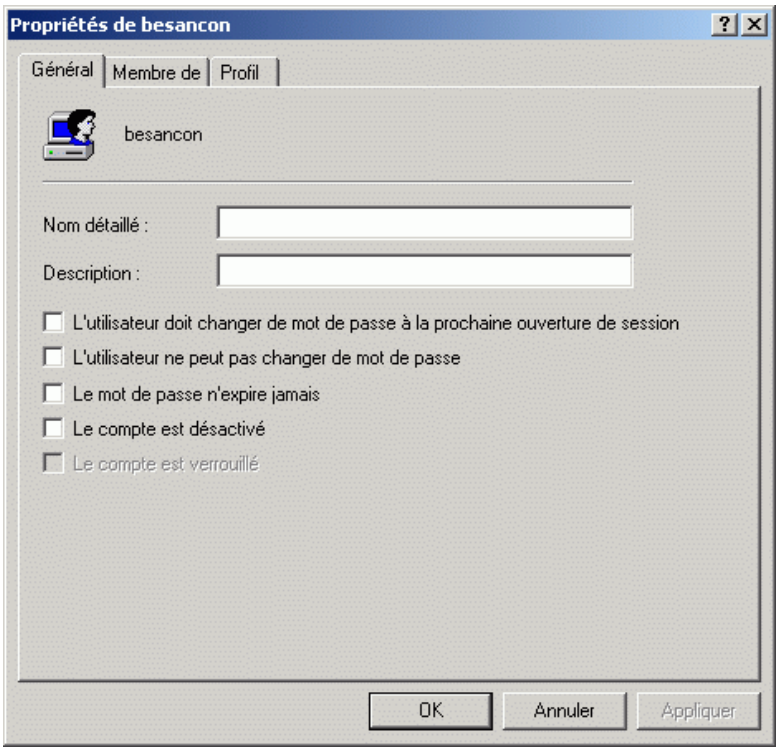
(en anglais *local user manager*)
Gestion graphique des utilisateurs : commande de base « `lusrmgr.msc` »



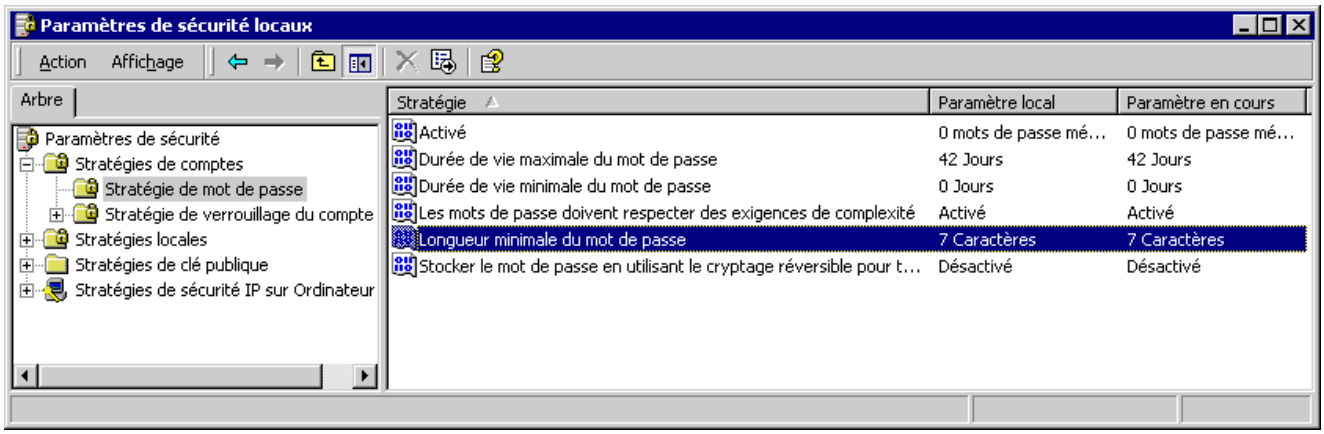





On peut bien sûr voir les propriétés d'un compte :



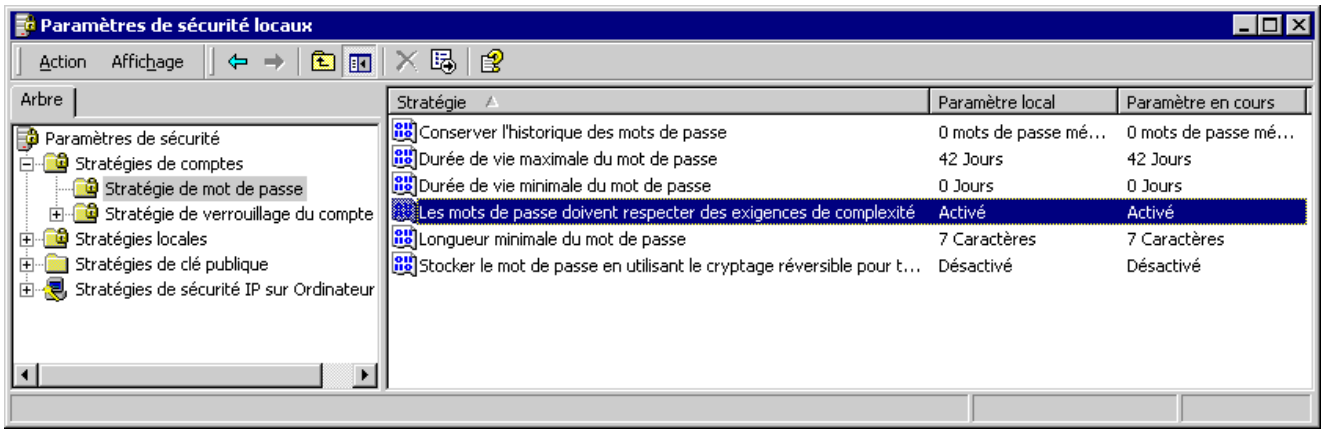
Via l'outil de politique de sécurité « secpol.msc, on peut configurer la longueur minimale des mots de passe :








Via l'outil de politique de sécurité « secpol.msc, on peut configurer la complexité des mots de passe :



Sur FreeBSD :

- login shell « `/sbin/nologin` »
- fichier « `/var/run/nologin` » qui sera affiché par « `/sbin/nologin` »

Sur LINUX :

- fichier « `/etc/nologin` » qui sera affiché par « `/sbin/login` »

```
# cat /etc/nologin
Exemple pour ARS : Interdiction de se connecter.

% ssh redhat9.example.com
Exemple pour ARS : Interdiction de se connecter.
Connection to redhat9.example.com closed.
```





Commande :

- interdiction des logins : « `change logon /disable` »
- autorisation des logins : « `change logon /enable` »
- pour savoir : « `change logon /query` »

Pas de possibilité d'afficher un message d'explications.

En cas de perte du mot de passe de l'administrateur sur un UNIX, plusieurs solutions :

- booter en single user si c'est possible (voir page 290)
  Attention : sur Solaris, le boot en single user demande le mot de passe de root
  Attention : sur certains LINUX, le boot en single user demande le mot de passe de root
- booter sur un CDROM LINUX et modifier les fichier texte « `/etc/passwd` » et « `/etc/shadow` »





En cas de perte du mot de passe de l'administrateur sur un Windows 2000 ou plus récent, solution : booter sur un CDROM LINUX avec un utilitaire de modification de la SAM

Image ISO à graver disponible sur
« `http://home.eunet.no/pnordah/ntpasswd/` »

(en anglais : *terminal type*)

Pour connaître le nom du terminal courant, utiliser la commande **tty** :

```
% tty
/dev/pts/1
```

Un fichier spécial, **/dev/tty** :

```
% tty
/dev/pts/1
% echo oui > /dev/tty
oui
```

Les entrées / sorties de « /dev/tty » sont associées à votre terminal.





Problèmatique : Root est-il autorisé à se connecter sur tout terminal ?

```
% rlogin linux.example.com -l root
Password:
Not on system console
Connection closed.
```

Pourquoi ?
Parce qu'un login au nom de root est anonyme et interdit en conséquence.

Comment contrôler l'accès ?

## ◇ Contrôle d'accès sur Linux

Au niveau de **/etc/securetty** :

```
## /etc/securetty is used by login(1); the file contains  tty
## lines  (without leading /dev/) on which root is allowed to
## login.
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
ttyp0
ttyp1
```





## ◇ Contrôle d'accès sur Solaris

Au niveau de **/etc/default/login** :

```
...

# If CONSOLE is set, root can only login on that device.
# Comment this line out to allow remote login by root.
#
CONSOLE=/dev/console

...
```

## ◇ Contrôle d'accès sur BSD

Au niveau de **/etc/ttys** ou **/etc/ttytab** :

```
...
The super-user root may only log in on those terminals
marked as "secure" in the /etc/ttytab file. Otherwise, the
super-user must log in as an ordinary user and become
super-user using su(1v). For example, if the file con-
tained:

     console   "/etc/getty Console-9600"     sun  on secure
     tty00     "/etc/getty Console-9600"     sun  on
     ...

the super-user could only log in directly on the console.
See ttytab(5) for a discussion of secure and other getty(8)
options used in /etc/ttytab.
...
```





## ◇ Contrôle d'accès sur DIGITAL Unix

Au niveau de **/etc/securettys** :

```
#
# Note: Each entry in this file represents a device (secure
# device) from which root login is permitted. For, pseudo
# ttys (ptys) a single keyword entry "ptys" means all ptys
# are to be treated as being secure. Note that the entry
# is "ptys" and not prefixed by /dev as is customary for the
# other physical devices. The absence of this key word
# implies that root login will not be permitted from all
# ptys. X displays are indicated with the display name.
#
#  <device name>
/dev/console
local:0
:0
ptys
```

Traditionnellement « /var/adm/utmp ».

Ce fichier contient des informations concernant les accès des utilisateurs et concernant les reboots de la machine.

```
struct utmp {
  char ut_user[8];            /* User login name */
  char ut_id[4];              /* /etc/inittab id(usually line #) */
  char ut_line[12];           /* device name (console, lnxx) */
  short ut_pid;               /* short for compat. - process id */
  short ut_type;              /* type of entry */
  struct exit_status ut_exit; /* The exit status of a process */
                              /* marked as DEAD_PROCESS. */
  time_t ut_time;             /* time entry was made */
};
```





Ce fichier ne grossit pas vraiment :

- un enregistrement est écrit à la connexion d'un utilisateur et mentionne le couple *(utilisateur, terminal)*
- l'enregistrement est marqué de type « USER_PROCESS »
- lorsque la connexion se termine, l'enregistrement est marqué « DEAD_PROCESS »
- à la prochaine connexion du même utilisateur sur le même terminal, on réutilisera cet enregistrement.

(en anglais *utmp extended*)
C'est une extension de la structure « utmp » :

```
struct utmpx {
  char    ut_user[32];           /* user login name */
  char    ut_id[4];              /* inittab id */
  char    ut_line[32];           /* device name (console, lnxx) */
  pid_t   ut_pid;                /* process id */
  short   ut_type;               /* type of entry */
  struct ut_exit_status ut_exit; /* process termination/exit status */
  struct timeval ut_tv;          /* time entry was made */
  int     ut_session;            /* session ID, used for windowing */
  int     __pad[5];              /* reserved for future use */
  short   ut_syslen;             /* significant length of ut_host */
                                 /* including terminating null */
  char    ut_host[257];          /* remote host name */
};
```

Le principe reste le même sauf que le fichier stockant les données est « /var/adm/utmpx »





Traditionnellement « /var/adm/wtmp ».

Ce fichier contient les historiques des connexions des utilisateurs et de l'accounting.

Ce fichier grossit au cours du temps car on y accumule les enregistrements « utmp » en fin de fichier.

(en anglais *wtmp extended*)

Le principe reste le même que celui pour « wtmp » sauf que le fichier stockant les données est « /var/adm/wtmpx »





Syntaxe : « last [options] [utilisateur] [terminal] »

La commande « last » consulte le fichier « /var/adm/wtmpx » et affiche les enregistrements à l'écran :

```
besancon ftp        localhost  Sun Aug 25 12:13 - 12:15  (00:01)
besancon ftp        localhost  Sun Aug 25 12:04 - 12:05  (00:01)
besancon ftp        localhost  Sun Aug 25 12:03 - 12:04  (00:00)
besancon console               Sat Aug 24 11:24   still logged in
besancon console               Sat Aug 24 01:12 - 11:24  (10:12)
reboot   system boot           Sat Aug 24 01:10
besancon pts/9                 Tue Aug 20 14:52 - down  (3+10:17)
besancon pts/9                 Tue Aug 20 14:51 - 14:52  (00:00)
...

wtmp begins Wed Feb 20 20:58
```

Ci joint dans la version imprimée de ce cours, un document sur les mots de passe sur MacOS X.

Adresse web du document fourni :
« `http://www.dribin.org/dave/OS_X_Passwords.pdf` »

# Passwords, Hashes, and Cracks, Oh My!

How Mac OS X Implements Password Authentication

Dave Dribin

---

# "Why Should I Care?"

- Application Developer
- System Administrator
- End User

# Authentication

- Authentication is the process of attempting to verify a user's identity
- Passwords authenticate using a "shared secret"

---

# History

- Mac OS X based on NeXTSTEP
- NeXTSTEP Unix on top of Mach
- Unix developed by AT&T Bell Labs on DEC PDP-11
- Unix based on mainframe time-sharing systems

# UNIX Time-Sharing System Version 1

- Released in 1971
- First release of Unix as we know it
- Plaintext passwords

---

11/3/71 PASSWD (V)

NAME passwd -- password file

SYNOPSIS --

DESCRIPTION passwd contains for each user the following information:

name (login name)
password
numerical user ID
default working directory
program to use as Shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the Shell field is null, the Shell itself is used.

This file, naturally, is inaccessible to anyone but the super-user.

This file resides in directory /etc.

FILES --

SEE ALSO /etc/init

DIAGNOSTICS --

BUGS --

OWNER super-user

# Plaintext Problems

"Perhaps the most memorable [example] occurred in the early 60's when a system administrator on the CTSS system at MIT was editing the password file and another system administrator was editing the daily message that is printed on everyone's terminal on login. Due to a software design error, the temporary editor files of the two users were interchanged and thus, for a time, the password file was printed on every terminal when it was logged in."

-- Robert Morris and Ken Thompson, April 3, 1978

---

# Unix Versions 3, 4, 5, 6

- Released 1973 through 1975
- Encrypted Password
- Password file is readable by all

NAME passwd -- password file

DESCRIPTION passwd contains for each user the following information:

name (login name, contains no upper case)
encrypted password
numerical user ID
GCOS job number and box number
initial working directory
program to use as Shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. The job and box numbers are separated by a comma. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the Shell field is null, the Shell itself is used.

This file resides in directory /etc. Because of the encrypted passwords, it can and does have general read permission and can be used, for example, to map numerical user ID's to names.

SEE ALSO login(I), crypt(III), passwd(I)

---

NAME crypt -- password encoding

SYNOPSIS
```
mov $key,r0
jsr pc,crypt
```

DESCRIPTION On entry, r0 should point to a string of characters terminated by an ASCII NULL. The routine performs an operation on the key which is difficult to invert (i.e. encrypts it) and leaves the resulting eight bytes of ASCII alphanumerics in a global cell called "word".

Login uses this result as a password.

FILES kept in /lib/liba.a

SEE ALSO passwd(I),passwd(V), login(I)

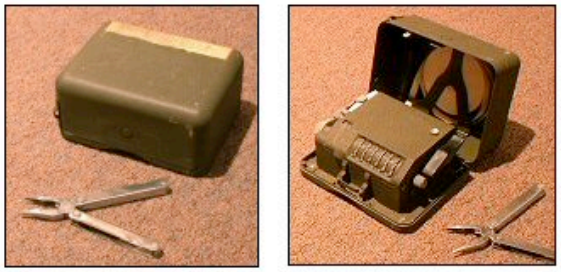
DIAGNOSTICS there are none; garbage is accepted.

BUGS --

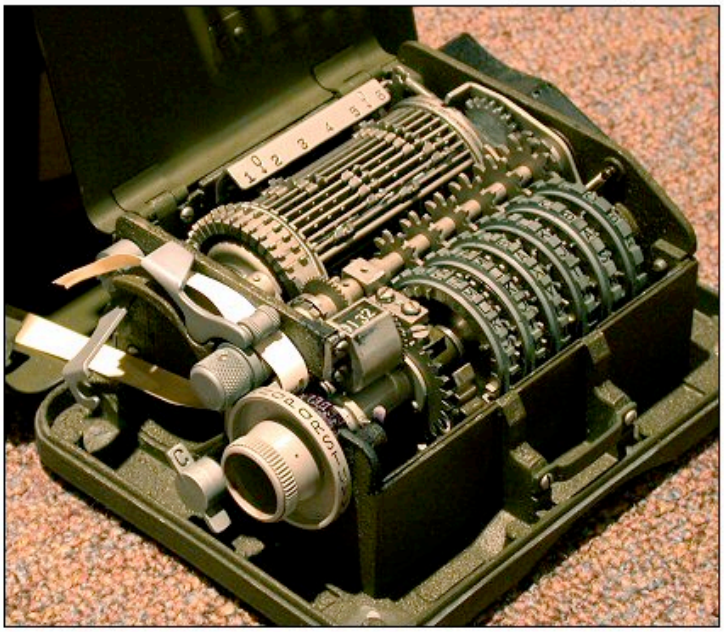
# Encrypting vs. Hashing

- Encrypting is Two-Way
  - $E_K(M) = C$
  - $D_K(C) = M$
- Hashing is One-Way
  - $H(M) = H$

---



HAGELIN M-209 CIPHER MACHINE (GVG / PD)



# World's First Hash?

"It turned out that the M-209 program was usable, but with a given key, the ciphers produced by this program are trivial to invert. It is a much more difficult matter to find out the key given the cleartext input and the enciphered output of the program. Therefore, the password was used not as the text to be encrypted but as the key, and a constant was encrypted using this key. The encrypted result was entered into the password file."

---

# Methods of Attacking Hashed Passwords

- Exploiting Algorithm Weaknesses
- Brute Force
- Dictionary
- Rainbow Tables

# Analysis of 3,289 Passwords

- 15 were a single ASCII character
- 72 were strings of two ASCII characters
- 464 were strings of three ASCII characters
- 477 were string of four alphanumerics
- 706 were five letters, all upper-case or all lower-case
- 605 were six letters, all lower-case

# PDP-11/70 Brute Force

| n | 26 lower-case letters | 36 lower-case letters and numbers | 62 alphanumeric characters | 95 printable characters | all 128 ASCII characters |
|---|---|---|---|---|---|
| 1 | 30 msec | 40 msec | 80 msec | 120 msec | 160 msec |
| 2 | 800 msec | 2 sec | 5 sec | 11 sec | 20 sec |
| 3 | 22 sec | 58 sec | 5 min | 17 min | 43 min |
| 4 | 10 min | 35 min | 5 hrs | 28 hrs | 93 hrs |
| 5 | 4 hrs | 21 hrs | 318 hrs | | |
| 6 | 107 hrs | | | | |

# AT&T Version 7 Unix

- Released in 1979
- Slower Hashing
- Less Predictable Passwords
- Salted Passwords

---

# Data Encryption Standard

- 56-bit key block cipher
- Designed by IBM, published in 1975
- Designed to be slow
- Reviewed (and revised) by NSA
- Government Standard in 1977

# DES crypt(3)

- The first 8 bytes of the key are null-padded, and the low-order 7 bits of each character is used to form the 56-bit DES key
- 12 bits of salt, encoded into 2 characters
- Example:
  - `foobar = jFUO4DTdeEmpw`
  - `foobar = cye1Y9RMJIk/2`

---

# Problems with System 7

- Limits length of password to 8 characters
- Small salt
- /etc/passwd still world readable
- DES not exportable outside US

# “The” Internet Worm

- Written by Robert T. Morris
- Unleashed in November 1988
- Designed for VAX and Sun-3 running BSD
  - 10% of Internet infected
- Included dictionary attack

---

# Shadow Passwords

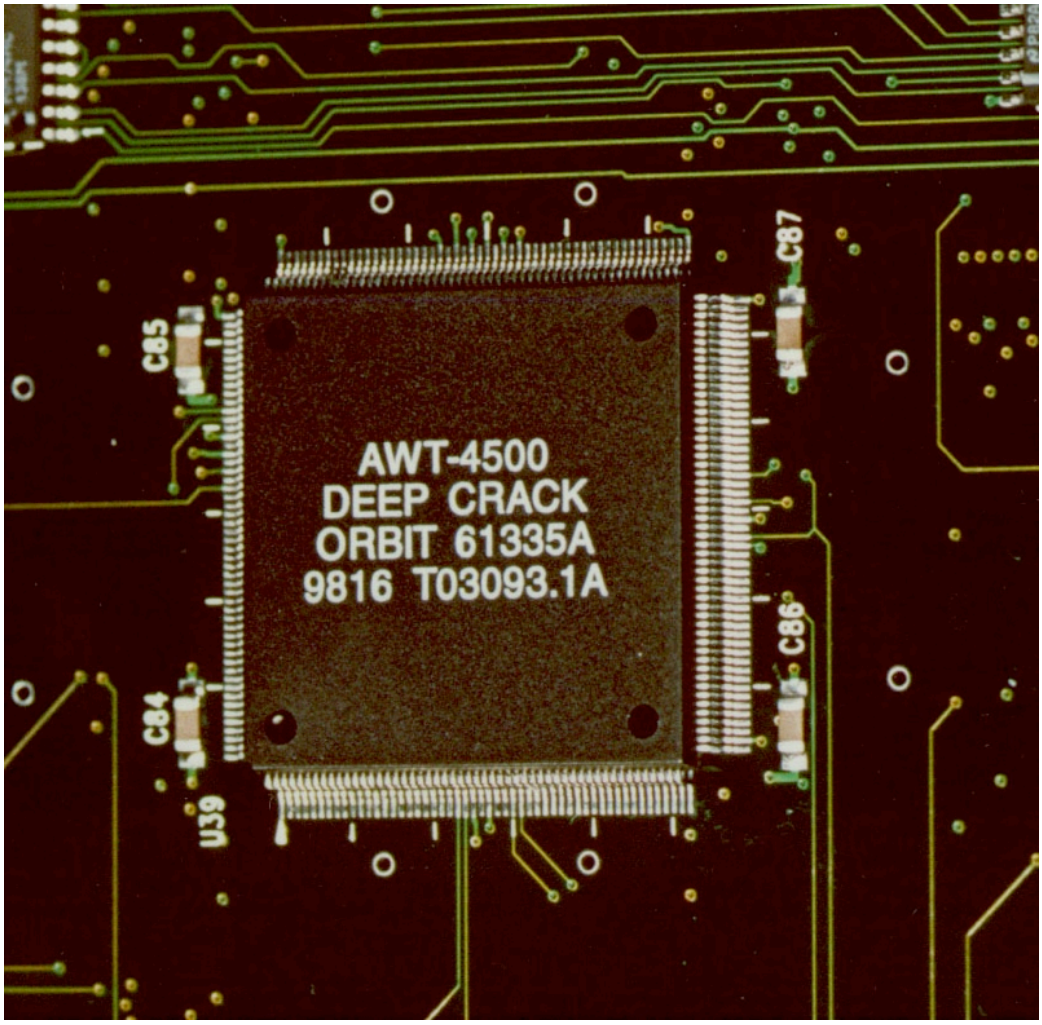
- Hashed password stored in /etc/shadow
- Introduced in System V Release 3.2 (1987)
- BSD4.3 Reno (1990)

# Message Digest 5

- 128-bit hash
- Designed by Ron Rivest (of RSA fame)
- Released in 1991

# MD5 Passwords

- FreeBSD introduced MD5 based passwords in version 2.0 (1994)
- No limit on password length
- Larger salt
- Example:
  - foobar = $1$exljwX64$cqLmh9lATzNuXZU388mzq0




# Other Algorithms

- Blowfish block cipher, 1993
- SHA-1 hash, released in 1995
- SHA-2 hash, released in 2001
- AES block cipher, released 2001
  - Replaces DES as Gov't standard

# Mac OS X 10.2, Jaguar

- Release in 2002
- Unix Version 7 DES crypt(3)
- World-readable
  - `nidump passwd .`

---

# Mac OS X 10.3, Panther

- Released in 2003
- Shadow Passwords
  - /var/db/shadow/hash
- Uses SHA-1 (no salt)
- Windows LanManager (LANMAN)

# LANMAN Weaknesses

- Case-insensitive
- Effectively limited to 7 characters
  - Truncated at 14
- Computationally cheap (10x DES)
- No salt

---

# Mac OS X 10.4, Tiger

- Released in 2005
- Still shadowed in /var/db/shadow/hash
- SHA-1 with 64-bit salt
- LANMAN only if Windows sharing enabled

# Password Attacking Tools

- John the Ripper
- RainbowCrack

---

# Rainbow Tables

- Complete, 7 character LANMAN
  - 229.96 Gb -- 1 year, 7 months
- Complete, 7 character unsalted MD5
  - 1.4 TB -- 10 years, 3 months
- Impractical on salted hashes

# Protection

- A good algorithm
- Good passwords

# “Good” Passwords

- Random is best!
  - Don’t use words (or variations)
  - Don’t use personal information
- Don’t use password hints
- The longer the better
  - Use passphrase

# Password Entropy

- Number of bits required to represent set of all possible passwords
- Example: 11 alphanumeric characters
  - Characters = 26+26+10 = 62
  - Combinations = $62^{11}$ = $5.2 \times 10^{19}$
  - Entropy, E = $\log_2(62^{11})$ = 65.5 bits

---

# Diceware

- Generates easy to remember random passphrases
- Each word created by rolling 5 dice
  - 14324 = blaze
- Each word has an entropy of 12.9 bits
  - $\log_2(6^5)$ = 12.9 bits
  - 5 words = 64.5 bits

# Diceware vs. Random

- 11 character alphanumeric (65.5 bits)
  - 3AsIPRQY6pD
- 5 word Diceware (64.5 bits)
  - 52163-42366-41666-11125-41366
  - roar mirage mccoy aback macon

---

# Four Words

- Four words (51.7 bits) are breakable with a hundred or so PCs
- "You would be content to keep paper copies of the encrypted documents in an ordinary desk or filing cabinet in an unsecured office"

# Five Words

- Five words (64.5 bits) are only breakable by an organization with a large budget
- "You need or want strong security, but take no special precautions to protect your computer from unauthorized physical access, beyond locking the front door of your house or office"

---

# More Words

- Six words (77.4 bits) appear unbreakable for the near future, but may be within the range of large organizations by around 2014
- Seven words (90.3 bits) are unbreakable with any known technology, but may be within the range of large organizations by around 2030
- Eight words (103.2 bits) should be completely secure through 2050

# Alphanumeric Brute Force

| | 7 | 9 | 11 |
|---|---|---|---|
| Entropy | 41.7 bits | 53.6 bits | 65.5 bits |
| LANMAN | 8 days | 8 days | 8 days |
| SSHA1 | 71 days | 751 years | 2 million years |
| SMD5 | 28 years | 110,342 years | 424 million years |

---

# Demo Time

Chapitre 7

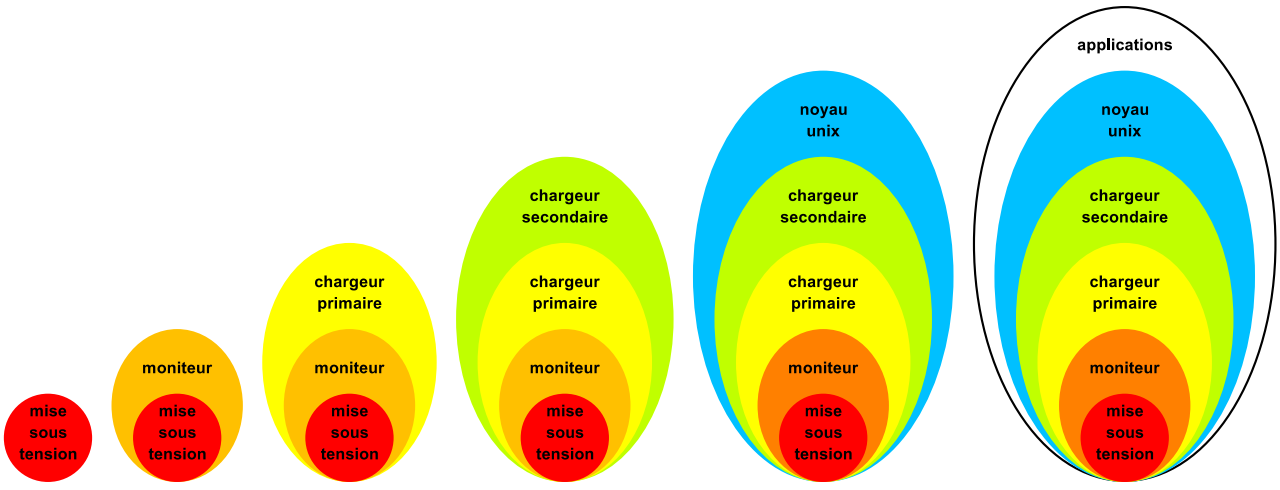
# *Démarrage d'une station UNIX*

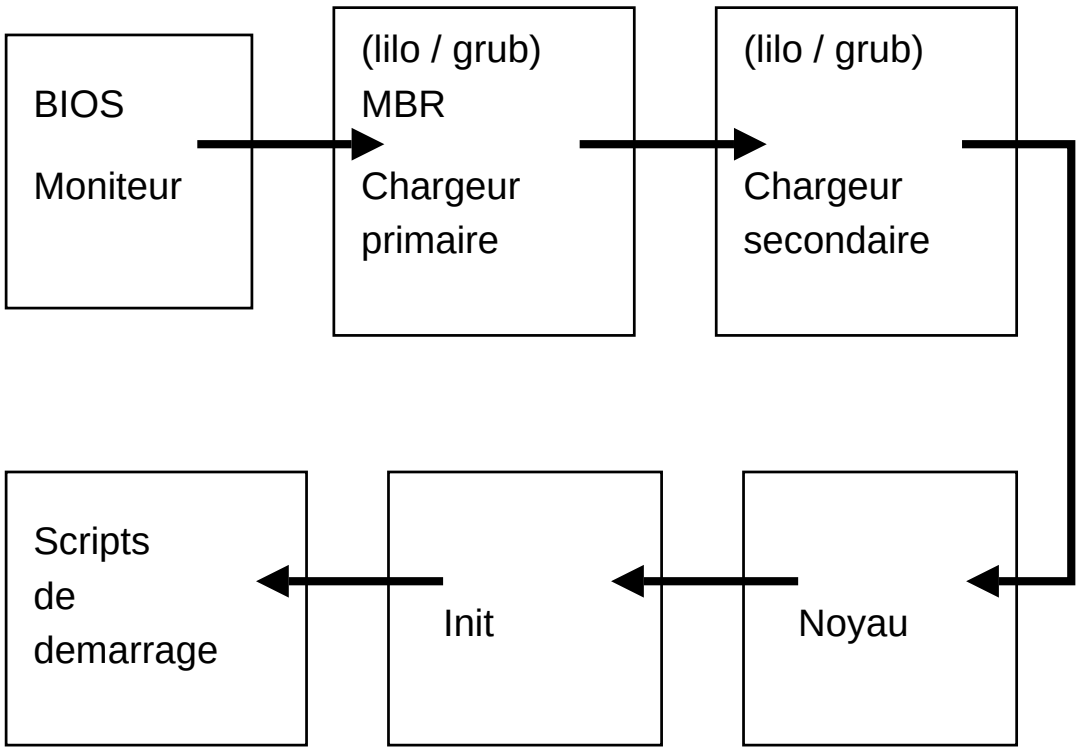




Objectif : arriver à un système 100% fonctionnel à partir de rien

Méthode : par étapes successives faisant boule de neige





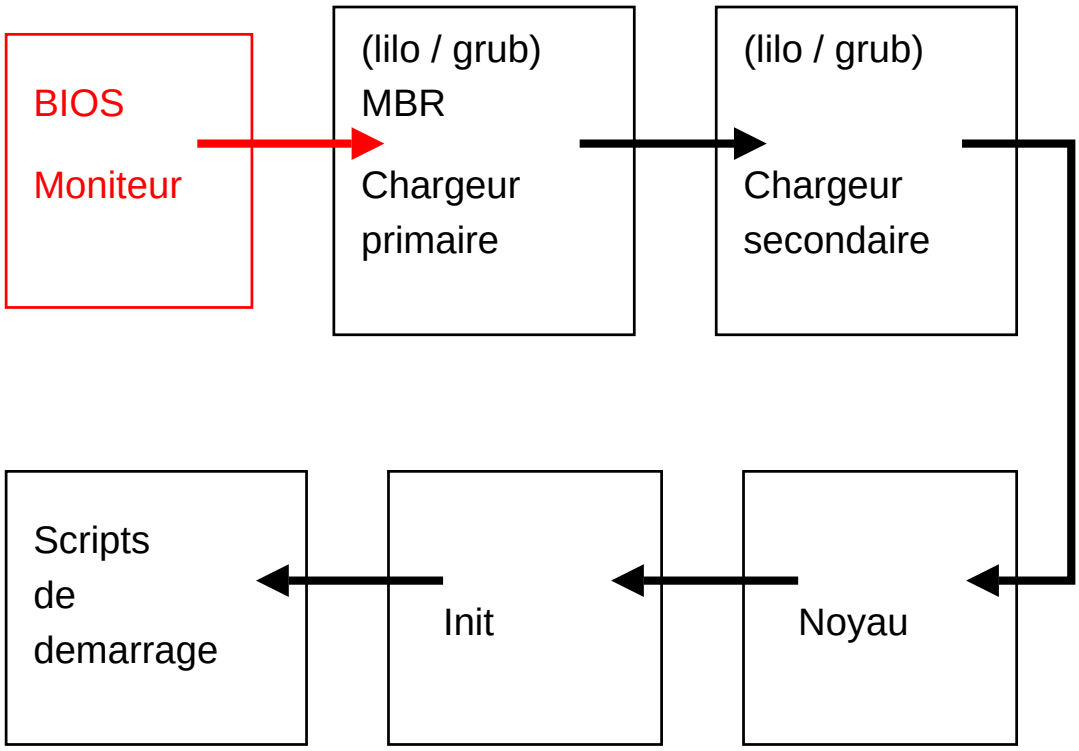





POST = *Power On Self Test*

Tests électriques et électroniques de la machine.

Après le POST, on passe la main au moniteur.








Etape 1 = exécution du premier programme :

- Dans le monde des machines UNIX, on appelle cela le moniteur.
- Dans le monde des PC, on appelle cela le BIOS (Basic Input Output System)

Rôle : fournir les routines pour les différents types de matériels et permet de charger le système d'exploitation sur les disques.

On peut ainsi utiliser des technologies récentes de disques durs juste en mettant à jour le moniteur/BIOS.

Le moniteur/BIOS peut être stocké :

- en ROM (Read Only Memory)
- en PROM (Programmable Read Only Memory)
- en EEPROM (Electrically Erasable Programmable Read Only Memory) ⇒ flashable à volonté (par exemple SUN, HP, BIOS des PC)

Sauvegarde de la configuration du moniteur/BIOS dans une NVRAM (*Non Volatile RAM*) ou EEPROM.











Rôles du moniteur :

- inspecter l'état matériel de la machine (cartes, disques, RAM)
- initialiser l'affichage
- permettre à l'utilisateur d'interagir si besoin pour divers réglages :
  - précision du disque de boot
  - consultation d'adresses mémoire
  - modification de valeurs dans EEPROM ou NVRAM
  - recherche des périphériques sur le bus SCSI
  - options de boot (single ou multi)
  - commandes de tests hardware
- **charger en mémoire vive le secteur de boot d'un certain périphérique et exécuter ce programme**.

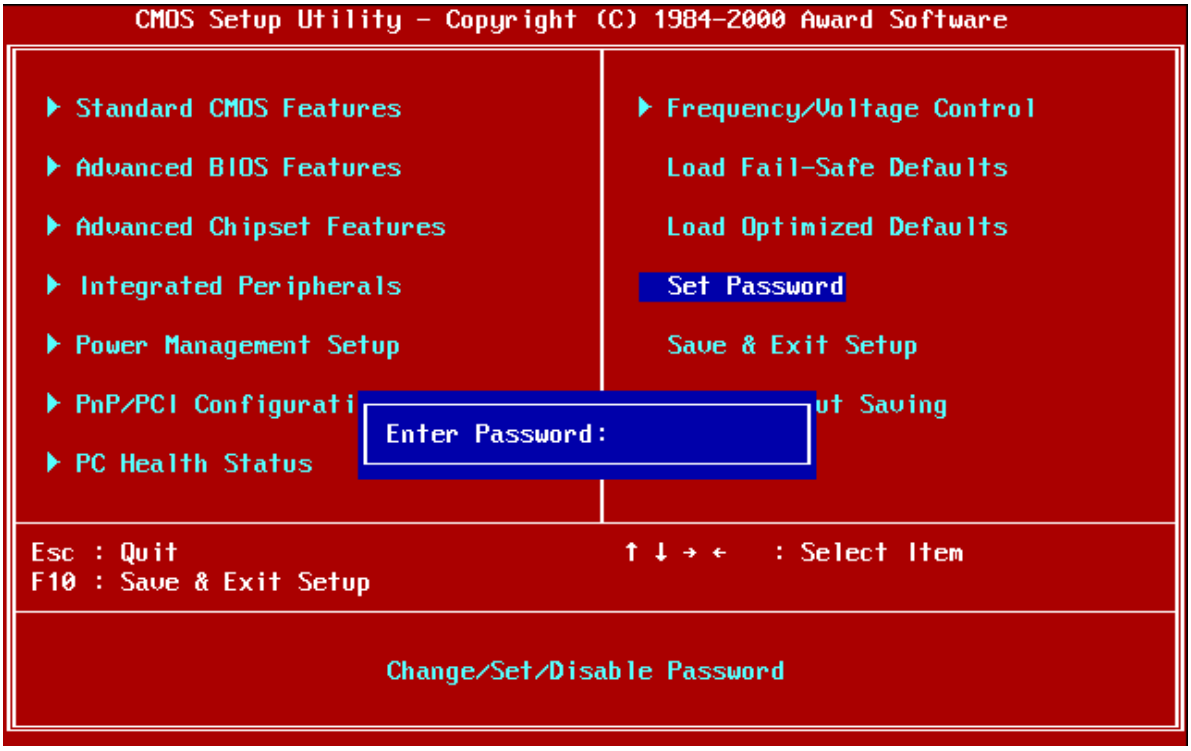
Problèmes possibles :

- si le moniteur/BIOS ne fonctionne pas ⇒ panne matérielle
- le moniteur/BIOS donne l'accès à la sélection du disque de boot ⇒ certains constructeurs permettent de protéger le moniteur/BIOS par mot de passe
- si perte du mot de passe du moniteur/BIOS ⇒ retour usine, à moins d'un super mot de passe secret connu du constructeur
- le fonctionnement du moniteur/BIOS est particulier à chaque constructeur ; plus ou moins riche selon chaque constructeur

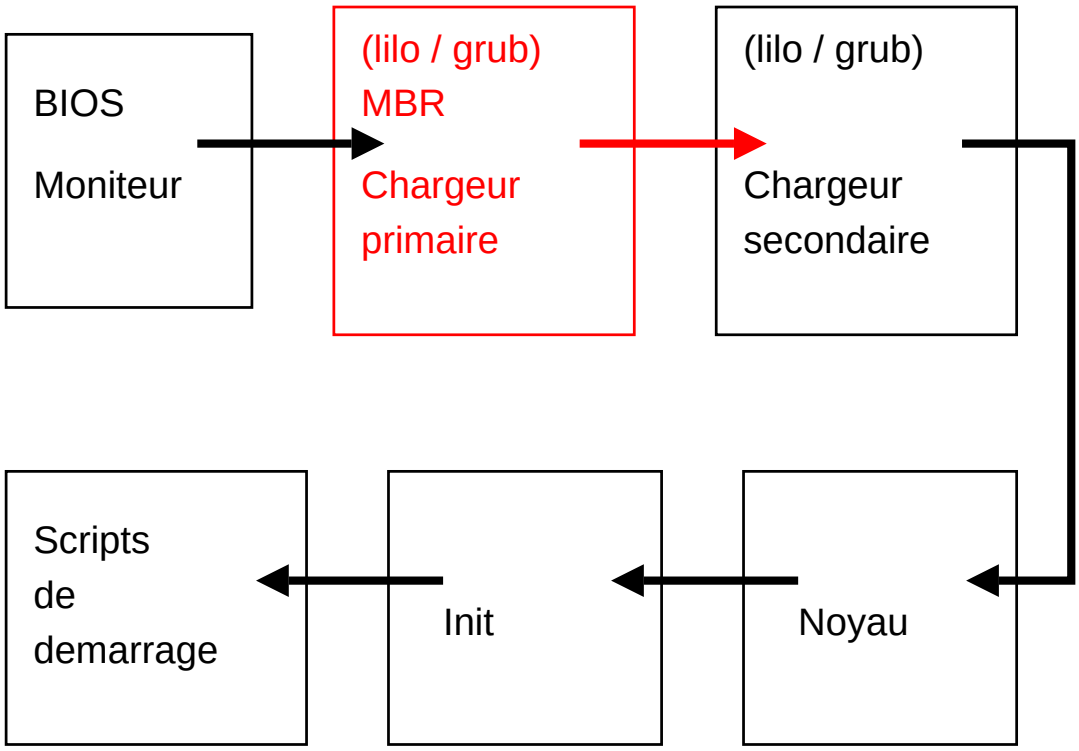
Pour le programme moniteur des stations SUN, se reporter à
`http://sunsolve.sun.com/handbook_pub/General/OBP.html`








BIOS ≡ moniteur des plateformes PC
Illustration de la possibilité de mettre un mot de passe








Chargeur primaire = gestionnaire d'amorçage = boot loader

Le chargeur primaire est stocké en général dans le secteur de boot d'un périphérique (en général un disque dur).

Il est chargé en mémoire vive par le moniteur puis exécuté.

Il va lui même charger et exécuter un programme plus gros qui poursuivra le démarrage de la machine.

Ce secteur de boot rend le disque stratégique.

A noter : Le chargement en mémoire vive du secteur de boot se fait en faisant appel uniquement aux firmwares des périphériques. Aucun pilote logiciel (*device driver*) n'entre en jeu.

La création de ce secteur de boot est un mécanisme propre à chaque constructeur UNIX.

Par exemple sur Solaris :

```
# installboot /usr/platform/`uname -i`/lib/fs/ufs/bootblk
/dev/rdsk/c0t3d0s0
```

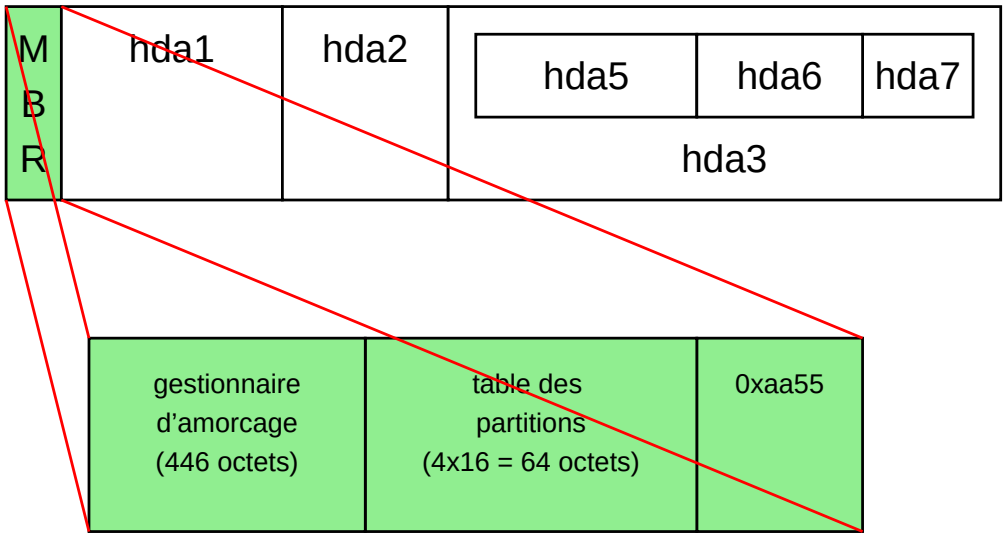




◇ Cas de LINUX

Complexité sur LINUX : le chargeur primaire et le chargeur secondaire (voir après) sont assurés par le même programme.

Le BIOS charge le secteur de boot : MBR (*Master Boot Record*).

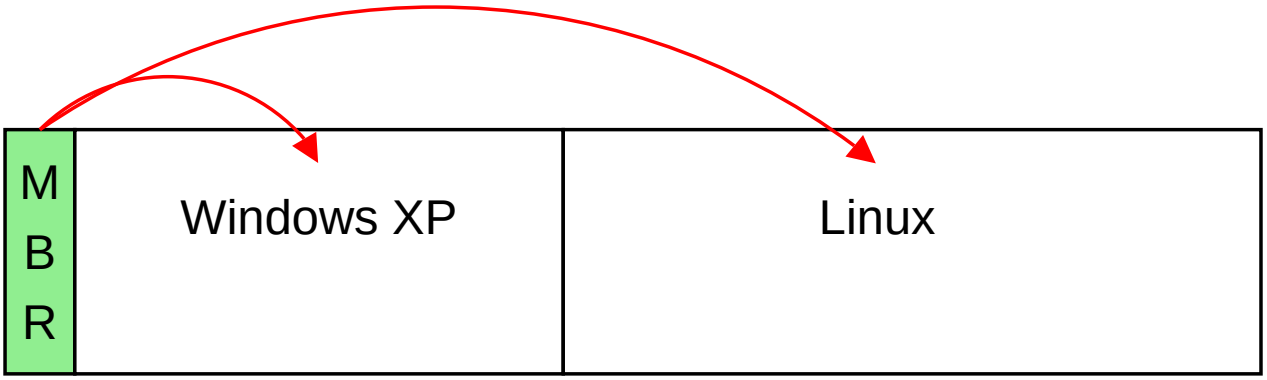


```
Absolute sector 0 (cylinder 0, head 0, sector 1)
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000   EB 48 90 00 00 00 00 00 00 00 00 00 00 00 00 00  .H..............
0010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 02  ................
0040   80 00 00 80 DF 0A 93 01 00 08 FA EA 50 7C 00 00  ............P|..
0050   31 C0 8E D8 8E D0 BC 00 20 FB A0 40 7C 3C FF 74  1....... ..@|<.t
0060   02 88 C2 52 BE 76 7D E8 34 01 F6 C2 80 74 54 B4  ...R.v}.4....tT.
0070   41 BB AA 55 CD 13 5A 52 72 49 81 FB 55 AA 75 43  A..U..ZRrI..U.uC
0080   A0 41 7C 84 C0 75 05 83 E1 01 74 37 66 8B 4C 10  .A|..u....t7f.L.
0090   BE 05 7C C6 44 FF 01 66 8B 1E 44 7C C7 04 10 00  ..|.D..f..D|....
00A0   C7 44 02 01 00 66 89 5C 08 C7 44 06 00 70 66 31  .D...f.\..D..pf1
00B0   C0 89 44 04 66 89 44 0C B4 42 CD 13 72 05 BB 00  ..D.f.D..B..r...
00C0   70 EB 7D B4 08 CD 13 73 0A F6 C2 80 0F 84 F3 00  p.}....s........
00D0   E9 8D 00 BE 05 7C C6 44 FF 00 66 31 C0 88 F0 40  .....|.D..f1...@
00E0   66 89 44 04 31 D2 88 CA C1 E2 02 88 E8 88 F4 40  f.D.1..........@
00F0   89 44 08 31 C0 88 D0 C0 E8 02 66 89 04 66 A1 44  .D.1......f..f.D
0100   7C 66 31 D2 66 F7 34 88 54 0A 66 31 D2 66 F7 74  |f1.f.4.T.f1.f.t
0110   04 88 54 0B 89 44 0C 3B 44 08 7D 3C 8A 54 0D C0  ..T..D.;D.}<.T..
0120   E2 06 8A 4C 0A FE C1 08 D1 8A 6C 0C 5A 8A 74 0B  ...L......l.Z.t.
0130   BB 00 70 8E C3 31 DB B8 01 02 CD 13 72 2A 8C C3  ..p..1......r*..
0140   8E 06 48 7C 60 1E B9 00 01 8E DB 31 F6 31 FF FC  ..H|`......1.1..
0150   F3 A5 1F 61 FF 26 42 7C BE 7C 7D E8 40 00 EB 0E  ...a.&B|.|}.@...
0160   BE 81 7D E8 38 00 EB 06 BE 8B 7D E8 30 00 BE 90  ..}.8.....}.0...
0170   7D E8 2A 00 EB FE 47 52 55 42 20 00 47 65 6F 6D  }.*...GRUB .Geom
0180   00 48 61 72 64 20 44 69 73 6B 00 52 65 61 64 00  .Hard Disk.Read.
0190   20 45 72 72 6F 72 00 BB 01 00 B4 0E CD 10 AC 3C   Error........<
01A0   00 75 F4 C3 00 00 00 00 00 00 00 00 00 00 00 00  .u..............
01B0   00 00 00 00 00 00 00 00 A8 E1 A8 E1 00 00 80 01  ................
01C0   01 00 07 FE FF 6D 3F 00 00 00 AF 39 D7 00 00 00  .....m?....9....
01D0   C1 6E 0C FE FF FF EE 39 D7 00 BD 86 BB 00 00 FE  .n.....9........
01E0   FF FF 83 FE FF FF AB C0 92 01 CD 2F 03 00 00 FE  ........../....
01F0   FF FF 0F FE FF FF 78 F0 95 01 83 AF CC 00 55 AA  ......x.......U.
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
```

GRUB MBR example using version 0.92/0.93 code.







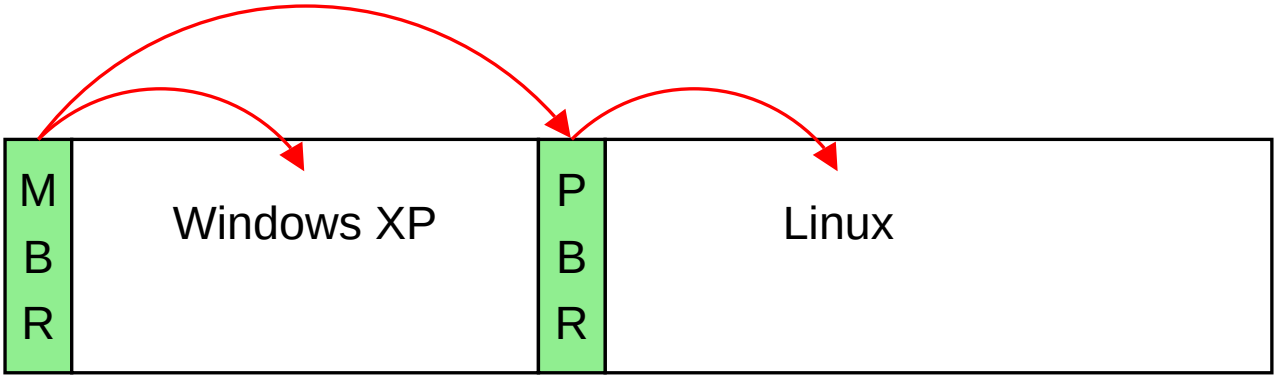

Il existe plusieurs programmes MBR. Comparaison :
« http://en.wikipedia.org/wiki/Comparison_of_boot_loaders »

Notions de partition « active ».

Certains MBR ne savent booter que certains systèmes (par exemple celui de WINDOWS XP ne sait pas booter un LINUX).

Principe de « chainloading » :

- un MBR qui ne sait pas charger la partition 2 mais tout MBR sait charger un autre MBR
- un autre MBR dans la partition 2 : PBR (Partition Boot Record)

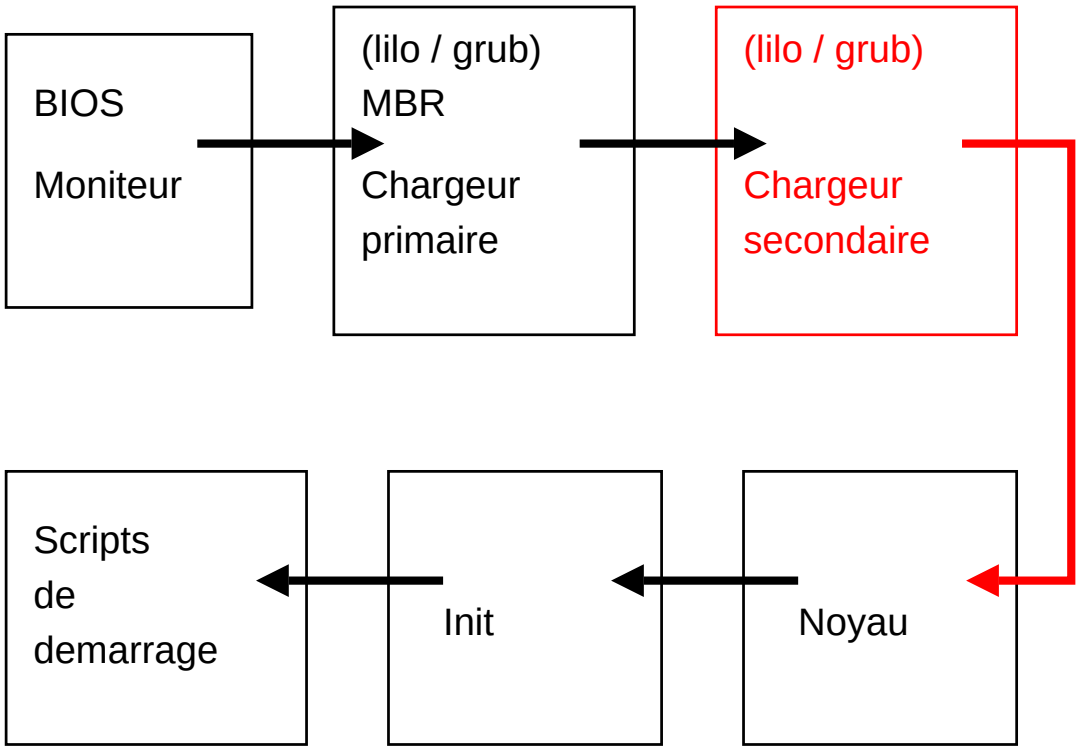







Sur LINUX, deux programmes de MBR principalement :

- LILO (*Linux Loader*) : de moins en moins utilisé
- GRUB (*Grand Unified Boot Loader*) : de plus en plus utilisé








Le chargeur secondaire est le programme chargé depuis le secteur de boot puis exécuté.

Il a une connaissance partielle de la structure d'un disque UNIX de façon à y trouver certains fichiers.

**Il charge en mémoire vive le noyau UNIX** et plus si nécessaire puis l'exécute.

A noter : Le chargement en mémoire vive du chargeur secondaire se fait en faisant appel uniquement aux firmwares des périphériques. Aucun pilote logiciel (*device driver*) n'entre en jeu.

### ◇ Cas de LINUX

Complexité sur LINUX : le chargeur primaire et le chargeur secondaire sont assurés par le même programme.

Sur LINUX, deux programmes principalement :

- LILO (*Linux Loader*) : de moins en moins utilisé
- GRUB (*Grand Unified Boot Loader*) : de plus en plus utilisé

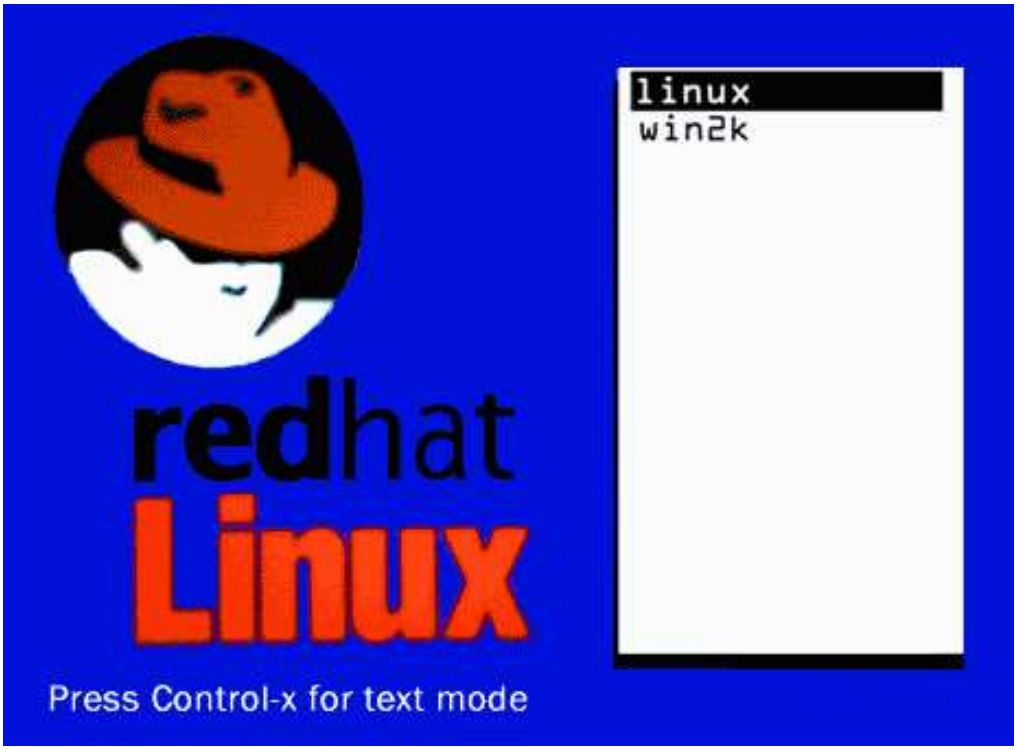
LILO et GRUB proposent une interface graphique de démarrage du système LINUX.

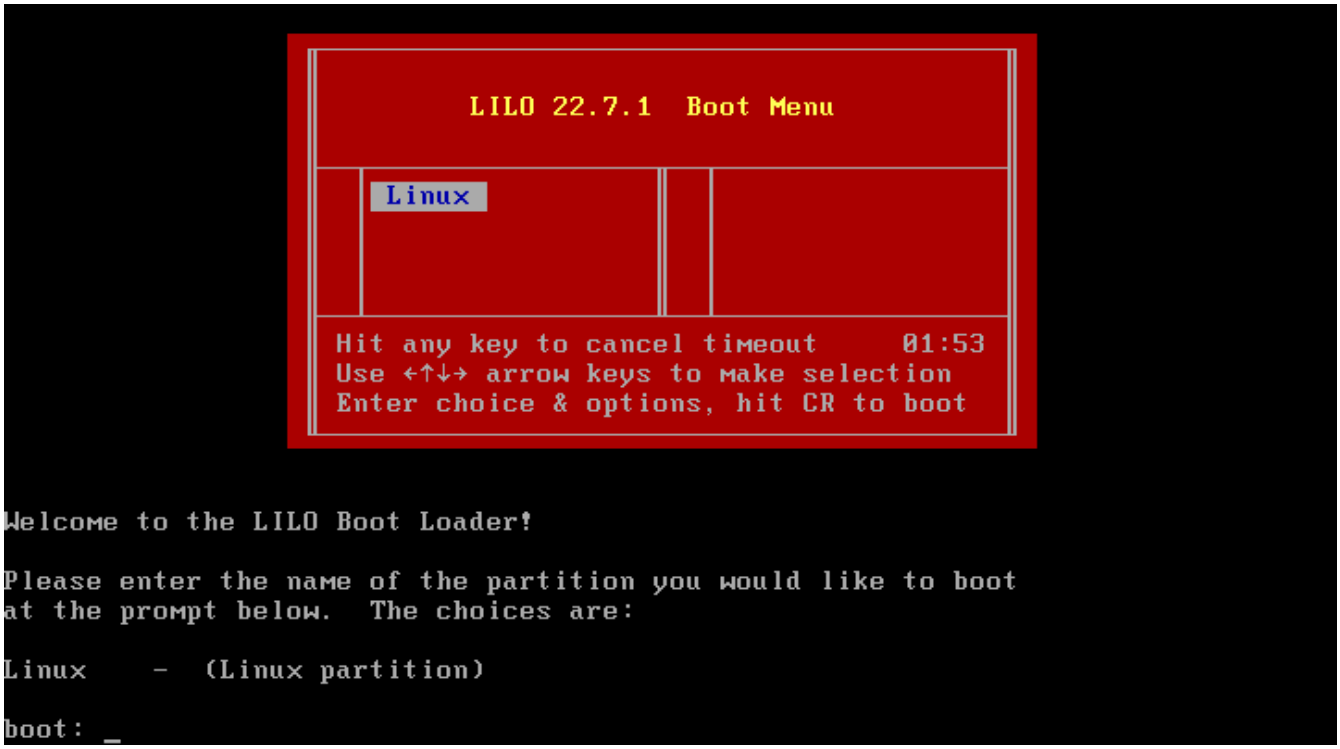
Attention : le clavier est souvent configuré en QWERTY (càd US) lorsque l'on agit dans LILO ou GRUB.





### ◇ LILO

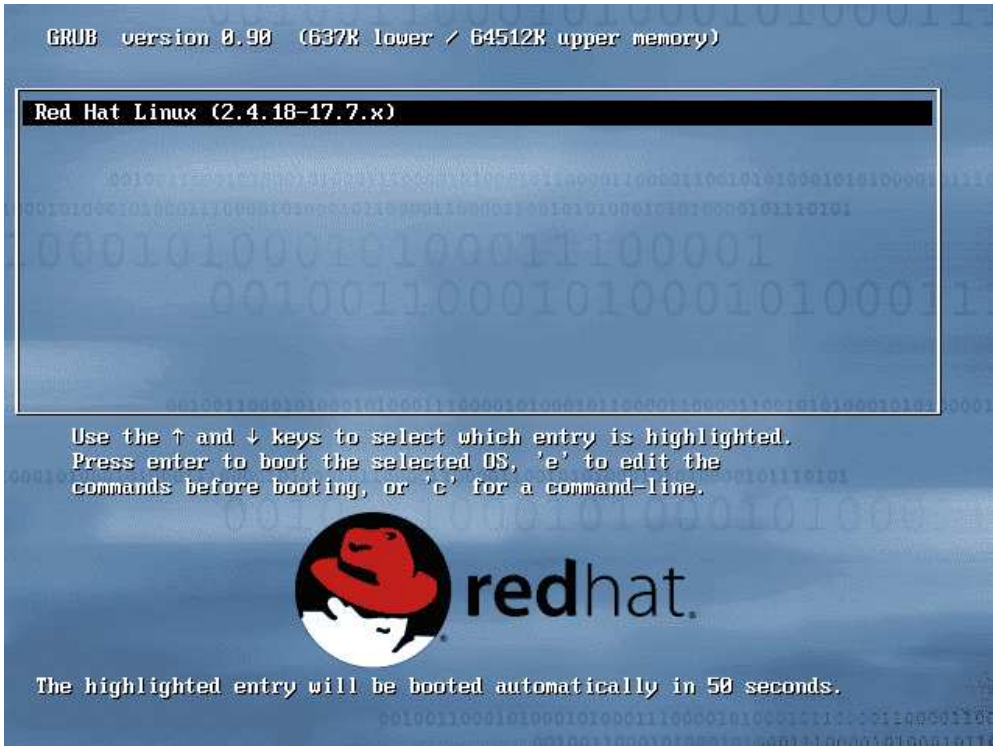



LILO 22.7.1 Boot Menu
Linux
Hit any key to cancel timeout 01:53
Use ←↑↓→ arrow keys to make selection
Enter choice & options, hit CR to boot
Welcome to the LILO Boot Loader!
Please enter the name of the partition you would like to boot
at the prompt below. The choices are:
Linux - (Linux partition)
boot: _





## ◇ GRUB

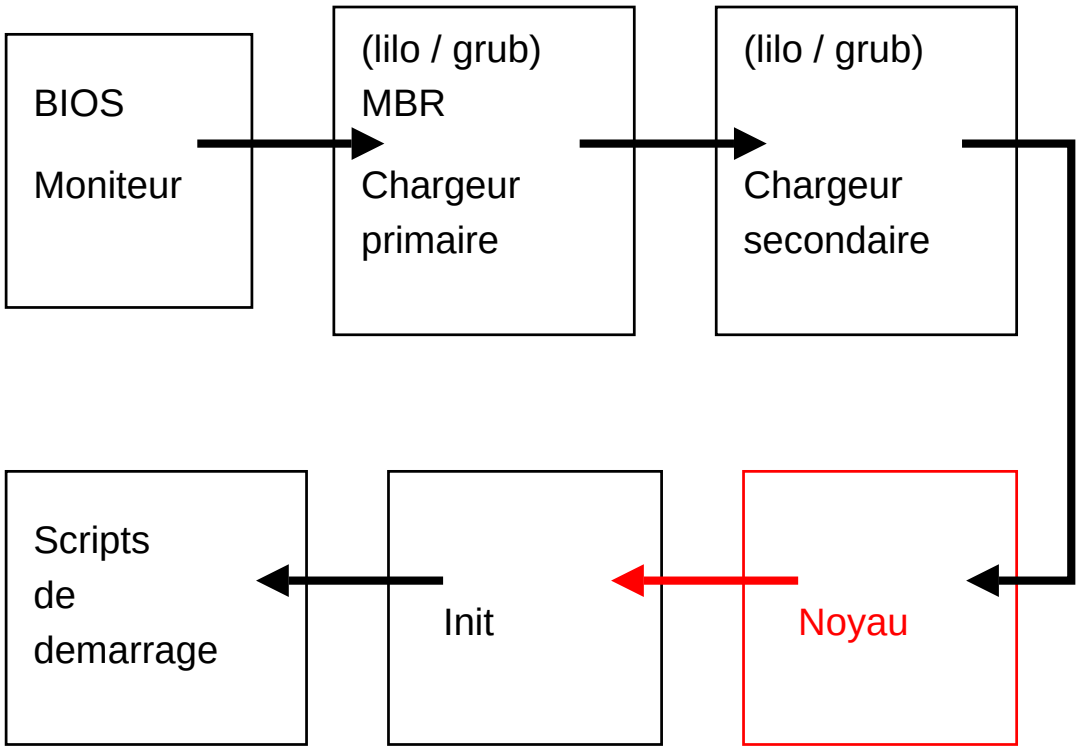

GRUB version 0.90 (637K lower / 64512K upper memory)
Red Hat Linux (2.4.18-17.7.x)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
redhat
The highlighted entry will be booted automatically in 50 seconds.

### ◇ Différences principales entre LILO et GRUB

- Nombre de filesystems reconnus :
  - LILO ne reconnait pas les filesystems usuels.
  - GRUB reconnait les filesystems ext2, ext3, reiserfs, FAT32, JFS et XFS.
- Activation des modifications :
  - LILO doit être reconfiguré par « `lilo` » car les modifications dans « `/etc/lilo.conf` » doivent être traduites en adresses de bloc.
  - GRUB n'a pas besoin d'être reconfiguré car il comprend la notion de chemin UNIX et les modifications indiquées dans « `/boot/grub/grub.conf` » sont donc directement compréhensibles.
- Systèmes démarrables :
  - LILO peut démarrer MSDOS, FreeBSD, LINUX.
  - GRUB peut démarrer quasiment tous les systèmes connus.







Le noyau est chargé puis exécuté par le chargeur secondaire.

Pas de convention en ce qui concerne le nom du noyau :

| **Système** | **Nom du noyau** |
|---|---|
| FreeBSD | `/kernel` puis `/boot/kernel` |
| Linux | `/boot/bzImage` |
| SunOS | `/vmunix` |
| Solaris 5.0 à 5.4 | `/kernel/unix` |
| Solaris $\geq$ 5.5 | `/kernel/genunix` combiné à `/platform/`\`uname -i\``/kernel/unix` |





Le noyau UNIX permet la configuration du système :

- prise en compte des périphériques
- configuration logicielle des interfaces réseau
- création de toutes les structures mémoire nécessaires
- lancement des deux premiers processus de la machine, PID 0 et 1. Ce ne sont pas des processus UNIX "traditionnels" puisqu'ils sont construits à partir de rien (par opposion aux processus ultérieurs qui auront tous un parent).

Les messages du noyau sont consultables via la commande « dmesg » :

```
% dmesg
Intel MultiProcessor Specification v1.4
    Virtual Wire compatibility mode.
OEM ID: OEM00000 Product ID: PROD00000000 APIC at: 0xFEE00000
Processor #0 Pentium(tm) Pro APIC version 17
Processor #1 Pentium(tm) Pro APIC version 17
I/O APIC #2 Version 17 at 0xFEC00000.
Enabling APIC mode: Flat.       Using 1 I/O APICs
Processors: 2
Kernel command line: ro root=/dev/hdd3
Initializing CPU#0
Detected 1002.276 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 1998.84 BogoMIPS
Memory: 1027216k/1048512k available (1294k kernel code,
17716k reserved, 1080k data, 144k init, 131008k highmem)
...
```





Suite du « dmesg » :

```
...
hda: QUANTUM FIREBALLP AS20.5, ATA DISK drive
hdb: HITACHI DVD-ROM GD-8000, ATAPI CD/DVD-ROM drive
...
eth0: Digital DS21143 Tulip rev 65 at 0xf8a96000,
00:80:C8:C9:83:F9, IRQ 5.
...
```

Processus initiaux sur LINUX :

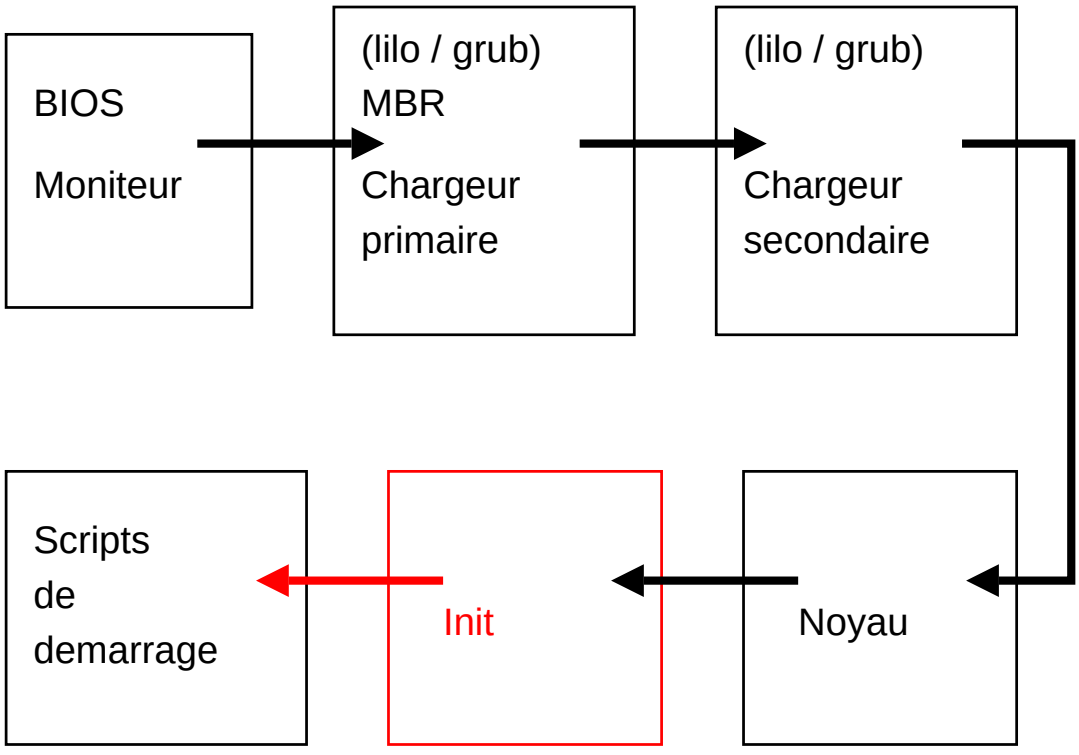
```
% ps -ax
  PID TT STAT   TIME COMMAND
    0 ?  D      0:52 swapper
    1 ?  IW     0:00 /sbin/init -
    2 ?  D      0:00 pagedaemon
...
```

Processus initiaux sur SOLARIS :

```
% ps -edf
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Oct 10 ?       0:18 sched
    root     1     0  0   Oct 10 ?       0:31 /etc/init -
    root     2     0  0   Oct 10 ?       0:01 pageout
    root     3     0  0   Oct 10 ?      29:51 fsflush
...
```







Le programme « init » a traditionnellement le PID 1.

Le programme « init » assure le reste du démarrage de la machine UNIX via le lancement de scripts de démarrage personnalisés et la surveillance des ports série.

Deux questions se posent :

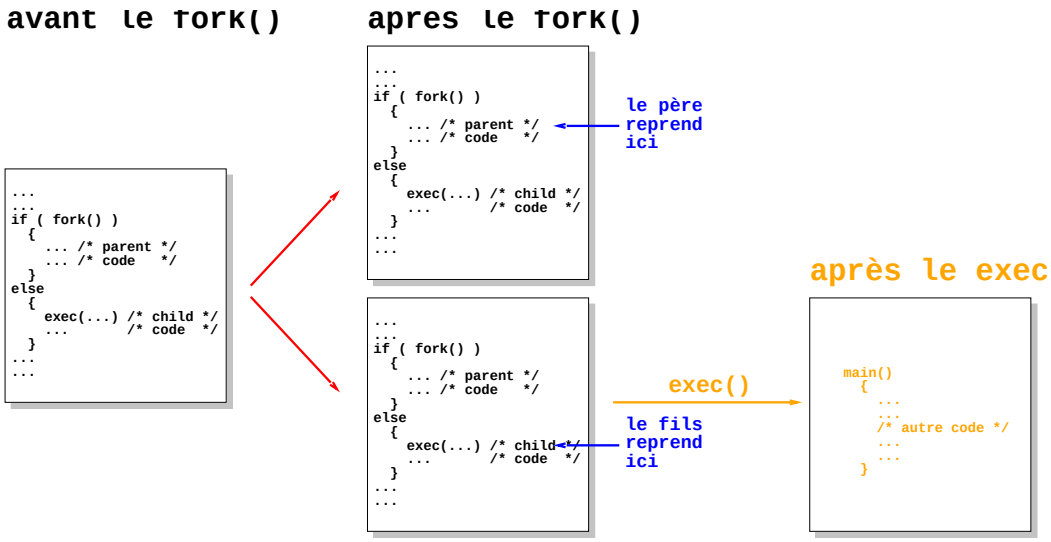
- Comment créer des processus supplémentaires ?
- Comment organiser les scripts de démarrage ?





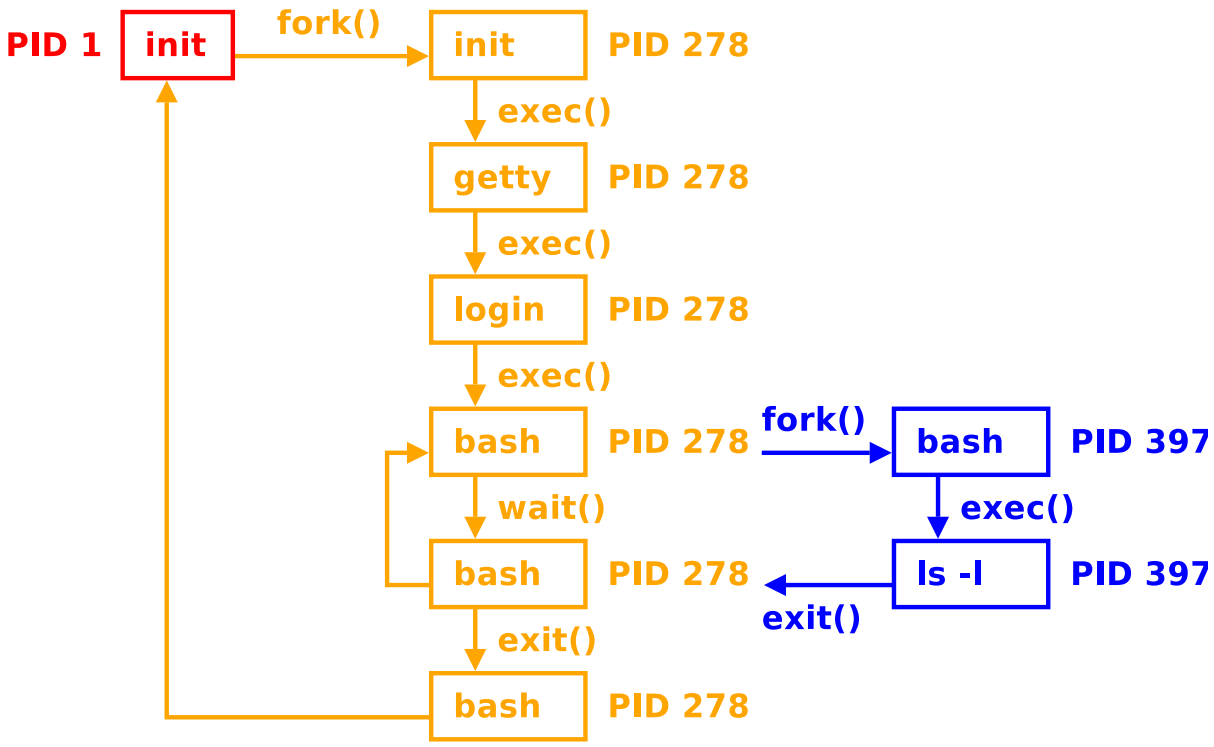
◇ Comment créer des processus supplémentaires ?

Tout processus UNIX est créé par un processus parent exécutant un « fork() », c'est-à-dire une copie de lui même (code + data), suivi d'un « exec() » pour remplacer le contenu de cette copie par un nouveau programme.



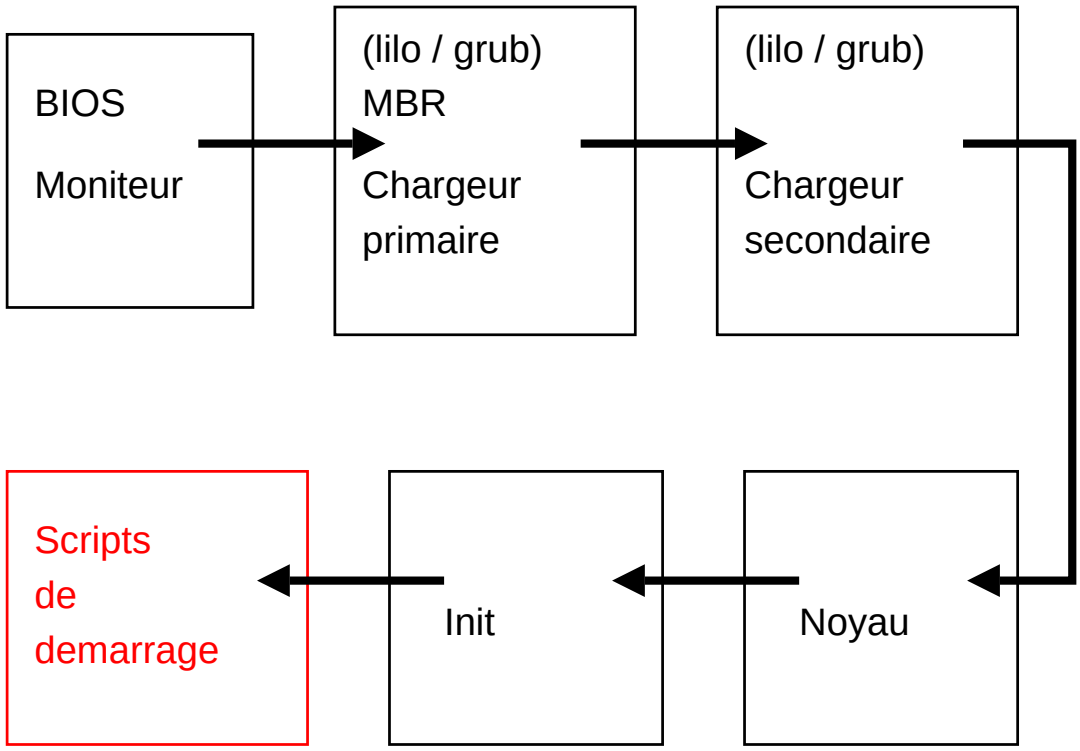

Voir page 781 pour un exemple complet de « fork() » et « exec() ».

Mécanisme de « fork() » et « exec() » appliqué à « init » :







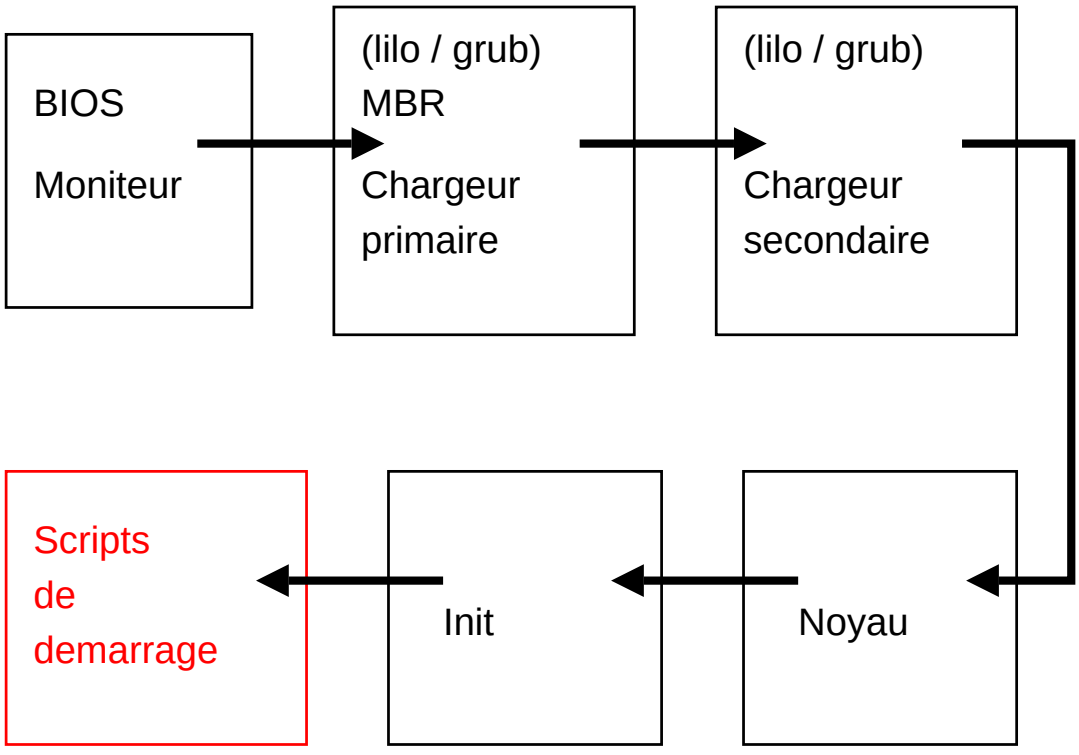
Comment organiser les scripts de démarrage que lance « `init` » ?

Il y a deux méthodes pour lancer les scripts de démarrage :

- la méthode System–V
- la méthode BSD

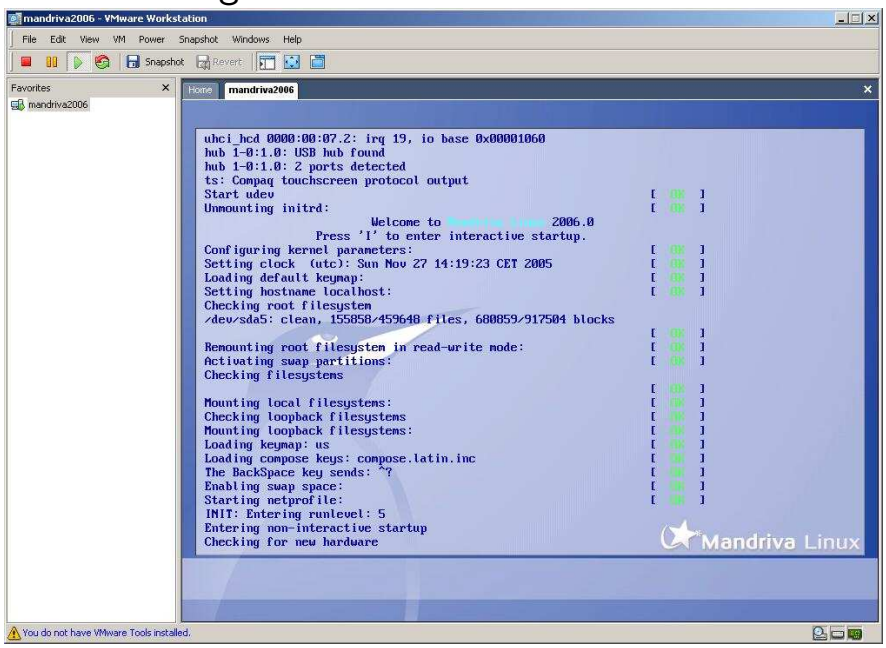






Par exemple AIX, DEC OSF1, HP-UX, IRIX, SOLARIS, LINUX, FreeBSD 5.x

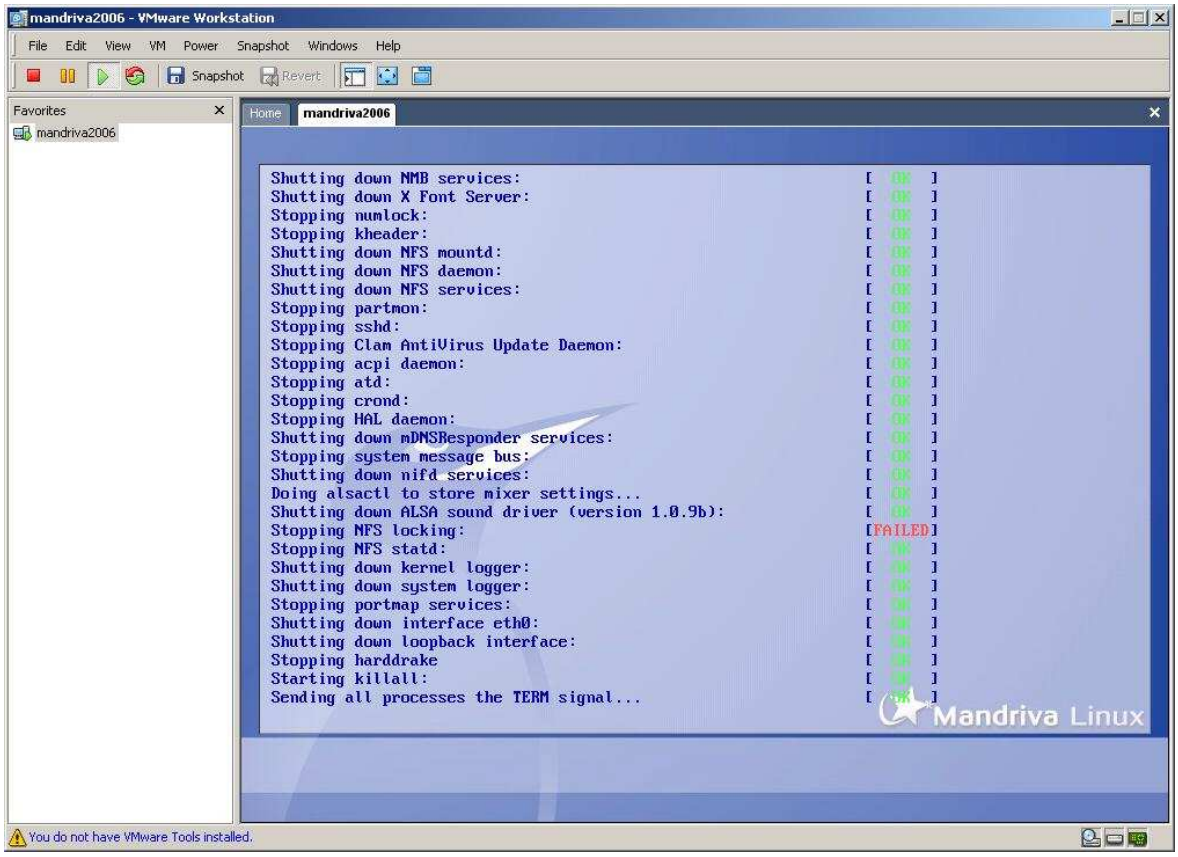
Par exemple au démarrage d'une machine LINUX Mandriva 2006 :








Par exemple au shutdown d'une machine LINUX Mandriva 2006 :



**Ce mécanisme est reconnaissable à la présence du fichier « `/etc/inittab` ».**

Exemple de fichier `/etc/inittab` SOLARIS (fichier partiel) :

```
fs::sysinit:/etc/rcS >/dev/console 2<>/dev/console </dev/console
is:3:initdefault:
s0:0:wait:/etc/rc0 >/dev/console 2<>/dev/console </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2<>/dev/console </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2<>/dev/console </dev/console
s5:5:wait:/etc/rc5 >/dev/console 2<>/dev/console </dev/console
s6:6:wait:/etc/rc6 >/dev/console 2<>/dev/console </dev/console
```





◇ Syntaxe de « `/etc/inittab` »

Ensemble de lignes au format :

```
label : niveaux : action : commande
```

### ◇ Label

Le label est une étiquette quelconque.
En général un terme adapté à ce que l'on fait.





### ◇ Niveaux

On distingue des niveaux ou *run-levels*.

Pour connaitre le niveau dans lequel on se trouve :

- sur LINUX :

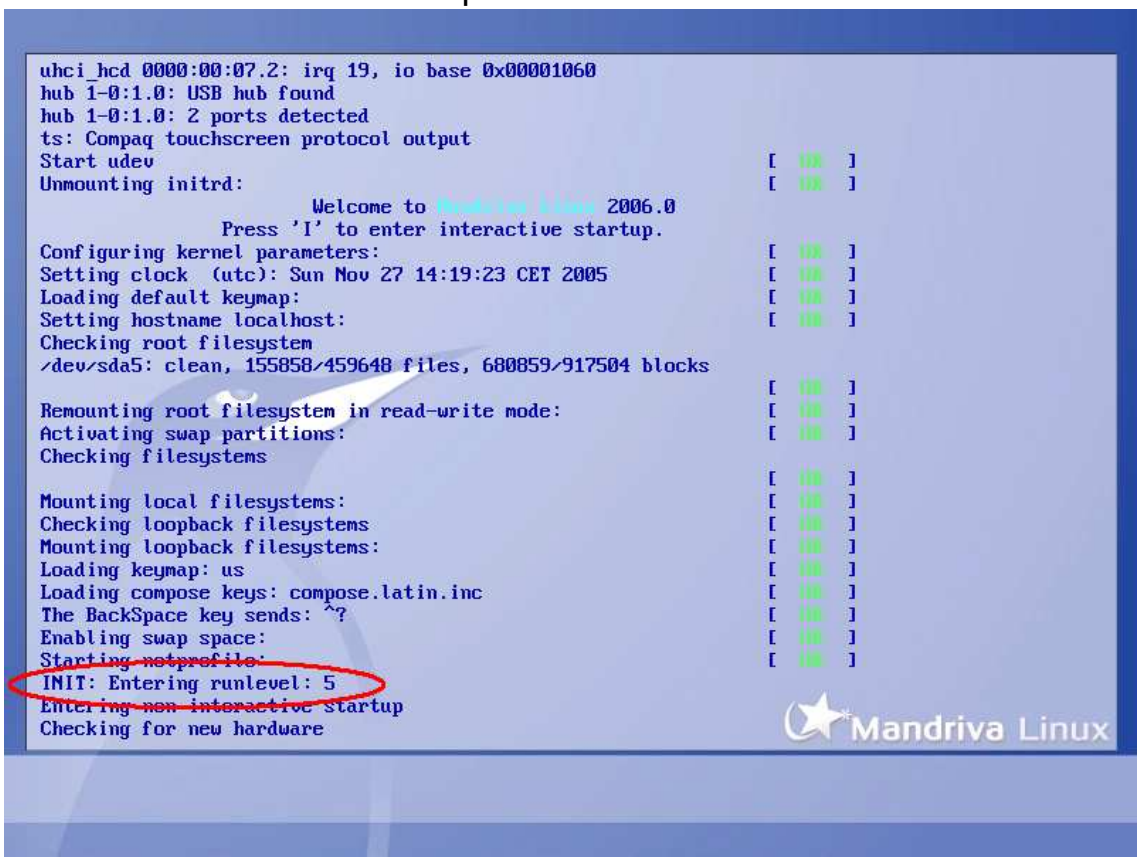
```
% /sbin/runlevel
N 5
```

(N signifie NONE ou NULL)

- sur SOLARIS (ici le 7 Mars est la date du dernier boot) :

```
% who -r
   .       run-level 3  Mar  7 22:29     3      0   S
```

On voit le runlevel sur cette copie d'écran LINUX Mandriva 2006 :








Signification des niveaux sur SOLARIS :

| SVR4 Run States | |
|---|---|
| S | Single-user (leaves filesystems mounted) |
| 0 | Power off |
| 1 | Single-user/System-admin (leaves only / mounted) |
| 2 | Multi-user, network disabled |
| 3 | Multi-user, network enabled |
| 4 | (not used) |
| 5 | PROM Monitor level |
| 6 | Halt & reboot to default state |

Faire « `man init` » pour plus de détails sur les niveaux d'un UNIX particulier.

Signification des niveaux sur Red Hat Entreprise Linux :

```
#
# inittab This file describes how the INIT process should set up
#         the system in a certain run-level.
#
# Author: Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#         Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have
#    networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
...
```





Faire « `man init` » pour plus de détails sur les niveaux d'un UNIX particulier.

### ◇ Principales actions

- `respawn`
  le programme init devra relancer la commande du champ commande chaque fois qu'elle se terminera.
- `wait`
  Lorsque init rentre dans le niveau considéré, il doit lancer la commande du champ commande et en attendre la fin avant de passer à la ligne suivante de inittab.
- `once`
  Si la commande n'a pas été lancée au préalable, init la lance et passe à la ligne suivante ; en aucun cas, cette commande ne sera relancée.
- `initdefault`
  Cette ligne précise le run-level dans lequel init se place au début de son invocation. Il ne peut y avoir qu'une ligne de ce type.





### ◇ Structure classique de `/etc/inittab`

La structure classique (LINUX, SOLARIS, etc.) est la suivante (exemple extrait de SOLARIS) :

```
fs::sysinit:/etc/rcS >/dev/console 2<>/dev/console </dev/console
is:3:initdefault:
s0:0:wait:/etc/rc0 >/dev/console 2<>/dev/console </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2<>/dev/console </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2<>/dev/console </dev/console
s5:5:wait:/etc/rc5 >/dev/console 2<>/dev/console </dev/console
s6:6:wait:/etc/rc6 >/dev/console 2<>/dev/console </dev/console
```

(On trouve parfois aussi les fichiers « `/sbin/rc0` », « `/sbin/rc1` », « `/sbin/rc2` », etc.)

Les fichiers « /etc/rc0 », « /etc/rc1 », « /etc/rc2 », etc. sont des scripts, tous bâtis sur le même modèle.

Signification du mot « rc » : *Runtime Configuration* en anglais
« /etc/rc0 » : *Runtime Configuration of runlevel 0*
« /etc/rc1 » : *Runtime Configuration of runlevel 1*
« /etc/rc2 » : *Runtime Configuration of runlevel 2*
etc.





Prenons le cas du script « /etc/rc0 ». Sa structure est la suivante :

```
#! /sbin/sh
if [ -d /etc/rc0.d ]
then
    for f in /etc/rc0.d/K*
    do
        if [ -s ${f} ]
        then
            /sbin/sh ${f} stop
        fi
    done

    for f in /etc/rc0.d/S*
    do
        if [ -s ${f} ]
        then
            /sbin/sh ${f} start
        fi
    done
fi
```

Le principe :

1. Quand on entre dans le runlevel N on lance le script « `rcN` ».
2. Le script « `rcN` » lance des programmes dont le nom commence par la lettre K et situés dans « `/etc/rcN.d` ».
   L'enchaînement de ces programmes se fait dans l'ordre alphabétique.
   Un argument est passé à chaque programme : « `stop` » (cf le K pour *Kill*)
3. Le script « `rcN` » lance des programmes dont le nom commence par la lettre S et situés dans « `/etc/rcN.d` ».
   L'enchaînement de ces programmes se fait dans l'ordre alphabétique.
   Un argument est passé à chaque programme : « `start` » (cf le S pour *Start*)





Exemple de script S :

```
#!/sbin/sh

state=$1
set `who -r`
if [ $8 != "0" ]
then
        exit
fi
case $state in
'start')
        if [ $9 = "2" -o $9 = "3" ]
        then
                exit
        fi
        echo "Starting process accounting"
        /usr/lib/acct/startup
        ;;
'stop')
        echo "Stopping process accounting"
        /usr/lib/acct/shutacct
        ;;
esac
```

Exemple de script K :

```
#!/sbin/sh

state=$1
set `who -r`
if [ $8 != "0" ]
then
        exit
fi
case $state in
'start')
        if [ $9 = "2" -o $9 = "3" ]
        then
                exit
        fi
        echo "Starting process accounting"
        /usr/lib/acct/startup
        ;;
'stop')
        echo "Stopping process accounting"
        /usr/lib/acct/shutacct
        ;;
esac
```



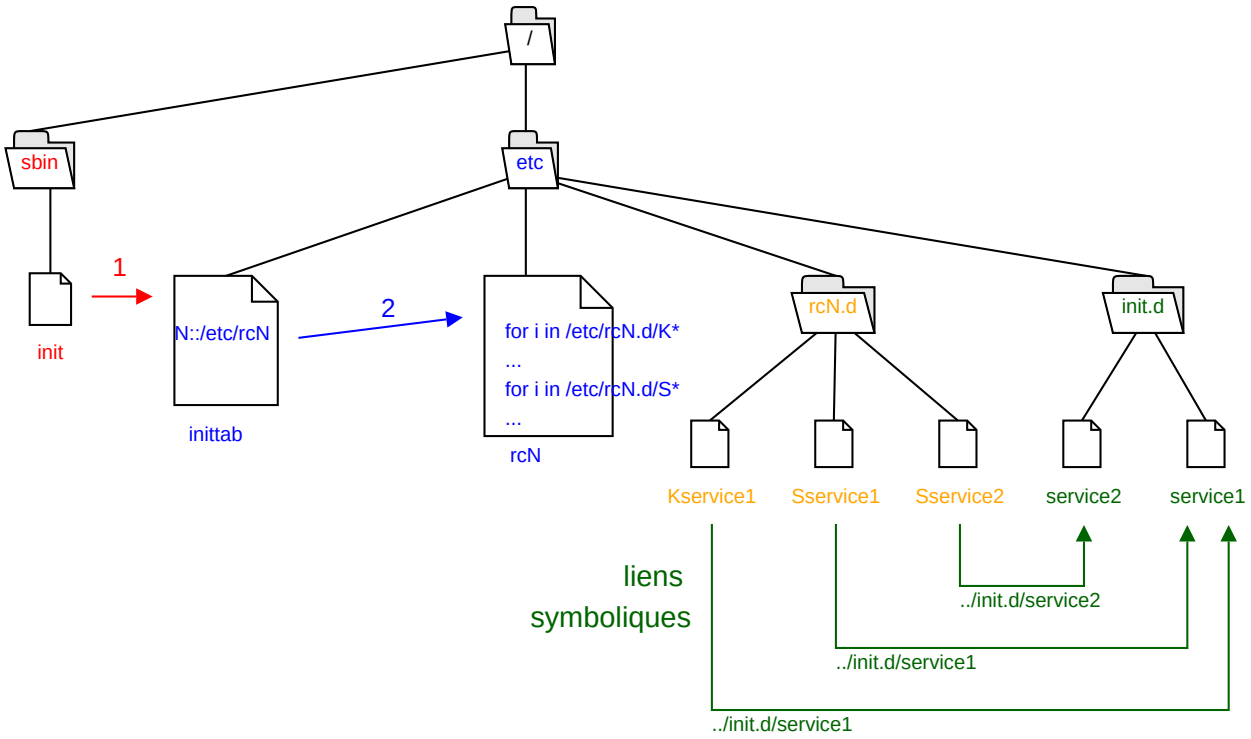


Moralité : les fichiers contiennent la même chose ; ils sont en fait identiques !

```
% ls -l /etc/rc2.d/*spc
lrwxrwxrwx  1 root       13 Nov 12 00:44 /etc/rc2.d/K20spc -> ../init.d/spc
lrwxrwxrwx  1 root       13 Nov 12 00:44 /etc/rc2.d/S80spc -> ../init.d/spc
```

**Via le mécanisme des liens symboliques (ou de liens hard), on regroupe au sein d'un même fichier la partie démarrage et la partie arrêt.**

Sur linux, on a affaire à « `/etc/rc.d/init.d` » et aux répertoires « `/etc/rc.d/rc[0-9].d` » mais le principe décrit précédemment reste le même.








## ◇ Comment personnaliser le démarrage de la machine ?

La meilleure façon de faire :

1. écrire un script qui sera placé dans `/etc/init.d`
   Par exemple :
   ```
   -rwxr-xr-x   1 root   sys   355 Jul 25   1996 /etc/init.d/ars
   ```
2. mettre dans le répertoire du bon niveau un lien symbolique vers le script précédent
   Par exemple :
   ```
   lrwxr-xr-x   1 root   sys    13 Jul 25   1996 /etc/rc2.d/S99ars -> ../init.d/ars
   ```

La commande « `telinit` » permet de passer d'un runlevel à un autre runlevel.

Syntaxe : « `telinit N` »

Par exemple :

- « `telinit 5` » fera passer du mode texte en mode graphique
- « `telinit 3` » fera passer du mode graphique au mode texte





On peut gérer à la main les scripts de démarrage vus précédemment :

- détermination des runlevels
- mise en place des liens symboliques
- mise à jour des liens symboliques

C'est une méthode assez simple.

Sous LINUX, on peut aussi gérer les scripts via la commande « `chkconfig` » (inspirée de la commande « `chkconfig` » de l'UNIX IRIX de SGI).

### ◇ Structure du shell script nécessaire à chkconfig

Le script de démarrage comprend des lignes supplémentaires de commentaires :

```
#!/bin/bash
#
# syslog        Starts syslogd/klogd.
#
# chkconfig: 2345 12 88
# description: Syslog is the facility by which many daemons \
# use to log messages to various system log files.  It is a \
# good idea to always run syslog.
...
```

Lecture :

- « 2345 » : script activé dans les runlevels 2, 3, 4 et 5
- « 12 » : démarré en « 12 ième » position ; on parle de priorité de démarrage
- « 88 » : arrêté en « 88 ième » position ; on parle de priorité d'arrêt

Tous les exemples qui suivent reposent sur le script ci-dessus pour syslog.





### ◇ Runlevels et liens symboliques

La chaine « 2345 » est interprétée correctement :

```
# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:13 /etc/rc2.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:13 /etc/rc3.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:13 /etc/rc4.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:13 /etc/rc5.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc6.d/K88syslog -> ../init.d/syslog

# chkconfig --list syslog
syslog          0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

## ◇ Désactivation d'un script dans un certain runlevel

Syntaxe : `chkconfig --level N name off`

```
# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc2.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc3.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc4.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc5.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc6.d/K88syslog -> ../init.d/syslog

# chkconfig --level 2 syslog off

# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc2.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc3.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc4.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc5.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc6.d/K88syslog -> ../init.d/syslog
```





## ◇ Désactivation d'un script dans tous les runlevels

Syntaxe : `chkconfig name off`

```
# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc2.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc3.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc4.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc5.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc6.d/K88syslog -> ../init.d/syslog

# chkconfig syslog off

# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc2.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc3.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc4.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:16 /etc/rc5.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Feb 13 09:57 /etc/rc6.d/K88syslog -> ../init.d/syslog
```

## ◇ Installation initiale/Réinstallation des liens symboliques

Syntaxe : `chkconfig name reset`

```
# rm /etc/rc[0-9].d/*syslog
# ls -l /etc/rc[0-9].d/*syslog
ls: /etc/rc[0-9].d/*syslog: No such file or directory

# chkconfig syslog reset

# ls -l /etc/rc[0-9].d/*syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc0.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc1.d/K88syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc2.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc3.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc4.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc5.d/S12syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root   16 Jul 22 13:57 /etc/rc6.d/K88syslog -> ../init.d/syslog
```





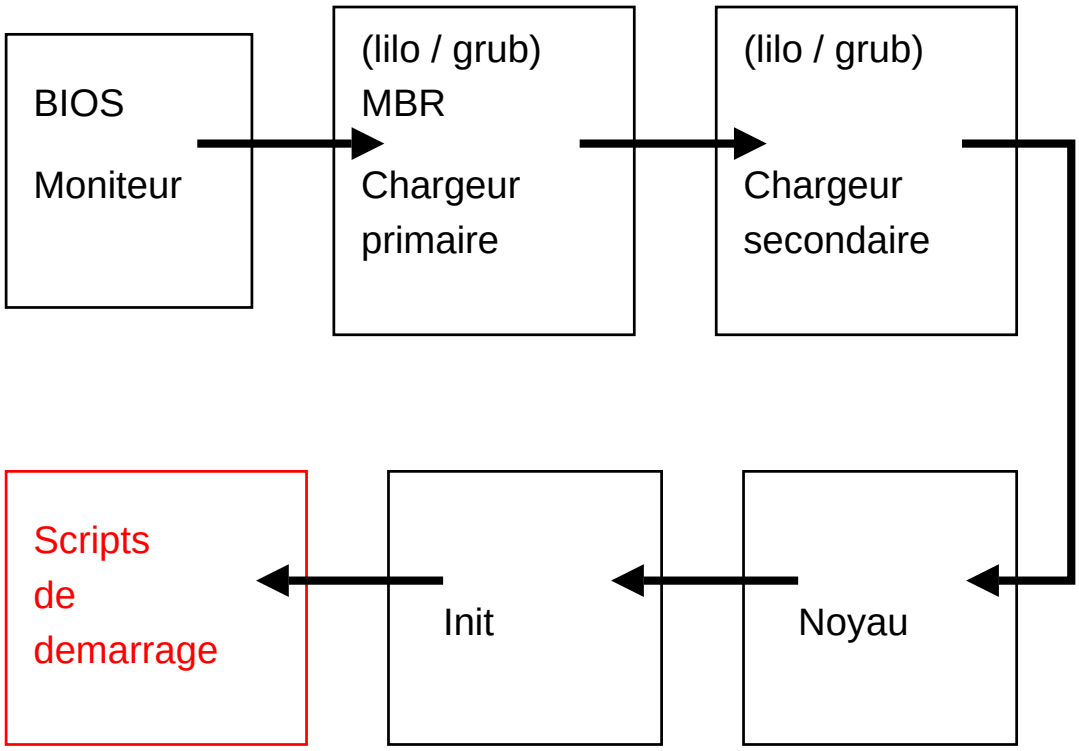
## ◇ chkconfig et processus

**La commande « chkconfig » n'agit pas sur des processus qui tourneraient déjà. Elle agit sur les scripts de démarrage de fonctionnalités.**

```
# ps ax | grep syslog
  884 ?        S      0:39 syslogd -m 0

# chkconfig syslog off

# ps ax | grep syslog
  884 ?        S      0:39 syslogd -m 0
```








(par exemple DEC ULTRIX, SunOS, FreeBSD 4.x, etc)

« `init` » n'a pas de notion de niveaux, d'états.

Le shell script « `/etc/rc` » **à caractére général** est appelé par le programme « `init` ».

Les particularités de la station sont regroupées dans le script « `/etc/rc.local` » appelé par « `/etc/rc` ».

Ce mécanisme tombe en désuétude car trop fouillis.
FreeBSD 5.x est en voie d'adoption du mécanisme des scripts de démarrage de System-V.

C'est un mode particulier de démarrage de la machine UNIX :

- la machine UNIX ne fonctionne pas encore en mode multi utilisateurs
- les fonctionnalités réseau ne sont pas encore actives

Utilité : intervenir sur la machine UNIX soit pour réparer un problème soit pour installer une partie logicielle qu'il est impossible de faire ultérieurement (changement de librairie dynamique par exemple).

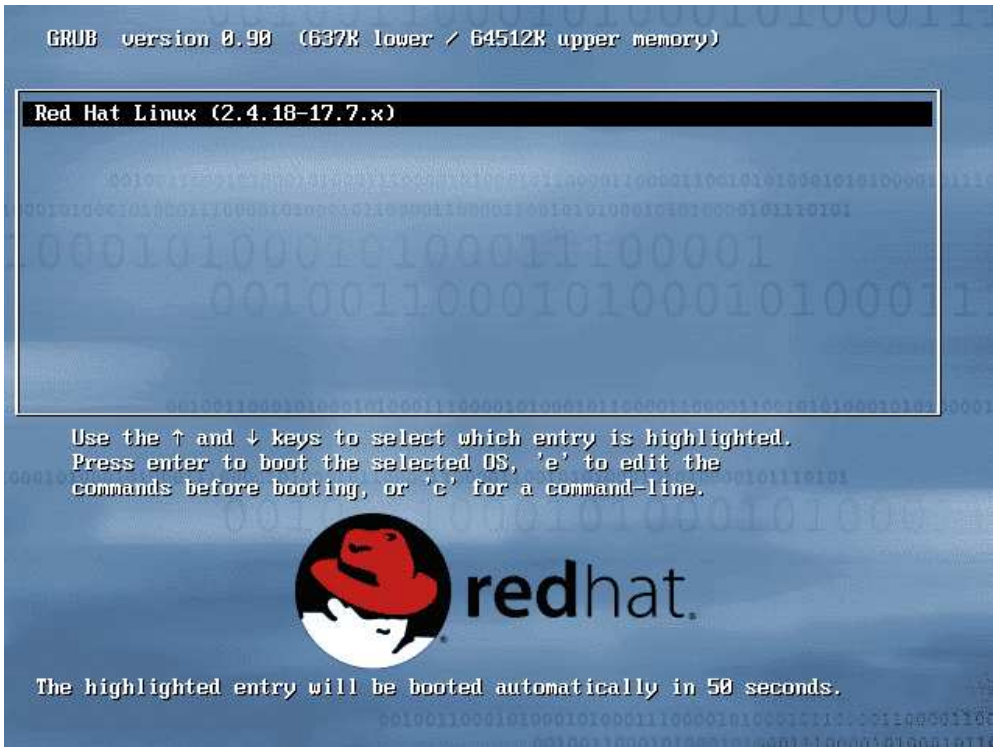
Pour booter en single user, il faut le préciser au niveau du moniteur de la machine.

On passe du mode mono utilisateur au mode multi utilisateur via « `exit` » qui fait se terminer le shell root du mode single user.





Exemple : boot en single user sur LINUX avec GRUB




GRUB version 0.90 (637K lower / 64512K upper memory)
root (hd0,0)
kernel /boot/vmlinuz-2.4.18-17.7.x ro root=/dev/hda1
Use the ↑ and ↓ keys to select which entry is highlighted.
Press 'b' to boot, 'e' to edit the selected command in the
boot sequence, 'c' for a command-line, 'o' to open a new line
after ('O' for before) the selected line, 'd' to remove the
selected line, or escape to go back to the main menu.
redhat.






[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ESC at any time exits. ]
grub edit> kernel /boot/vmlinuz-2.4.18-17.7.x ro root=/dev/hda1 single
redhat.







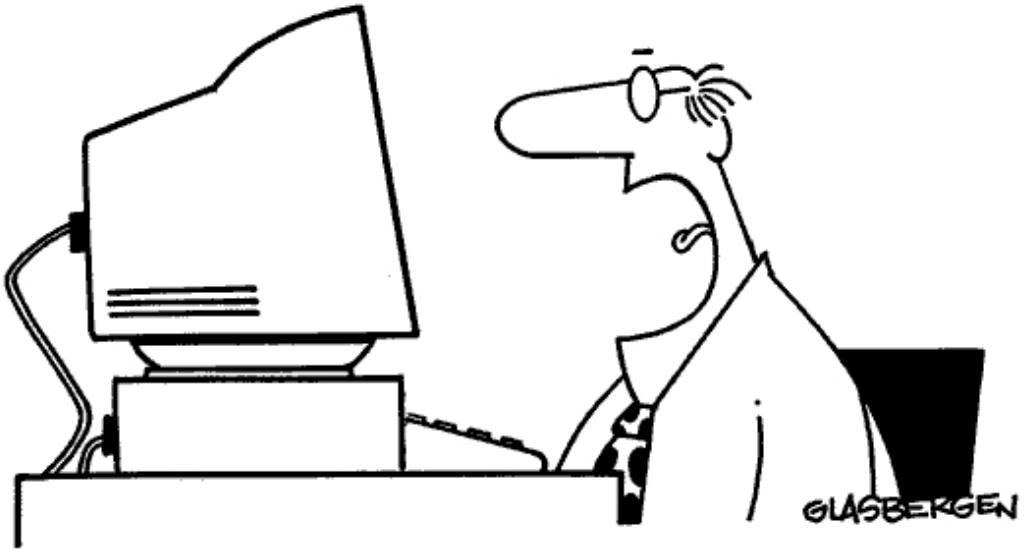

Exemple : boot en single user sur SOLARIS
Taper «`boot -s`»
Il faudra entrer le mot de passe de l'utilisateur `root` pour entrer en mode mono utilisateur sinon on restera en mode boot multi utilisateur.

Sur Sun, il faudra certainement régler le type de terminal (pour pouvoir faire «`vi`» par exemple) :

```
TERM=sun
export TERM
```

◇ Humour :

Copyright 2002 by Randy Glasbergen. www.glasbergen.com




**"I don't have to wait for my television to boot up!
I don't have to wait for my stereo to boot up!
I don't have to wait for my microwave to boot up!"**

Chapitre 8

# *Arrêt d'une station UNIX*





Théorie : il faut suivre une procédure d'arrêt du fait de l'aspect multi-utilisateurs :

1. Prévenir les utilisateurs
2. Signaler aux processus de se terminer proprement
3. Passer en single user
4. Flusher les disques avant d'éteindre (via la commande « `sync` »).

La pratique pourra être différente...

(en anglais *Advanced Power Management*)

Date de 1996 ; version 1.2 ; consortium Intel, Microsoft

Principalement pour hardware de type PC

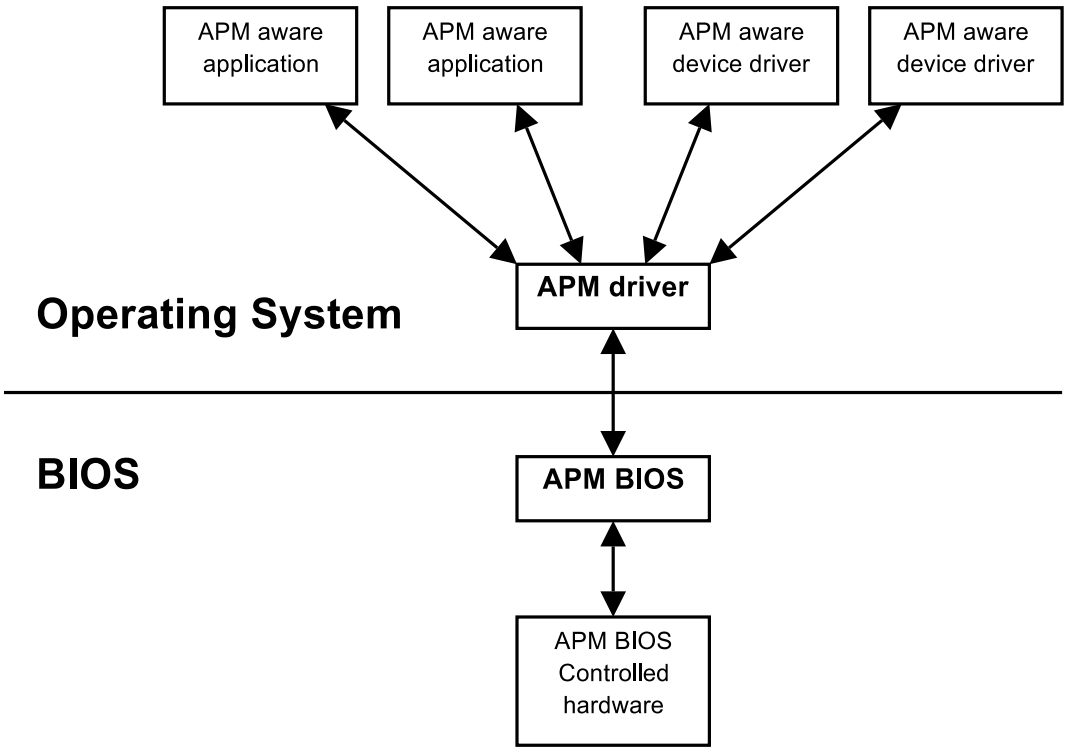
Obsolète ; remplacé par ACPI, cf chapitre suivant

**Un UNIX intégrant APM pourra s'éteindre.**





Principe :



(en anglais *Advanced Configuration and Power Interface*)

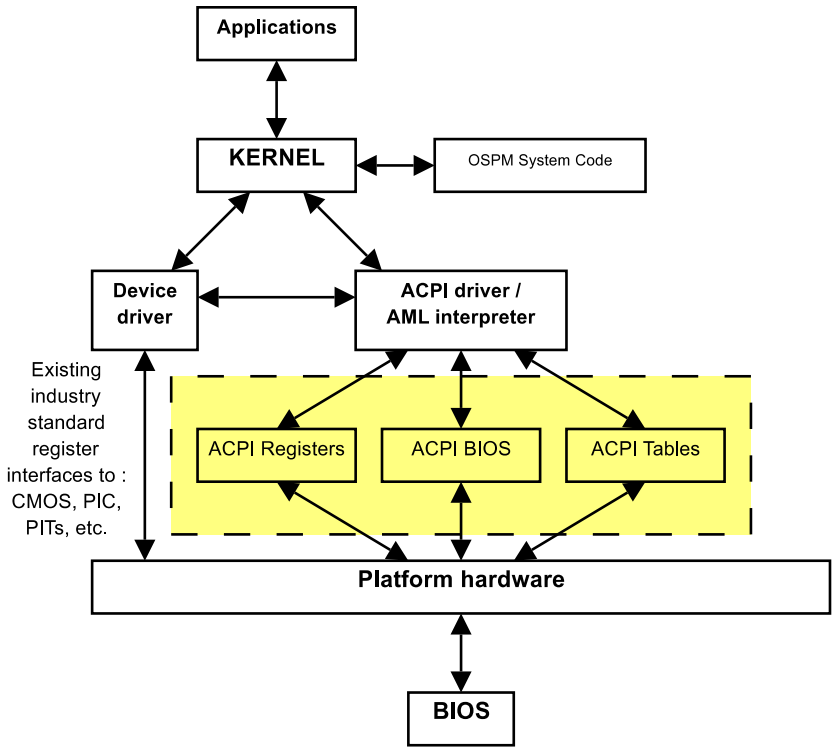
Date de 1999 ; version 2.0c ; consortium Intel, Toshiba, Microsoft

**Un UNIX intégrant APCI pourra s'éteindre.**



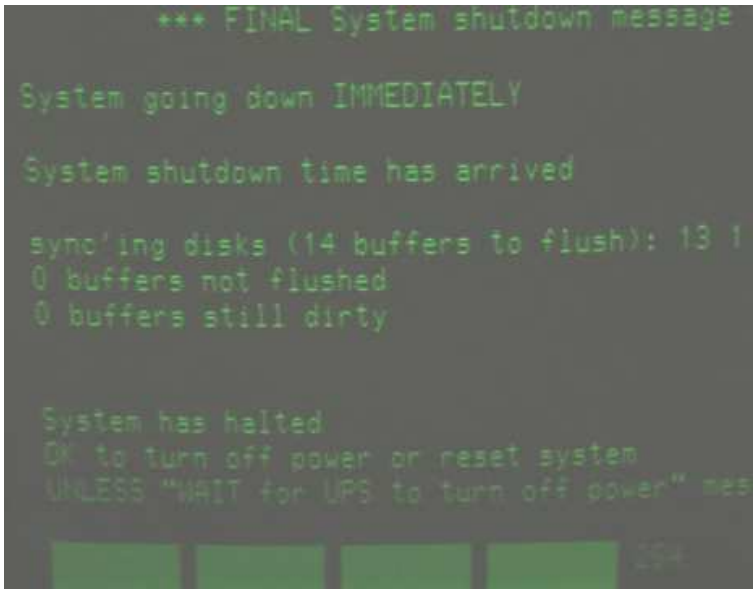


Principe radicalement différent de APM : **tout se passe au niveau du système d'exploitation**



Syntaxe : `halt`

Réalise un « `sync` » avant d'arrêter les processeurs et de rendre la main au mode moniteur en général.








Syntaxe : **`fasthalt`**

Disponible sur SUN SOLARIS et les UNIX BSD.

La commande créée un fichier « `/fastboot` » évitant de réaliser un « `fsck` » (voir page 502) au démarrage de la machine.

```
#! /bin/sh
# @(#)fasthalt.sh 1.6 88/02/07 SMI; from UCB 4.2
PATH=/bin:/usr/bin:/usr/etc:$PATH
export PATH
cp /dev/null /fastboot
halt $*
```

Syntaxe : **poweroff**

Disponible sur SUN SOLARIS, LINUX Mandriva 2006 et RHEL (machines virtuelles de TP).

Commande de plus en plus répandue.

Arrêt électrique de la machine.

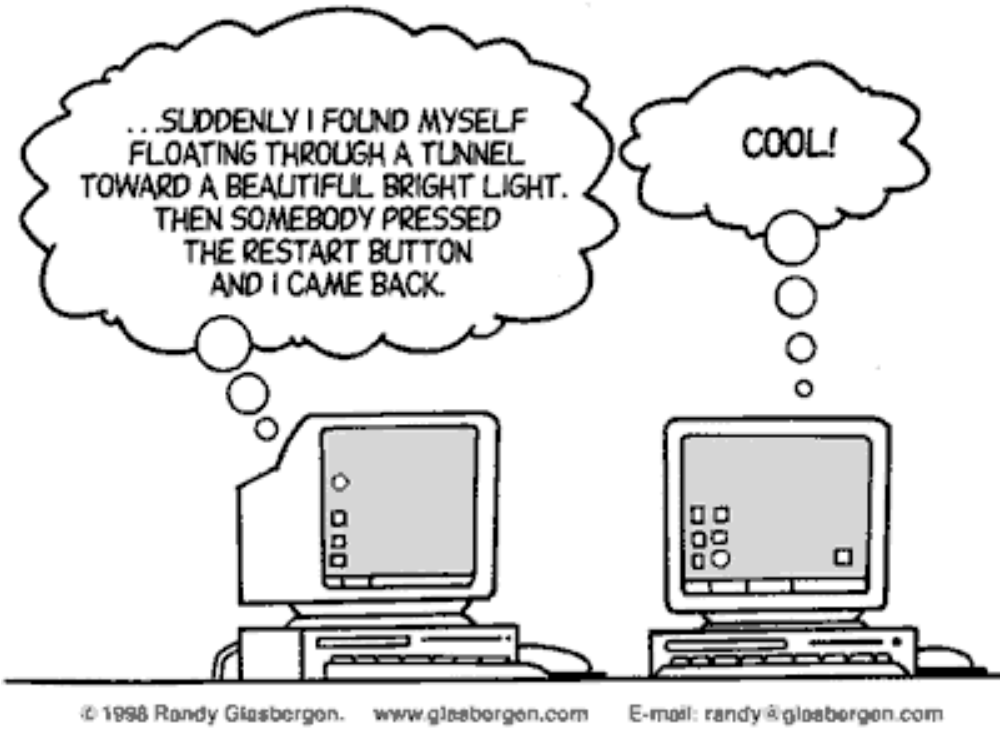




Syntaxe : **reboot**

Provoque un redémarrage du système.

Humour :






# Chapitre 8 • Arrêt d'une station UNIX

## §8.8 • Commande de redémarrage : fastboot

Syntaxe : **fastboot**

Disponible sur SUN SOLARIS et les UNIX BSD.

La commande créée un fichier « /fastboot » évitant de réaliser un « fsck » (voir page 502) au démarrage de la machine.

```
#! /bin/sh
# @(#)fastboot.sh 1.7 88/02/07 SMI; from UCB 4.2
PATH=/bin:/usr/bin:/usr/etc:$PATH
export PATH
cp /dev/null /fastboot
reboot "$@"
```

Syntaxe : **shutdown [options] heure [message]**

Envoi du signal « SIGTERM » aux processus.

Plusieurs syntaxes pour indiquer l'heure de redémarrage :

1. « `hh:mm` » : heure à laquelle l'opération est programmée
2. « `+min` » : nombre de minutes avant de redémarrer
3. « `now` » : redémarrage immédiat





Formes les plus utilisées :

- Pour redémarrer la machine : « `shutdown -r now` » (option « `-r` » comme *reboot*)
- Pour arrêter la machine : « `shutdown -h now` » (option « `-h` » comme *halt*)
- Pour arrêter la machine :
  - « `shutdown -p now` » (FreeBSD)
  - « `shutdown -P now` » (LINUX Mandriva 2006)

  (options « `-p` » ou « `-P` » comme *poweroff*)

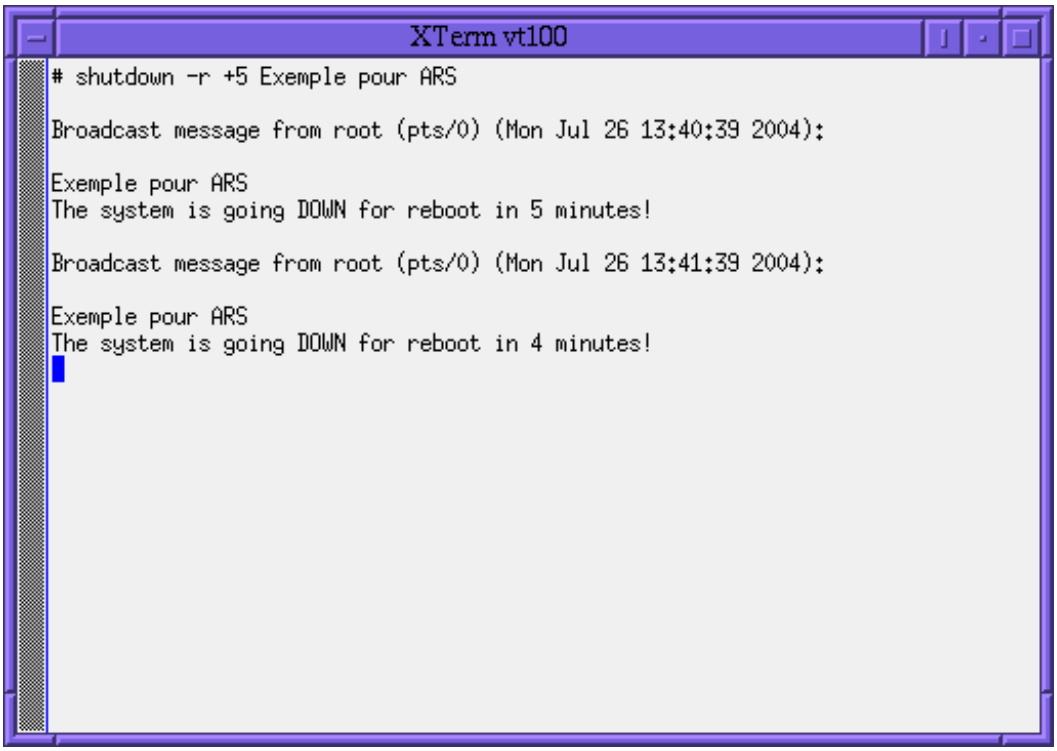
Commande de reboot dans 5 minutes :



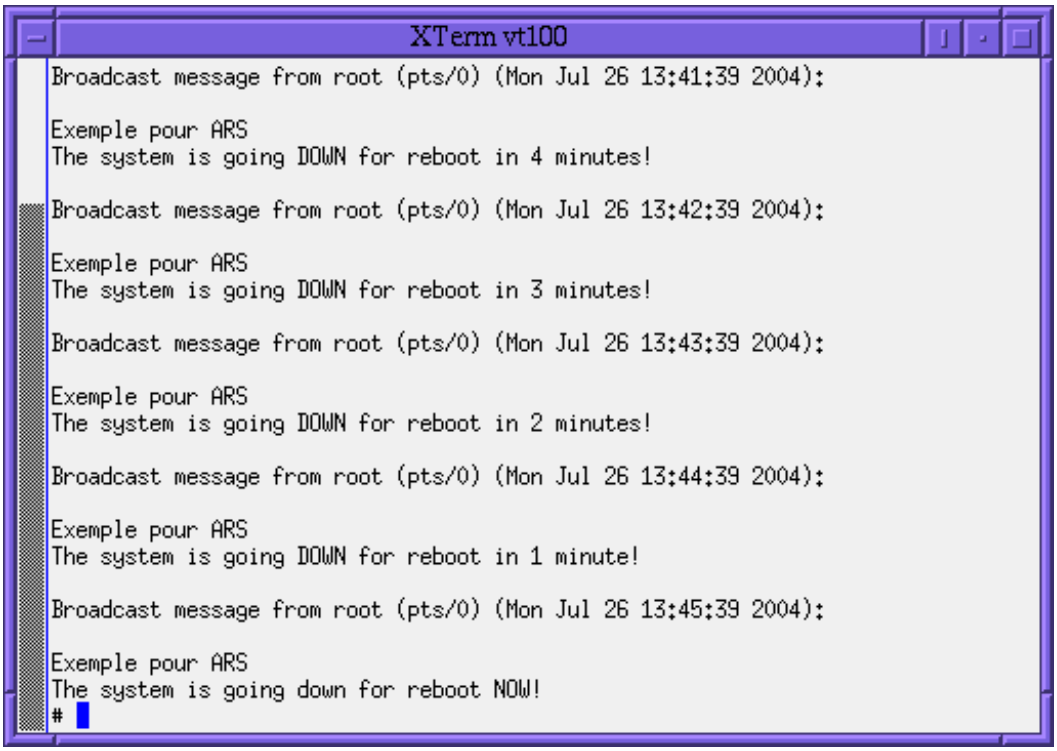





Annonce régulièrement envoyée pour rappeler le reboot :



Moment du reboot arrivé : la machine ne sera plus joignable le temps de son redémarrage :








◇ Ctrl-Alt-Del sur LINUX

Sur LINUX, on a la possibilité de redémarrer la machine via la séquence de touches Ctrl-Alt-Del.
Le réglage se fait au niveau de « `/etc/inittab` » :

```
...
ca::ctraltdel:/sbin/shutdown -t3 -r now
...
```

Si l'option « `-a` » est présente alors le fichier « `/etc/shutdown.allow` » permettra de vérifier si la personne peut faire Ctrl-Alt-Del.

PID 1 ≡ « `init` »

Donc « `kill -TERM 1` » provoque le passage du système du mode multi-utilisateurs en mode mono utilisateur.





Onduleurs = batteries = UPS (en anglais *Uninterruptible Power System*)





Quelques marques connues (liste non exhaustive) :

- APC / Merlin Gérin (même société en fait)
- SAEI
- ...








CAUTION: This equipment contains potentially hazardous voltages. The output may be energized when the unit is not connected to an AC power outlet. Do not attempt to disassemble the unit. Except for the battery, the unit contains no user serviceable parts. Repairs are performed only by factory trained service personnel.
BATTERY CONNECTOR
OUTPUT: 220-240V 50/60 Hz
750 MODEL: 750VA 600W 3.4A MAX
1000 MODEL: 1000VA 600W 4.5A MAX
USB PORT
OVERLOAD PROTECTOR
24 VDC
FOR USE WITH APC 24VDC APPROVED BATTERY PACKS
TVSS GND
220-240V~50/60 Hz
7A MAX

## Cable de raccordement : C13 - C14







## Cable de raccordement : C19 - C14



## Cable de raccordement : C19 - C20







## Cable de raccordement : C19 - CEE/7 SCHUKO



## Cable de raccordement : C19 - IEC309 16 A







## Cable de raccordement : C20 - IEC309 16 A



Principe d'un onduleur sur LINUX :

1. Le programme gérant l'onduleur envoie le signal « `SIGPWR` » à « `init` ».
2. Le programme « `init` » regarde le fichier « `/etc/powerstatus` » :
   - Si le fichier « `/etc/powerstatus` » contient « `FAIL` » alors on exécute les lignes « `powerfail` » et « `powerwait` » de « `/etc/inittab` ».
   - Si le fichier « `/etc/powerstatus` » contient « `LOW` » alors on exécute la ligne « `powerfailnow` » de « `/etc/inittab` ».
   - Si le fichier « `/etc/powerstatus` » contient « `OK` » alors on exécute la ligne « `powerokwait` » de « `/etc/inittab` ».





Si le fichier « `/etc/nologin` » existe, alors les connexions sont interdites et le contenu du fichier sert de message expliquant l'interdiction.

Par exemple, si « `/etc/nologin` » contient :

```
% cat /etc/nologin
Exemple pour ARS.
Interdiction de se connecter.
```

alors on verra :

```
% ssh redhat9.example.com
Exemple pour ARS.
Interdiction de se connecter.
Connection to redhat9.example.com closed.
```

Les opérations de redémarrage ou arrêt laissent des traces visibles par la commande « last » :

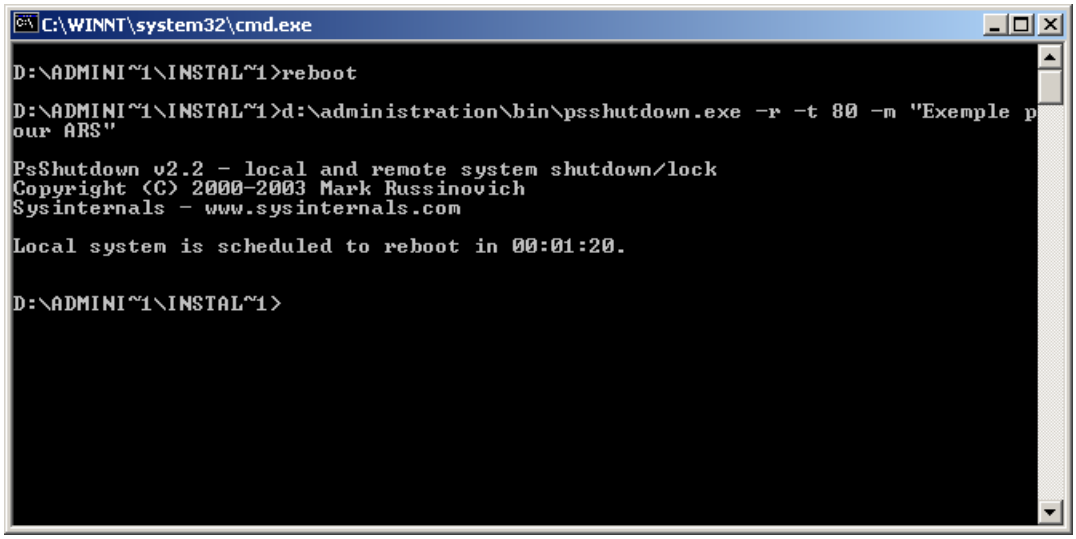
```
% last
...
besancon  ftp          dell.example.org Sun Mar 14 14:54 - 14:56
besancon  console                       Sun Mar  7 22:30   still l
reboot    system boot                   Sun Mar  7 22:29
besancon  console                       Sun Mar  7 21:57 - down
reboot    system boot                   Sun Mar  7 21:53
besancon  ftp          dell.example.org Tue Mar  2 00:48 - 00:49
besancon  console                       Sat Feb 28 23:24 - 21:43
root      console                       Sat Feb 28 23:23 - 23:24
reboot    system boot                   Sat Feb 28 23:22
besancon  ftp          dell.example.org Tue Feb 24 23:20 - 23:21
besancon  ftp          dell.example.org Mon Feb 23 00:28 - 00:28
...
```



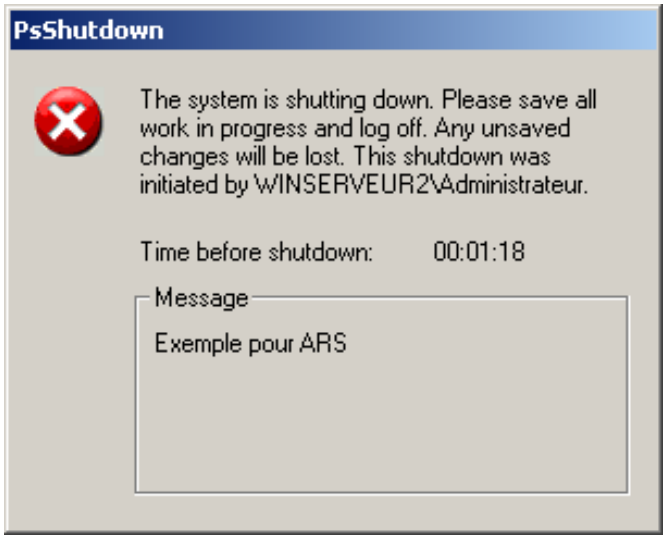


Comment programmer le redémerrage ou l'arrêt de machines Windows ? Plusieurs programmes disponibles :

- programme shutdown.exe du Resource Kit Windows 2000
- programme psshutdown.exe de http://www.sysinternals.com



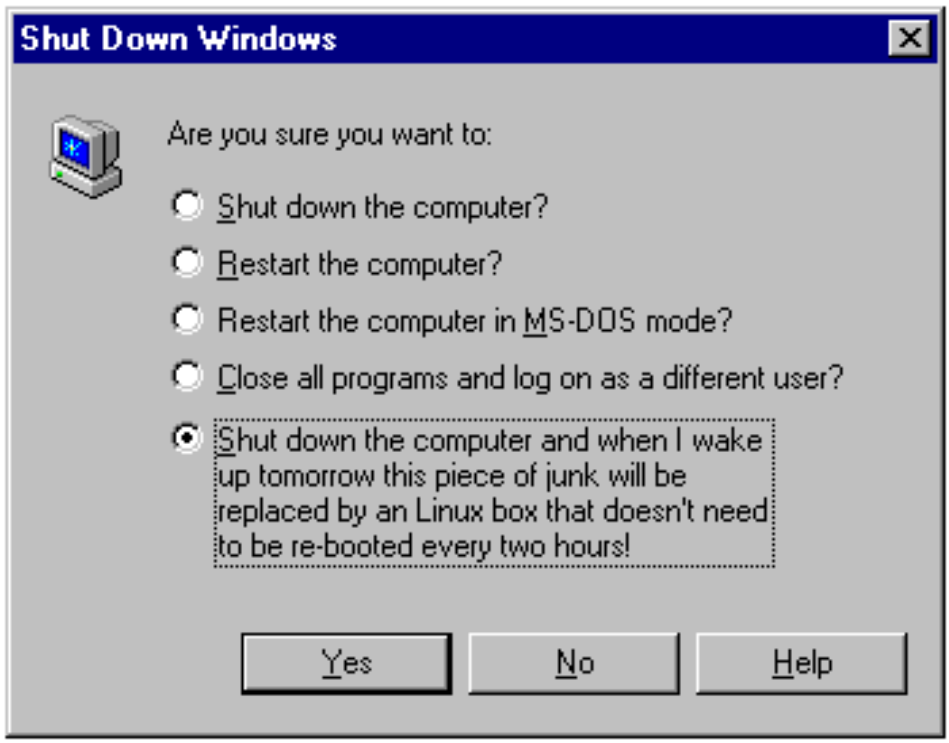
La commande `psshutdown.exe` affiche alors :








◇ Humour :



Chapitre 9

# *Le noyau UNIX et les périphériques*





Le noyau :

- un ensemble de fonctions, de procédures, de structures de données réalisant un certain nombre de tâches
- un exécutable lancé pendant le boot
- un exécutable contenu dans « / » (en général)

Les responsabilités du noyau :

- fournir une interface d'utilisation et de gestion efficace
- gérer le CPU
- savoir piloter les différents types de périphériques
- gérer les processus (création des processus, ordonnancement, accounting)
- gérer la mémoire (allocation aux processus, swap, paging)
- gérer le système de fichiers (accès locaux, accès réseau, quotas)
- gérer les autres ressources (entrées/sorties, réseau etc.)

Les tâches de l'administrateur concernant le noyau :

- assurer que le noyau du système fonctionne correctement
- assurer que le noyau est configuré pour représenter correctement le système
- modifier le noyau
- compiler le noyau



## Chapitre 9 • Le noyau UNIX et les périphériques
### §9.2 • Assurer la validité du noyau

Le boot de la machine UNIX se compose des étapes :

1. Mode moniteur de la machine
2. Chargeur primaire
3. Chargeur secondaire
4. Chargement du noyau – création du processus `init`
5. Scripts de démarrage lancés par `init`

Si le noyau est incorrect (raisons diverses), le boot ne peut pas se réaliser.

⇒ **Toujours avoir en réserve un noyau que l'on sait être correct.**

**En pratique, avoir au moins le noyau de départ fourni par le constructeur.**

Copie d'écran d'un kernel panic :

```
Serial: 8250/16550 driver $Revision: 1.90 $ 52 ports, IRQ sharing enabled
ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
ttyS1 at I/O 0x2f8 (irq = 3) is a 16550A
ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
ttyS1 at I/O 0x2f8 (irq = 3) is a 16550A
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 10000K size 1024 blocksize
mice: PS/2 mouse device common for all mice
input: AT Translated Set 2 keyboard on isa0060/serio0
NET: Registered protocol family 2
IP: routing cache hash table of 1024 buckets, 8Kbytes
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
NET: Registered protocol family 1
NET: Registered protocol family 17
ACPI wakeup devices:
 USB
ACPI: (supports S0 S1 S5)
RAMDISK: Compressed image found at block 0
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(3,9)
_
```





Copie d'écran d'un kernel panic :

```
    ide1: BM-DMA at 0xc008-0xc00f, BIOS settings: hdc:pio, hdd:pio
ne2k-pci.c:v1.03 9/22/2003 D. Becker/P. Gortmaker
  http://www.scyld.com/network/ne2k-pci.html
hda: QEMU HARDDISK, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hdc: QEMU CD-ROM, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
ACPI: PCI Interrupt Link [LNKC] enabled at IRQ 10
ACPI: PCI Interrupt 0000:00:03.0[A] -> Link [LNKC] -> GSI 10 (level, low) -> IRQ
 10
eth0: RealTek RTL-8029 found at 0xc100, IRQ 10, 52:54:00:12:34:56.
hda: max request size: 512KiB
hda: 180224 sectors (92 MB) w/256KiB Cache, CHS=178/255/63, (U)DMA
hda: set_multmode: status=0x41 { DriveReady Error }
hda: set_multmode: error=0x04 { DriveStatusError }
ide: failed opcode was: 0xef
hda: cache flushes supported
 hda: hda1
hdc: ATAPI 4X CD-ROM drive, 512kB Cache, (U)DMA
Uniform CD-ROM driver Revision: 3.20
Done.
Begin: Mounting root file system... ...
/init: /init: 151: Syntax error:  0xforce=panic
Kernel panic - not syncing: Attempted to kill init!
 _
```

But : refléter l'état actuel du système.
En particulier :

- De quels drivers a-t-on besoin ?
  Eliminer les drivers non nécessaires ⇒ gain de place mémoire pour d'autres choses.
- Configurer les tailles de diverses structures mémoires (appel à « `smit` » sur AIX, « `sam` » sur HP-UX, « `ndd` » sur Solaris).

La configuration du noyau est propre à chaque constructeur.
Chez certains constructeurs, c'est plus ou moins interactif (LINUX, DigitalUnix).





On peut modifier le noyau pour :

- incorporer des correctifs (alias « patches ») ;
- incorporer de nouvelles fonctionnalités (par exemple ajouter un pilote de graveur de CD-ROM) ;
  Ne pas oublier de générer d'éventuels fichiers spéciaux après incorporation dans un noyau d'un nouveau driver.
- simplement l'informer de la présence de nouveaux périphériques.

## Exemple de patch constructeur pour SunOS 4.1.4 :

```
Patch Installation Instructions:

   Become superuser, cp tcx.o to /usr/sys/sun4m/OBJ after first saving
   the old version aside as a precaution.
    % su
    # mv /usr/sys/sun4m/OBJ/tcx.o /usr/sys/sun4m/OBJ/tcx.o.FCS
    # cp sun4m/tcx.o /usr/sys/sun4m/OBJ

   Rebuild the kernel.  Refer to the Systems and Network administrations
   manual on how to configure a kernel.
    # cd /usr/sys/sun4m/conf
    # /etc/config GENERIC
    # cd ../GENERIC
    # make

   Preserve the original kernel, copy the new kernel to / and reboot.
    # mv /vmunix /vmunix-
    # cp vmunix /
    # reboot
```





## Exemple de patch constructeur pour Solaris 2.x :

```
Patch Installation Instructions:
--------------------------------

For Solaris 2.0-2.6 releases, refer to the Install.info file and/or
the README within the patch for instructions on using the generic
'installpatch' and 'backoutpatch' scripts provided with each patch.

For Solaris 7-8 releases, refer to the man pages for instructions
on using 'patchadd' and 'patchrm' scripts provided with Solaris.
Any other special or non-generic installation instructions should be
described below as special instructions.  The following example
installs a patch to a standalone machine:

       example# patchadd /var/spool/patch/104945-02

The following example removes a patch from a standalone system:

       example# patchrm 104945-02

For additional examples please see the appropriate man pages.
```

## C'est de plus en plus automatisé. . .

C'est un mécanisme propre à chaque constructeur.
Bien sûr on n'oubliera pas la sauvegarde du noyau actuel pour pouvoir faire marche arrière.

A noter que la mise en place d'un nouveau noyau sans rebooter peut conduire à l'arrêt du fonctionnement de certains programmes, par exemple « `ps` » :

```
# ps -ax
ps: /dev/mem: error on ktextseg: Bad address
ps: could not read kernel VM
```





Différences entre les systèmes à sources publiques et les systèmes constructeurs :

- la compilation d'un noyau implique la compilation de tous ses fichiers sources. La compilation d'un noyau linux est donc une procédure longue (de l'ordre de 6 minutes sur un bi-pentium Pro 200 MHz pour Linux).
- les UNIX constructeurs ne fournissent pas les fichiers sources mais par contre ils fournissent les fichier « .o » résultats de la compilation séparée intermédiaire.
  La compilation d'un noyau UNIX constructeur revient à :
  - compilation de quelques fichiers « .c » contenant l'indication des pilotes installés, des tailles mémoirede certaines structures
  - édition de liens des « .o » constructeurs et des « .o » locaux

Les sources du noyau LINUX sont sur « `http://www.kernel.org` ».

Traditionnellement, on désarchive les sources dans « `cd /usr/src/linux-x.y.z` ».





1. `cd /usr/src/linux-x.y.z`
2. `make mrproper` pour faire le ménage complet
3. `make config` ou `make menuconfig` ou `make xconfig`
4. `make checkconfig` facultativement
5. `make dep` pour construire les fichiers de dépendance pour la compilation C
6. `make clean` (à faire quasi impérativement)
7. `make bzImage` pour construire une image compressée
8. `make modules` compilera les modules dynamiques du noyau
9. `cp bzImage /boot`
10. `cp System.map /boot`
11. `make modules_install`

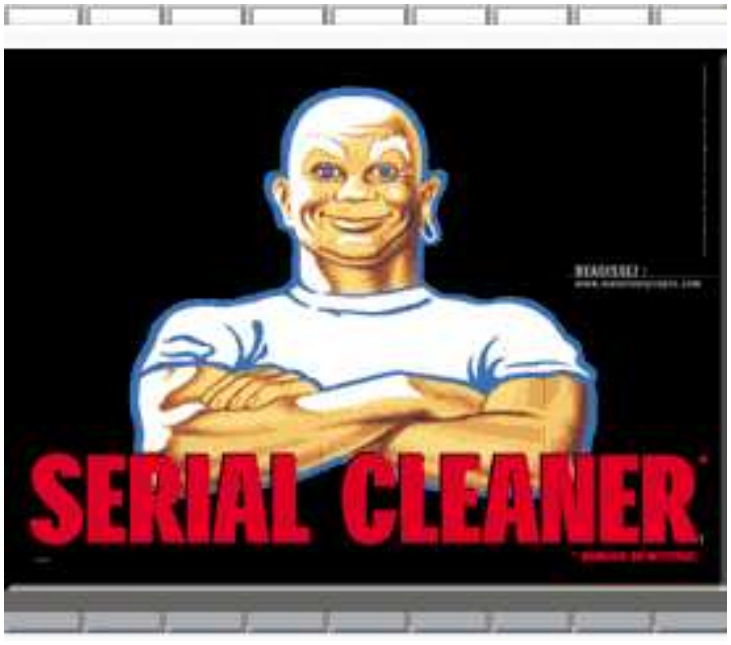
12 Editer `/etc/lilo.conf` pour refléter l'ajout d'un nouveau noyau et relancer la commande `lilo` (si utilisation de LILO et pas de GRUB)

13 Rebooter.





« `make mrproper` » parce que...



« `make mrproper` » parce que...






## Chapitre 9 • Le noyau UNIX et les périphériques
### §9.9 • Compilation du noyau FreeBSD

(exemple pour plateforme i386)

1. `cd /usr/src/sys/i386/conf` pour se placer à l'endroit des configurations des noyaux, y compris la configuration par défaut
2. `cp GENERIC MYKERNEL` pour partir de la configuration par défaut
3. `vi MYKERNEL` pour éditer le fichier `MYKERNEL` pour y configurer les aspects utiles du noyau
4. `/usr/bin/config MYKERNEL` pour générer les fichiers C résultats de la configuration
5. `cd ../compile/MYKERNEL` pour se placer dans le répertoire de compilation des sources
6. `make depend`
7. `make` pour compiler le noyau
8. `make install` pour installer le noyau

Inutile de recompiler le noyau SOLARIS car :

- noyau modulaire à 100%
- variables du noyau modifiables via la commande « `ndd` » ou le fichier de configuration « `/etc/system` »

(l'idée à l'origine est la constatation que seuls 10% des clients de SUN sont des informaticiens ⇒ simplifier les choses)





Le noyau UNIX traditionnel est monolithique. Il est malaisé d'y ajouter des drivers, d'y modifier des variables.

La mode est aux modules chargeables dynamiquement dans le noyau. Ils ajoutent alors les nouvelles fonctionnalités au noyau pendant qu'il tourne sans nécessiter d'arrêt.

## Exemple sur Linux :

```
% lsmod
Module                  Size  Used by
soundcore               2628   0 (autoclean) (unused)
vfat                    9276   0 (autoclean) (unused)
fat                    30400   0 (autoclean) [vfat]
nfs                    28896   3 (autoclean)
lockd                  31496   1 (autoclean) [nfs]
sunrpc                 53028   1 (autoclean) [nfs lockd]
ne                      6668   1 (autoclean)
8390                    6072   0 (autoclean) [ne]
```





## Exemple sur Solaris où le noyau est complétement modulaire :

```
% modinfo
 Id Loadaddr   Size Info Rev Module Name
  5 600ba000   3b30    1   1  specfs (filesystem for specfs)
  7 600e1000   2bc8    1   1  TS (time sharing sched class)
  8 600b0a88    4a4    -   1  TS_DPTBL (Time sharing dispatch table)
  9 600f0000  23538    2   1  ufs (filesystem for ufs)
 10 6012a000   dc4b 226   1  rpcmod (RPC syscall)
 10 6012a000   dc4b    1   1  rpcmod (rpc interface str mod)
 11 60144000  2752b    0   1  ip (IP Streams module)
 11 60144000  2752b    3   1  ip (IP Streams device)
 12 600e5550   11f3    1   1  rootnex (sun4u root nexus)
 13 60037a48    188   57   1  options (options driver)
 14 600e4010   14e8   12   1  sad (Streams Administrative driver's)
 15 600b0548    538    2   1  pseudo (nexus driver for 'pseudo')
 16 6018c000   f059   32   1  sd (SCSI Disk Driver 1.255)
 17 601a2000   5f26    -   1  scsi (SCSI Bus Utility Routines)
 18 601b8000   ca68   61   1  esp (ESP SCSI HBA Driver 1.250)
...
```

Sur LINUX, certaines variables du noyau peuvent être modifiées au boot de la machine ou après sans avoir à recompiler le noyau pour faire cela. Pour cela, utiliser la commande « sysctl »

Pour connaitre les valeurs de certains réglages du noyau, faire « sysctl -a » :

```
% /sbin/sysctl -a | grep ip_forward
net.ipv4.ip_forward = 0
```

Pour régler une variable, utiliser « sysctl -w variable=valeur » :

```
# sysctl -w net.ipv4.ip_forward="1"
net.ipv4.ip_forward = 1
```





Le fichier « /etc/sysctl.conf » peut contenir des réglages à faire au démarrage de la machine. Syntaxe du type « variable=valeur ».

Par exemple sur « serveur.formation.jussieu.fr » :

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
# Disables the magic-sysrq key
kernel.sysrq = 0
```

Sur FreeBSD, certaines variables du noyau peuvent être modifiées au boot de la machine ou après sans avoir à recompiler le noyau pour faire cela. Pour cela, utiliser la commande « /sbin/sysctl » :

Pour connaitre les valeurs de certains réglages du noyau, faire « sysctl -a » :

```
% sysctl -a | grep forwarding
net.inet.ip.forwarding: 0
net.inet.ip.fastforwarding: 0
net.inet6.ip6.forwarding: 0
```

Pour régler une variable, utiliser « sysctl variable=valeur » :

```
# sysctl net.inet.ip.forwarding="1"
net.inet.ip.forwarding: 0 -> 1
```





Le fichier « /etc/sysctl.conf » peut contenir des réglages à faire au démarrage de la machine. Syntaxe du type « variable=valeur ».

Par exemple :

```
net.inet.ip.forwarding=1
```

(en anglais *network device driver*)
Sur SOLARIS, certaines variables du noyau peuvent être modifiées au boot de la machine ou après sans avoir à recompiler le noyau pour faire cela.
Pas de commande « `sysctl` » à la LINUX ou FreeBSD, par contre...
Pour les paramètres réseau, utiliser la commande « `/usr/sbin/ndd` ».
Par exemple :

```
# ndd /dev/ip  | grep forwarding
ip_forwarding                   (read and write)
ip6_forwarding                  (read and write)
lo0:ip_forwarding               (read and write)
eri0:ip_forwarding              (read and write)

# ndd /dev/ip ip_forwarding
0
```

Pour régler un certain paramètre, on procéde ainsi :

```
# ndd -set /dev/ip ip_forwarding 1
```





Au niveau du fichier « `/etc/system` », on peut régler certains paramètres du noyau qui seront pris en compte au démarrage.

Par exemple pour rendre les piles des processus non exécutables :

```
set noexec_user_stack = 1
set noexec_user_stack_log = 1
```

Pour connaitre comment paramètrer certains aspects du noyau SOLARIS, se reporter au manuel *Solaris Tunable Parameters Reference Manual* :
`http://docs-pdf.sun.com/817-1759/817-1759.pdf`

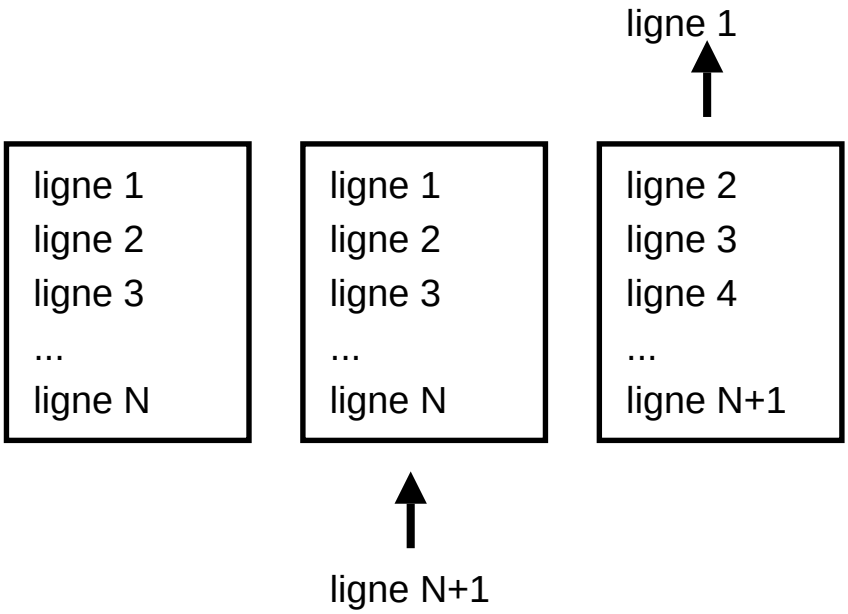
(en anglais *Diagnostic Messages*)
Syntaxe : `dmesg`

```
% dmesg
Intel MultiProcessor Specification v1.4
    Virtual Wire compatibility mode.
OEM ID: OEM00000 Product ID: PROD00000000 APIC at: 0xFEE00000
Processor #0 Pentium(tm) Pro APIC version 17
Processor #1 Pentium(tm) Pro APIC version 17
I/O APIC #2 Version 17 at 0xFEC00000.
Enabling APIC mode: Flat. Using 1 I/O APICs
Processors: 2
Kernel command line: ro root=/dev/hdd3
Initializing CPU#0
Detected 1002.276 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 1998.84 BogoMIPS
Memory: 1027216k/1048512k available (1294k kernel code, 17716k reserved,
1080k data, 144k init, 131008k highmem)
...
```





**ATTENTION** : la commande « `dmesg` » utilise un buffer mémoire de taille fixe si bien que les messages les plus vieux disparaissent au fur et à mesure que de nouveaux messages arrivent.



Pour sauvegarder le résultat de « dmesg » automatiquement après le boot : shell script de démarrage :

```
#!/bin/sh
# chkconfig: 35 99 01
# description: sauvegarde de la sortie de dmesg

case "$1" in
  start) dmesg > `date +/root/dmesg-%Y%m%d-%H%M`
         ;;
   stop) :
         ;;
esac
```





Point clé d'UNIX : **abstraction d'un périphérique sous la forme d'un objet**

⇒ tout se ramène à des fonctions du langage C d'accès à des fichiers (« open() », « write() », « close() », etc.)
⇒ interface **unifiée** de programmation

Les objets classiques du monde de l'utilisateur de base :

- fichiers « réguliers » (`test -f objet`)

```
-rw-r--r--  1 besancon  software  940 Sep 21 14:23 cours.tex
```

- directories (`test -d objet`)

```
drwxr-sr-x  17 besancon  software  1536 Sep 21 14:33 .
```

- liens symboliques (`test -L objet`)

```
lrwxrwxrwx  1 root  wheel  23 Jun 10 13:24 /etc/termcap@ -> \
/usr/share/misc/termcap
```





Des objets moins classiques que l'administrateur utilise occasionnellement :

- objets périphériques blocs (`test -b objet`) : ils font référence à des périphériques

```
brw-r-----  1 root  operator    4,   0 Jun 10 13:27 /dev/sd0a
```

- objets périphériques caractères (`test -c objet`) : ils font référence à des périphériques

```
crw-r-----  1 root  operator   13,   0 Jun 10 13:27 /dev/rsd0a
```

- pipes nommés ou *FIFO* (`test -p objet`)

```
prw-r--r--   1 besancon software        0 Sep 21 14:38 essai
```

- sockets (`test -S objet`)

```
srwxrwxrwx  1 root  bin  0 Sep  4 17:21 /tmp/.X11-unix/X0
```

Point commun entre tous ces types d'objets :

*effaçables par* « `rm` »





(en anglais *device*)

« `/dev` » est le lieu de dépôt des fichiers spéciaux.

**Les droits d'accès aux fichiers spéciaux dans /dev sont très importants.**
Pourquoi ?

Une précaution : conserver une trace écrite d'un « `ls -l` » de « `/dev` »

Quel est le contenu du répertoire « /dev » ?
Dans le cas de machines de type System-V (comme Solaris), « /dev » contient des sous répertoires :

| Répertoire | Contenu |
|---|---|
| /dev/rdsk | disques en mode caractère |
| /dev/dsk | disques en mode bloc |
| /dev/rmt | bandes magnétiques |
| /dev/pty | pseudo-terminaux esclaves |
| /dev/ptym | pseudo-terminaux maîtres |

(le « r » dans « rdsk » ou « rmt » signifie « raw »)

Les machines BSD mettent tout en vrac.





Les machines LINUX ont une position intermédiaire :

- moitié organisée
- moitié en vrac

```
# du -k /dev
0       /dev/pts
4       /dev/ataraid
32      /dev/cciss
4       /dev/compaq
4       /dev/cpu/0
4       /dev/cpu/1
...
4       /dev/cpu/9
68      /dev/cpu
4       /dev/dri
32      /dev/i2o
32      /dev/ida
4       /dev/inet
4       /dev/input
12      /dev/logicalco
4       /dev/net
4       /dev/raw
68      /dev/rd
4       /dev/scramdisk
0       /dev/shm
4       /dev/usb
4       /dev/video
4       /dev/video1394
4       /dev/snd
4       /dev/cpqhealth
436     /dev
```

Reconnaissable lors d'un « `ls -l` » :

```
brw-r----- 1 root operator  4, 0 Jun 10 13:27 /dev/sd0a
brw-r----- 1 root sys      32, 8 Jul  9  1999 /dev/dsk/c0t1d0s0
```

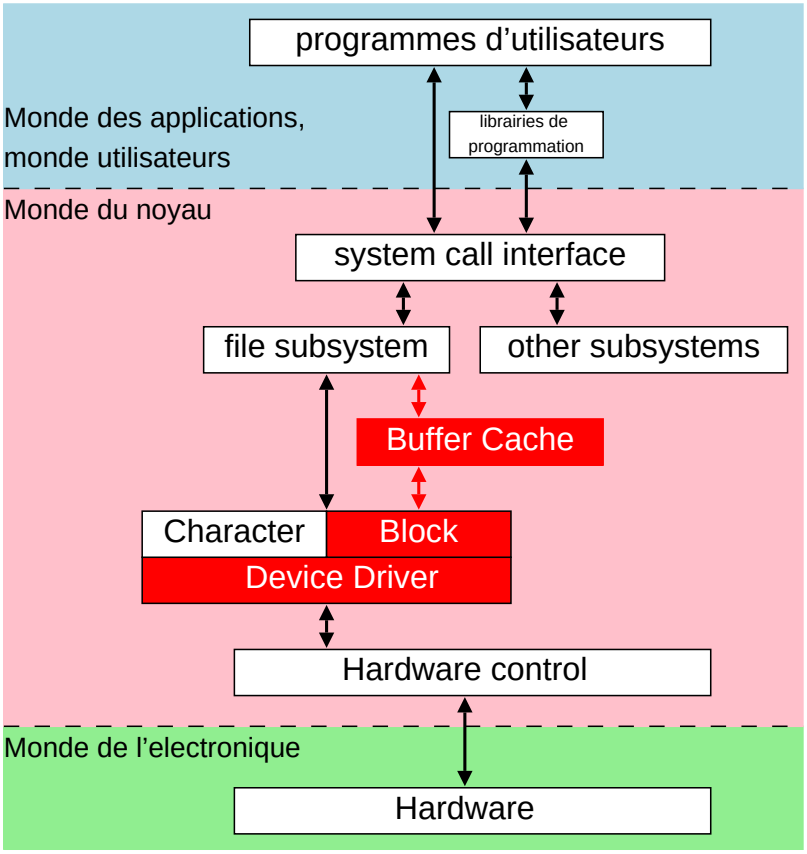




Ce mode concerne surtout les disques durs :

- la lecture d'un octet de données se traduit par la lecture d'un bloc complet conservé alors dans un peu de mémoire du noyau, le **buffer cache**.
- la lecture de données supplémentaires passe par la consultation du cache d'abord
- une écriture est asynchrone : les données sont écrites dans le buffer cache
- les données écrites s'ajoutent au buffer cache
- en cas de remplissage du cache, les données sont écrites sur disque et le cache se vide

Finalité du mode bloc : **minimaliser les entrées / sorties**
**Les transferts se font toujours par blocs de taille fixée.**








Reconnaissable lors d'un `ls -l` :

```
crw-r----- 1 root operator 13,  0 Jun 10 13:27 /dev/rsd0a
crw-r----- 1 root sys      32,  8 Jan  6  2000 /dev/rdsk/c0t1d0s0
```

Un périphérique en mode caractère accepte un transfert d'une quantité quelconque d'octets.
**Pas de transit par une zone cache intermédiaire.** ⇒ accès rapide

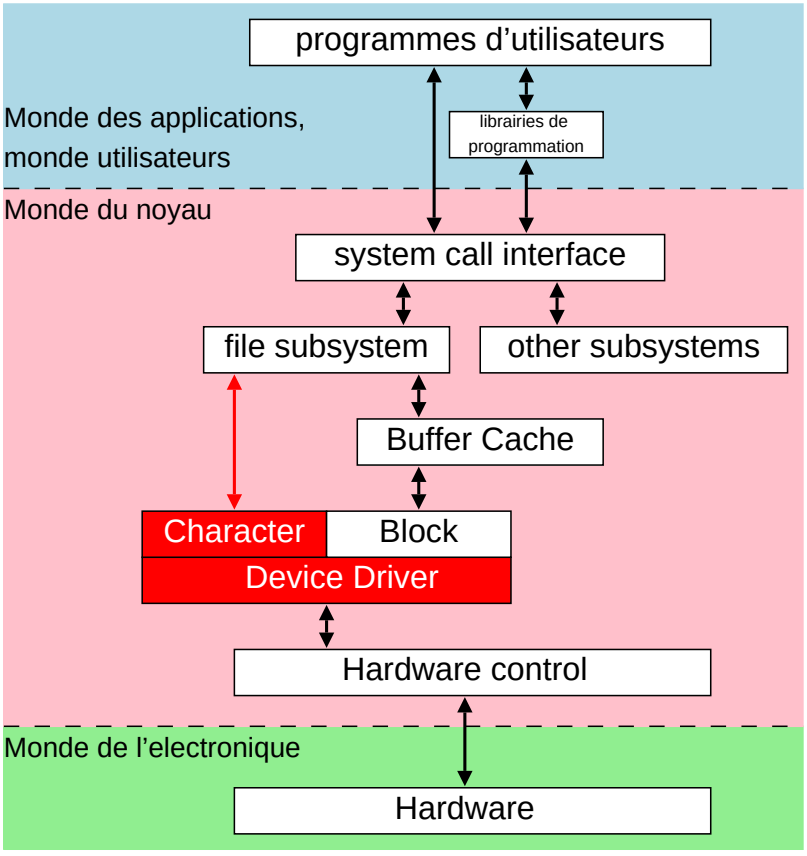
D'autres opérations sont permises selon le périphérique en mode caractère :

- formatage
- réglage de paramètres de liaison série
- autres opérations de bas niveau
- ...

Les périphériques en mode block ont souvent aussi une interface en mode caractères.







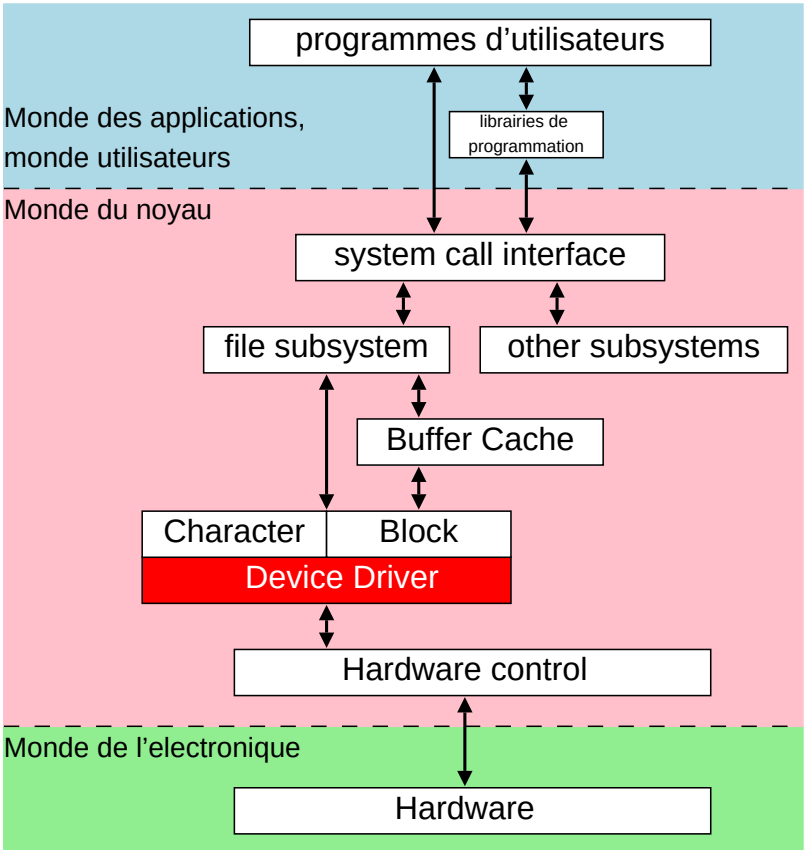
L'utilisation d'une pièce de hardware nécessite un pilote spécial (dit « device driver »).

⇒ un device driver par périphérique de type différent.

⇒ installation au niveau du noyau UNIX du device driver (donc reconfiguration en cas d'ajout de device) ; l'installation est spécifique à chaque UNIX.







L'association entre le périphérique et le noyau est assurée par un couple : (minor number, major number)

- major number
  Il identifie le pilote dans le noyau.
- minor number
  Il identifie le périphérique pour un certain driver.





(en anglais *make node*)

La commande pour créer les fichiers spéciaux est **mknod**.
Syntaxe : `mknod nom [c|b] major minor`

Qui connait les minor et les major numbers à mettre en paramètres ?
Ils ne s'inventent pas ! La réponse est dans le système !

Actuellement on distingue 2 approches différentes pour les générer :

- manuelle
- automatique

◇ Approche création statique des devices

On a une liste exhaustive des minors et majors pour les drivers de base.
Sur ces systèmes, le programme « /dev/MAKEDEV » suit cette liste et génére correctement le périphérique qu'on lui désigne comme par exemple *le disque SCSI 1*.

```
# cd /dev
# ls
MAKEDEV         MAKEDEV.local
# ./MAKEDEV std
# ls
MAKEDEV         fd              lkm             random          tty
MAKEDEV.local   io              mem             stderr          urandom
console         klog            null            stdin           zero
drum            kmem            pci             stdout
# ./MAKEDEV all
MAKEDEV         perfmon         rsd0s4          rwd3g           ttypb
MAKEDEV.local   psm0            rsd1            rwd3h           ttypc
apm             ptyp0           rsd1.ctl        rwd3s1          ttypd
bpf0            ptyp1           rsd1a           rwd3s2          ttype
...
```

Difficile à gérer lorsque beaucoup de périphériques. Gestion des périphériques amovibles ?
⇒ approche statique abandonnée par la plupart des UNIX





◇ Approche création dynamique des devices

Le système gère dynamiquement les devices au fur et à mesure qu'il rencontre des périphériques.
Cas de LINUX (kernel 2.4 ou 2.6), SOLARIS (version 9 ou 10), FREEBSD (version 5.x)

Historiquement deux approches sur LINUX :

- premier mécanisme dynamique apparu : **DevFS** (noyau 2.3)
  - http://www.atnf.csiro.au/people/rgooch/linux/docs/devfs.html
  - http://chl.tuxfamily.org/glmf/ et plus précisemment dans le numéro 25
    http://okki666.free.fr/docmaster/articles/linux086.htm
- abandon de DevFs au profit de **udev** (noyau 2.6)
  - http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
  - http://www.kroah.com/linux/talks/ols_2003_udev_talk/
  - http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html
  - http://www.linuxjournal.com/article/7316





Par exemple sur FreeBSD 5.x :

1. insertion d'une clef USB ⇒ le système détecte la clef comme le montrent les lignes de dmesg suivantes :

```
Jul 26 13:27:50 alpha kernel: umass0: M-Systems DiskOnKey, rev 1.10/2.00, adc
Jul 26 13:27:50 alpha kernel: umass0: Get Max Lun not supported (SHORT_XFER)
Jul 26 13:27:50 alpha kernel: da0 at umass-sim0 bus 0 target 0 lun 0
Jul 26 13:27:50 alpha kernel: da0: <M-Sys DiskOnKey 2.51> Removable Direct Ac
Jul 26 13:27:50 alpha kernel: da0: 1.000MB/s transfers
```

2. le système a créé les devices :

```
% ls -l /dev/*da0*
crw-rw----  1 root  operator    4,  34 Jul 20 10:16 /dev/da0
```

3. retrait de la clef USB : le système détecte que la clef n'est plus là :

```
Jul 26 13:29:49 alpha kernel: umass0: at uhub1 port 2 (addr 3) disconnected
Jul 26 13:29:49 alpha kernel: (da0:umass-sim0:0:0:0): lost device
Jul 26 13:29:49 alpha kernel: (da0:umass-sim0:0:0:0): removing device entry
Jul 26 13:29:49 alpha kernel: umass0: detached
```

4. suppression des devices associés à la clef

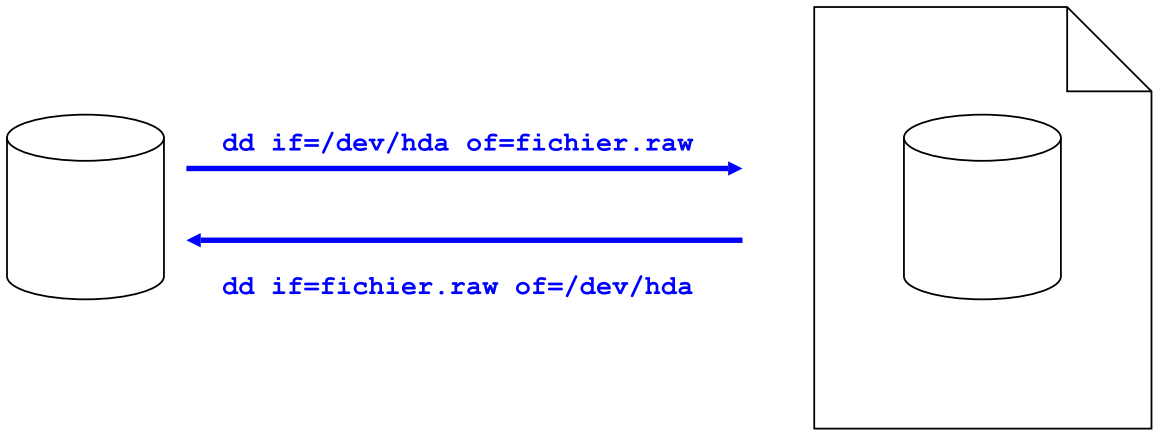
```
% ls -l /dev/*da0*
ls: /dev/*da0*: No such file or directory
```

(en anglais *device to device*)
Syntaxe non habituelle, attention :

```
dd if=device-ou-fichier of=device-ou-fichier [bs=taille-bloc] [count=N]
```

où :

- « if » signifie *input file*
- « of » signifie *output file*
- « bs » signifie *block size*








## ◇ Exemple 1

Générer un fichier de 50 Mo :

```
# dd if=/dev/zero of=grosfichier bs=1M count=50
50+0 records in
50+0 records out
52428800 bytes transferred in 3.533640 secs (14837052 bytes/sec)
# ls -l grosfichier
-rw-------  1 root             52428800 Nov 16 01:27 grosfichier
```

## ◇ Exemple 2

Constituer un fichier image d'une disquette :

```
# dd if=/dev/fd0 of=/tmp/image.raw bs=16k
90+0 records in
90+0 records out
# ls -l /tmp/image.raw
-rw-r--r--   1 root root      1474560 Jan 25 22:36 /tmp/image.raw
```





## ◇ Exemple 2 bis

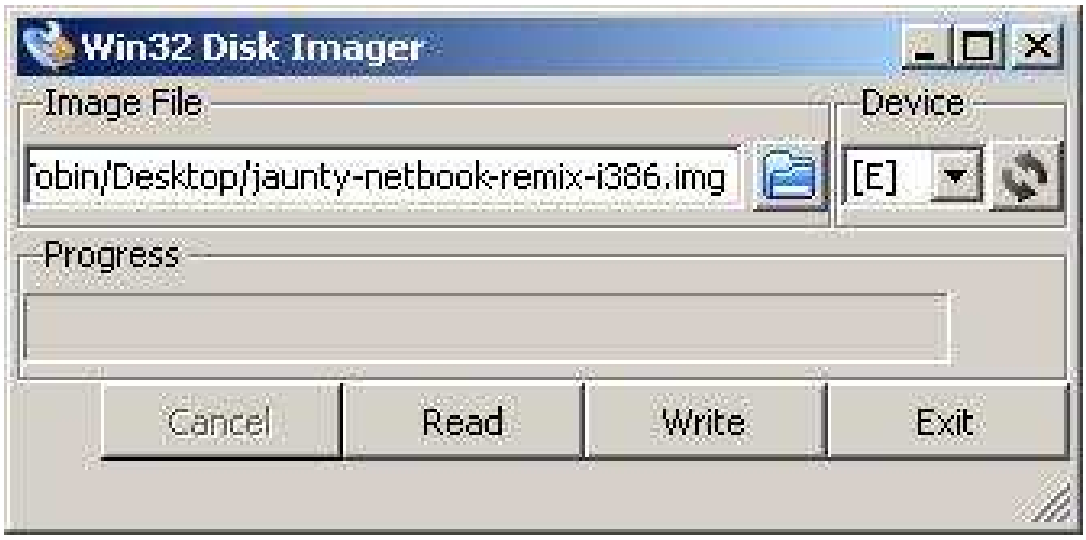
Recopier l'image sur une autre disquette :

```
# dd if=/tmp/image.raw of=/dev/fd0 bs=16k
90+0 records in
90+0 records out
# mdir a:
 Volume in drive A has no label
 Volume Serial Number is 2186-280C
Directory for A:/

document doc    378880 10-26-2003  18:45  document.doc
        1 file              378 880 bytes
                          1 078 784 bytes free
```

Ecriture d'images sur des clefs USB :

« `https://launchpad.net/win32-image-writer/` »



Chapitre 10

# *Systèmes de fichiers*





Exemple :

> *Source : LIBÉRATION vendredi 18 juin 2004 (Reuters - 08 :32) – par Lucas van Grinsven*
>
> LES SITES DE STOCKAGE DE PHOTOS NUMÉRIQUES SE MOBILISENT
>
> *AMSTERDAM - Trouver un stockage gratuit pour leurs données en ligne est devenu une préoccupation de millions d'internautes, que ce soit pour leur courrier électronique ou leurs photographies numériques.*
>
> *Deux des principaux sites de photographie numérique, Ofoto (`www.ofoto.com`), contrôlé par Kodak, et Fotango (`www.photos.fotango.com`), détenu par Canon, proposent une capacité illimitée de sauvegarde.*
>
> *Ofoto revendique plus de 13 millions d'usagers pour 450 millions de photos stockées sur ses serveurs.* ***Chaque semaine, le site ajoute cinq téraoctets de mémoire*** *pour accueillir cinq millions de nouvelles images.*

Comment cela marche-t-il ?

Plusieurs technologies de disques durs : ST-506, ESDI, SMD (*Storage Module Device*), ESMD, IPI, SCSI (1, 2, 3, Fibre Channel), IDE (ATA100, ATA133), Firewire, Disk On Flash, SATA, SAS, Solid State Disk (SSD), etc. etc.

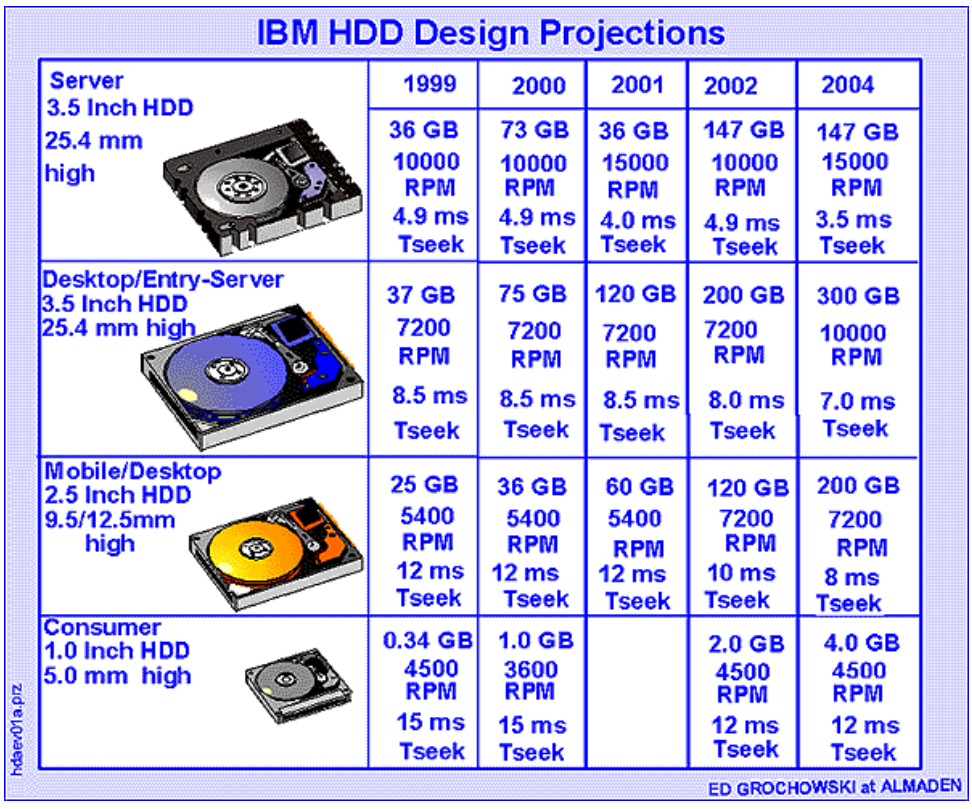



(source `http://www.almaden.ibm.com/sst/`)



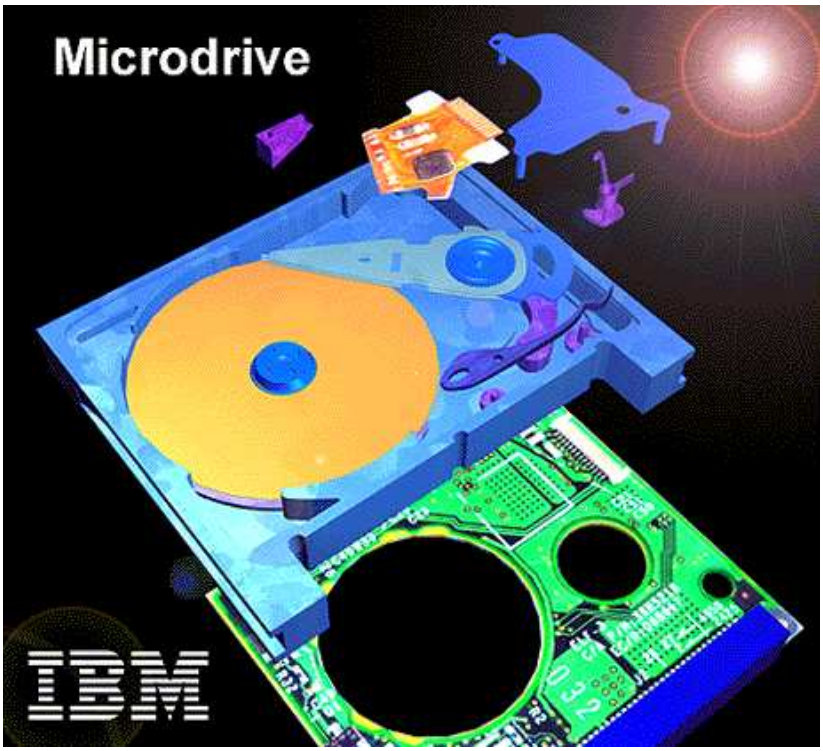


Perspectives :




(source `http://www.almaden.ibm.com/sst/`)

Structure d'un disque dur (quelle que soit sa technologie) :

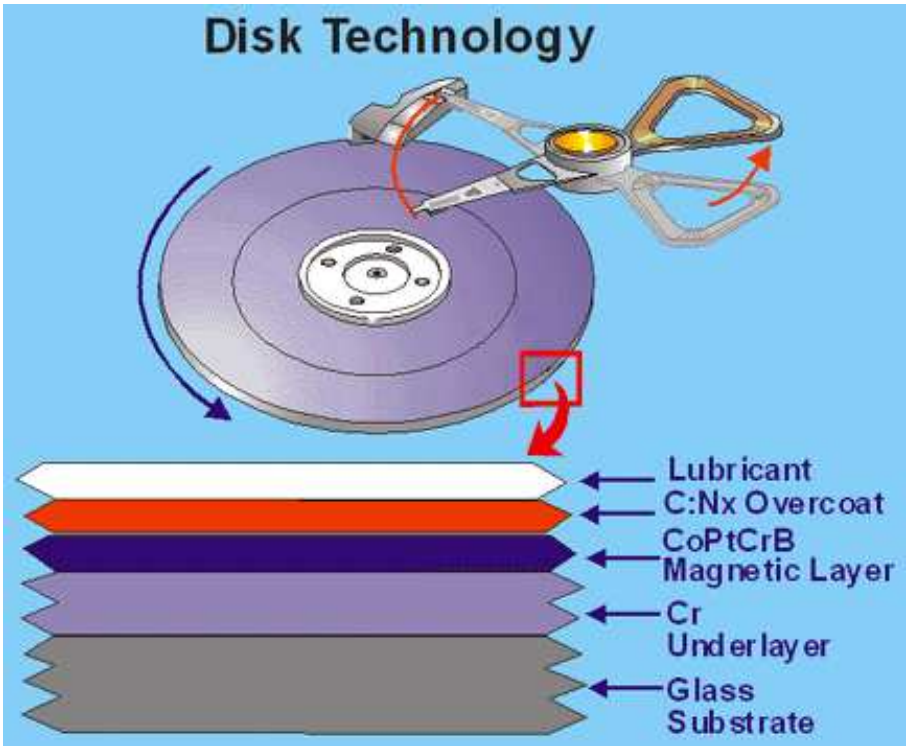



(source `http://www.almaden.ibm.com/sst/`)



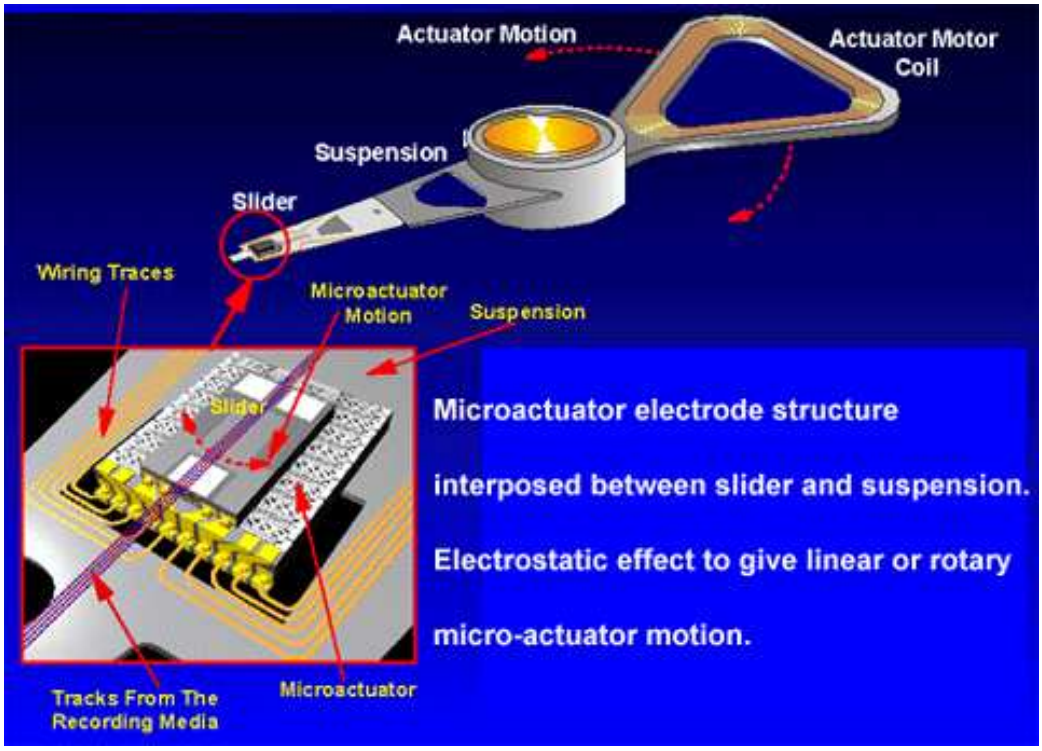


La tête d'un disque dur plane au dessus du plateau magnétique :




(source `http://www.almaden.ibm.com/sst/`)

Une tête de lecture d'un disque dur est un objet très sophistiquée, très fragile en cas de choc :




(source `http://www.almaden.ibm.com/sst/`)



## Chapitre 10 • Systèmes de fichiers

### §10.3 • Disques durs IDE

DE ≡ *Integrated Device Electronics*
ATA ≡ *AT Attachment* (le premier PC = *IBM pc AT*)
IDE → EIDE (*Extended IDE*) → ATAPI (*ATA Pocket Interface*)
(CDROM, DVDROM, DAT, etc.)

Type de disques apparu avec les PC.
Différences avec un disque SCSI :

- prix inférieur
- carte contrôleur différente (la mécanique est donc identique)

Evolution ?

- plus grosse capacité
- vitesse d'accès, de rotation

⇒ cela devrait profiter aussi à SCSI

(en anglais *Small Computer System Interface*)

SCSI : pas uniquement des disques
1986 : SCSI 1
1990 : SCSI 2
? : SCSI 3

Limitations de SCSI 2 :

- limitation à 16 périphériques par bus SCSI
- longueur de cable limitée
- l'ajout de périphérique diminue la longueur de cable autorisé
- c'est du ***parallel SCSI***





Améliorations : SCSI 3

- passage d'un cable de cuivre à du **fibre chanel** dont les trames peuvent transiter sur du cable de cuivre ou sur de la fibre optique
- nombre de périphériques illimité (limite à 16 millions)
- décomposition de la norme SCSI 3 en plusieurs couches (la norme SCSI 3 $\equiv$ 20 documents contre un seul pour SCSI 2)
  $\Rightarrow$ séparation entre les couches physiques de SCSI 3 et les couches hautes
- c'est du ***serial SCSI***

Par exemple :




Se reporter par exemple à :

- `http://www.pgsoftware.net` pour un revendeur français de deux marques d'appareils de ce type
- `http://www.aberdeeninc.com` pour un fabricant





Norme SMART :

- Hard Disks with SMART, by Bruce Allen, Linux Journal, Janu- ary 2004, pages 74-77. This is `http://www.linuxjournal.com/article.php?sid=6983` online
- If you would like to understand better how SMART works, and what it does, a good place to start is Section 8.41 of the "AT Attachment with Packet Interface-5" (ATA/ATAPI-5) specification. This documents the SMART functionality which the smartmontools utilities provide access to. You can find Revision 1 of this document at `http://www.t13.org/project/d1321r1c.pdf`
- Future versions of the specifications (ATA/ATAPI-6 and ATA/ATAPI-7), and later revisions (2, 3) of the ATA/ATAPI-5 specification are available from `http://www.t13.org/#FTP_site`
- The functioning of SMART was originally defined by the SFF-8035i revision 2 and the SFF-8055i revision 1.4 specifi- cations. These are publications of the Small Form Factors (SFF) Committee. Links to these documents may be found in the References section of the smartmontools home page at `http://smartmontools.sourceforge.net/`

## ◇ SOLARIS : commande `iostat -XEn`

```
% iostat -XEn
c0t0d0           Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Model: IC35L080AVVA07-0 Revision: VA4OA52A Serial No:       VNC402A4L4
Size: 82.34GB <82344714240 bytes>
Media Error: 0 Device Not Ready: 0  No Device: 0 Recoverable: 0
Illegal Request: 0

c0t2d0           Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Model: ST360015A        Revision: 3.33     Serial No: 5KC0G2A8
Size: 60.02GB <60019998720 bytes>
Media Error: 0 Device Not Ready: 0  No Device: 0 Recoverable: 0
Illegal Request: 0

c0t1d0           Soft Errors: 14 Hard Errors: 0 Transport Errors: 0
Vendor: _NEC     Product: DVD_RW ND-2500A  Revision: 1.06 Serial No:
Size: 0.00GB <0 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 14 Predictive Failure Analysis: 0
```





```
c1t5d0           Soft Errors: 5 Hard Errors: 0 Transport Errors: 0
Vendor: HP       Product: CD-Writer+ 9200  Revision: 1.0e Serial No:
Size: 0.00GB <0 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 5 Predictive Failure Analysis: 0

c2t0d0           Soft Errors: 0 Hard Errors: 222 Transport Errors: 258
Vendor: M-Sys    Product: DiskOnKey        Revision: 2.51 Serial No:
Size: 0.00GB <0 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0

rmt/0            Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: SUN      Product: DLT7000          Revision: 245F Serial No:
```

Ici : deux disques durs IDE internes, un graveur IDE de DVD, un graveur SCSI externe, une clef USB, un lecteur SCSI externe de bande DLT

## ◇ UNIX, WINDOWS : SMARTMONTOOLS

Outil SMARTMONTOOLS (version 5.33 au moment de l'écriture de ce support)
Cf site web : `http://smartmontools.sourceforge.net/`

Utilitaire contenu dans le CDROM live LINUX appelé KNOPPIX.
Cf `http://www.knoppix.org`

Comment utiliser SMARTMONTOOLS ?
Lancer « `smartctl -t long /dev/hda` »

Si l'on voit des lignes contenant « `FAILING_NOW` » alors **INQUIÈTUDE** !
Si l'on voit des lignes contenant « `ATA Error Count:` » alors **INQUIÈTUDE** !
Si l'on voit des lignes contenant « `Completed:` » alors **INQUIÈTUDE** !





Disque OK :

```
# smartctl -a /dev/hda
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
Home page is http://smartmontools.sourceforge.net/

=== START OF INFORMATION SECTION ===
Device Model:     TOSHIBA MK3021GAS
Serial Number:    Y3OO5910T
Firmware Version: GA129D
User Capacity:    30,005,821,440 bytes
Device is:        Not in smartctl database [for details use: -P showall]
ATA Version is:   5
ATA Standard is:  Exact ATA specification draft version not indicated
Local Time is:    Sun Nov 13 14:51:50 2005 Romance Standard Time
SMART support is: Available - device has SMART capability.
SMART support is: Enabled

=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
...
```

```
SMART Error Log Version: 1
No Errors Logged

SMART Self-test log structure revision number 1
Num  Test_Description    Status                  Remaining  LifeTime(hour
# 1  Short offline       Completed without error       00%       146
# 2  Short offline       Completed without error       00%        10
# 3  Short offline       Completed without error       00%         8
```

Tout parait normal :

- car pas de « `FAILING_NOW` »
- car pas de « `ATA Error Count:` »
- car pas de « `Completed:` »





Disque en panne :

```
# smartctl -a /dev/hda
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
Home page is http://smartmontools.sourceforge.net/

=== START OF INFORMATION SECTION ===
Device Model:     HDS722580VLAT20
Serial Number:    VNR21EC2CGXASL
Firmware Version: V32OA60A
User Capacity:    82,348,277,760 bytes
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   6
ATA Standard is:  ATA/ATAPI-6 T13 1410D revision 3a
Local Time is:    Mon Nov 14 01:04:10 2005 Paris, Madrid
SMART support is: Available - device has SMART capability.
SMART support is: Enabled

=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
...
```

```
SMART Error Log Version: 1
ATA Error Count: 199 (device log contains only the most recent five error
...

Error 199 occurred at disk power-on lifetime: 1393 hours (58 days + 1 hou
  When the command that caused the error occurred, the device was active

  After command completion occurred, registers were:
  ER ST SC SN CL CH DH
  -- -- -- -- -- -- --
  40 51 08 af 10 60 e0  Error: UNC 8 sectors at LBA = 0x006010af = 629572

  Commands leading to the command that caused the error were:
  CR FR SC SN CL CH DH DC   Powered_Up_Time  Command/Feature_Name
  -- -- -- -- -- -- -- --  ----------------  --------------------
  c8 00 08 af 10 60 e0 00      02:07:04.200  READ DMA
  ca 00 08 2f 0d 60 e0 00      02:07:04.200  WRITE DMA
  ca 00 10 df 86 5e e0 00      02:07:04.200  WRITE DMA
  ca 00 01 47 5e 00 e0 00      02:07:04.200  WRITE DMA
  c8 00 08 79 80 71 e1 00      02:07:04.100  READ DMA
...
```





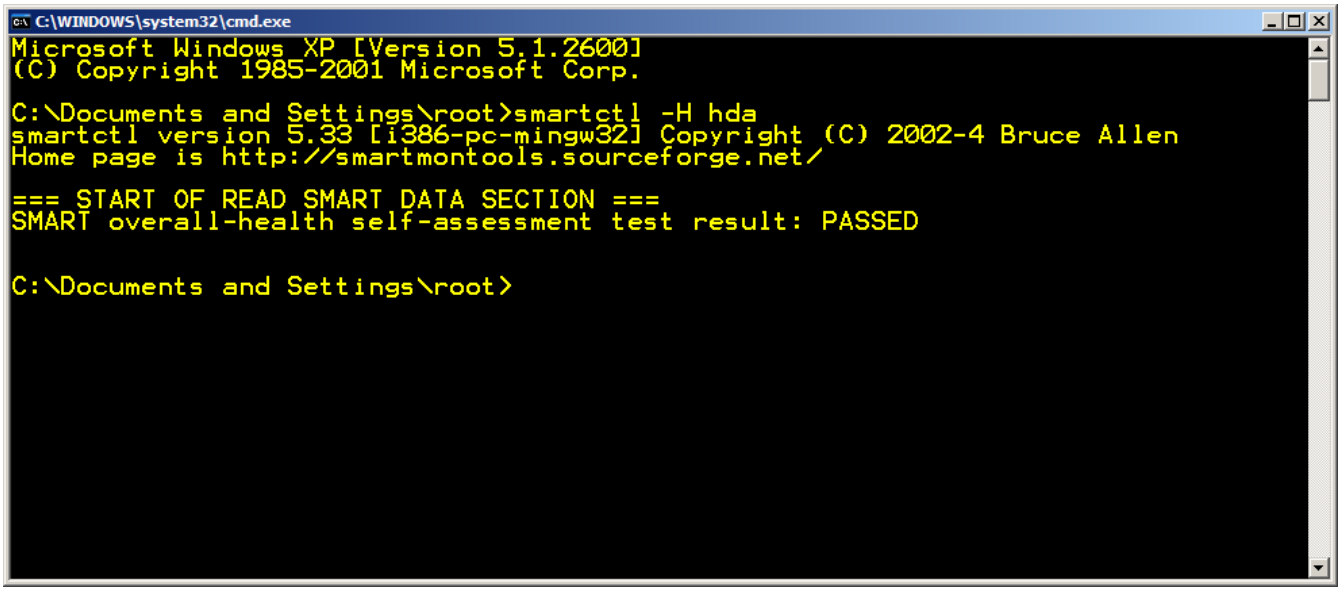
```
SMART Self-test log structure revision number 1
Num  Test_Description    Status                  Remaining  LifeTime(hour
# 1  Short offline       Completed: read failure       10%      1393
# 2  Short offline       Completed: read failure       10%      1393
# 3  Extended offline    Completed: read failure       90%      1393
# 4  Extended captive    Completed: read failure       90%      1391
```

Ici des anomalies :

- pas de « `FAILING_NOW` »
- « `ATA Error Count: 199` » c'est-à-dire 199 erreurs détectées par le hardware
- 4 lignes « `Completed:` » c'est-à-dire 4 tests lancés se terminant sur « read failure »

⇒ disque HS

Pour WINDOWS, programme SMARMONTOOLS aussi. Cf
`http://smartmontools.sourceforge.net/`





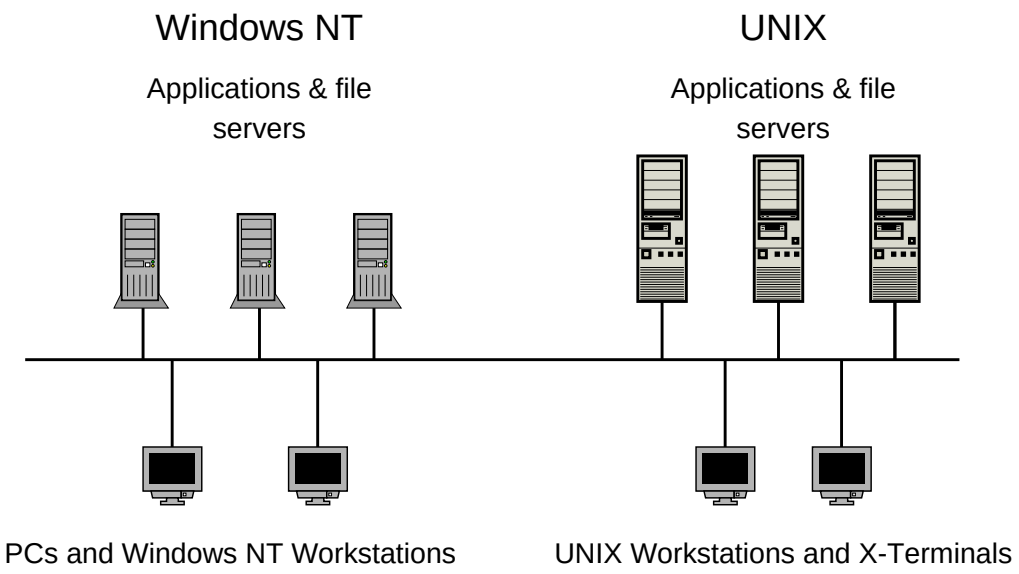



## ◇ Tests de disques durs : outils de constructeurs

Des applications de tests de disques durs sont disponibles sur les sites suivants :

- applications pour disques de marque Western Digital :
  `http://support.wdc.com/download/index.asp#diagutils`
- applications pour disques de marque Seagate :
  `http://www.seagate.com/support/seatools/`
- applications pour disques de marque Hitachi :
  `http://www.hitachigst.com/hdd/support/download.htm`
- applications pour disques de marque Fujitsu :
  `http://www.fcpa.com/support/hard-drives/software_utilities.htr`
- applications pour disques de marque Maxtor :
  `http://www.maxtor.com/en/support/downloads/`
- applications de la société OnTrack pour marques diverses de disques :
  `http://www.ontrack.fr/`

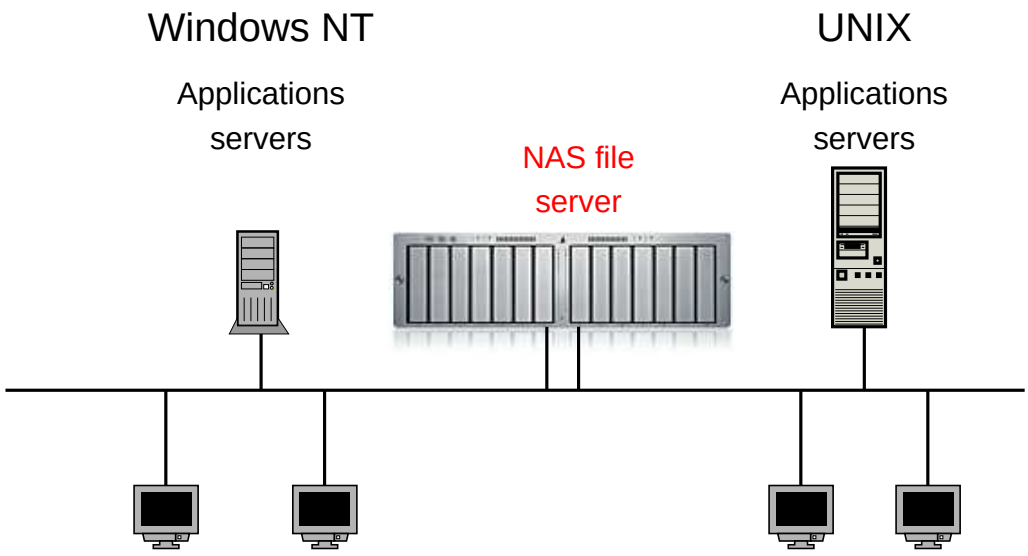
Problème : les multiples serveurs de fichiers du réseau





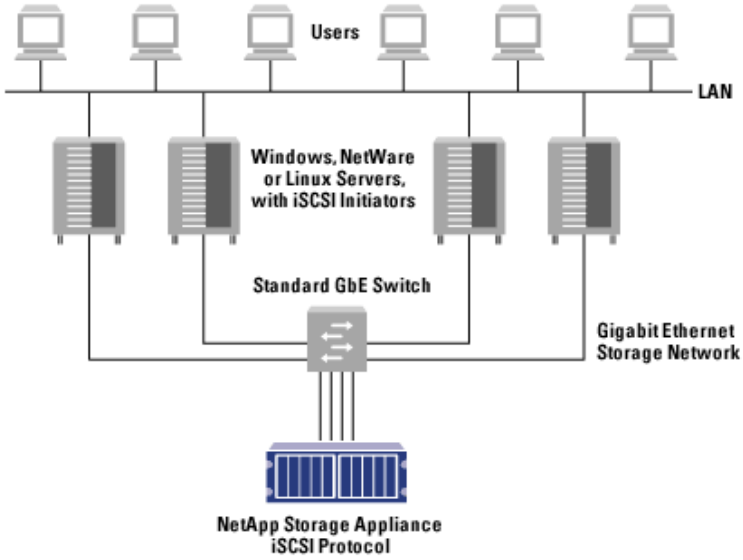



Solution du NAS : une machine spécialisée équipée de disques RAID et qui sert ces fichiers via le réseau IP en protocoles NFS, CIFS en général.




La marque la plus connue : Network Appliance, `http://www.netapp.com`.

Principe : chaque machine cliente de fichiers est équipée d'une carte adaptateur l'interconnectant à un réseau Fibre Channel composé d'unités de stockage (RAID, TAPE, etc.) mises en commun
**SAN est un réseau qui utilise le protocole serial SCSI**.








RAID ≡ Redundancy Array of Inexpensive Disks

Papier historique : « A Case tor Redundant Arrays of Inexpensive Disks (RAID) » par Patterson, Gibson, et Katz de l'Université de Berkeley Californie.

`http://sunsite.berkeley.edu:80/Dienst/Repository/2.0/Body/ncstrl.ucb/CSD-87-391/pd`

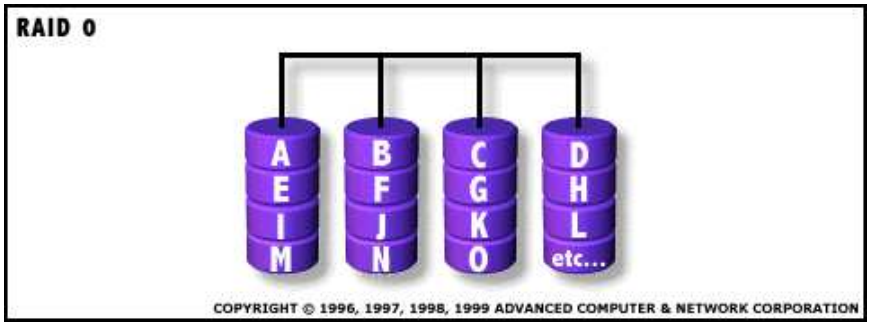
« Principe (en simplifiant) » : on n'utilise plus de disques durs indépendants mais un ensemble de disques durs (en général rackables) gérés par des contrôleurs disques spécialisés.



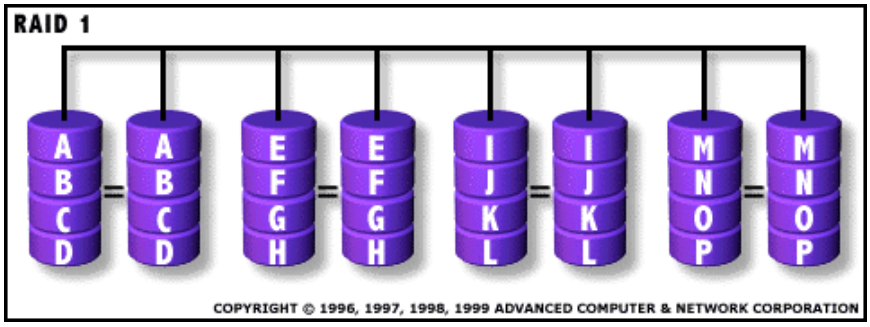
L'ensemble est censé encaisser la panne d'au moins un disque dur. En cas de panne d'un disque dur, le système continue de travailler. Un disque en remplacement sera rempli par des données reconstruites à partir des données sur le reste des disques de l'ensemble.







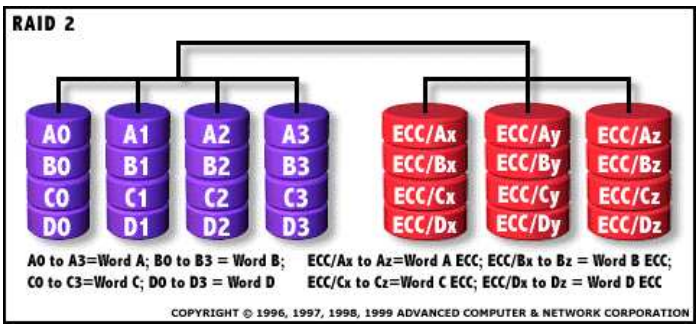

- Principe du RAID 0 : **striping**
  - minimum de 2 disques
  - les données sont découpées en blocs (A, B, C, etc.) écrits sur des disques distincts
- Avantages :
  - I/O en paralléle
  - simple à réaliser
  - maximum de performance
- Inconvénients :
  - non tolérant aux pannes (un disque en panne ⇒ tout est perdu)



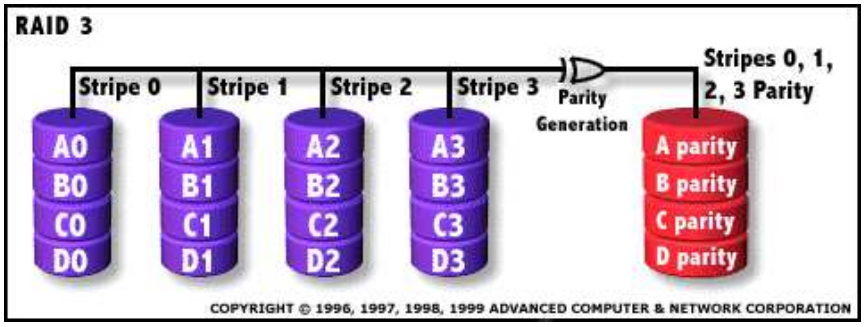

- Principe du RAID 1 : **mirroring et duplexing**
  - minimum de 2 disques
  - les données sont découpées en blocs (A, B, C, etc.) écrits sur des disques jumeaux
- Avantages :
  - lecture sur n'importe lequel des disques jumeaux ⇒ lecture deux fois plus performante
  - simple à réaliser
- Inconvénients :
  - pas très efficace (100% de disques en plus)
  - charge le CPU en général (car réalisation en logiciel et pas en hardware)







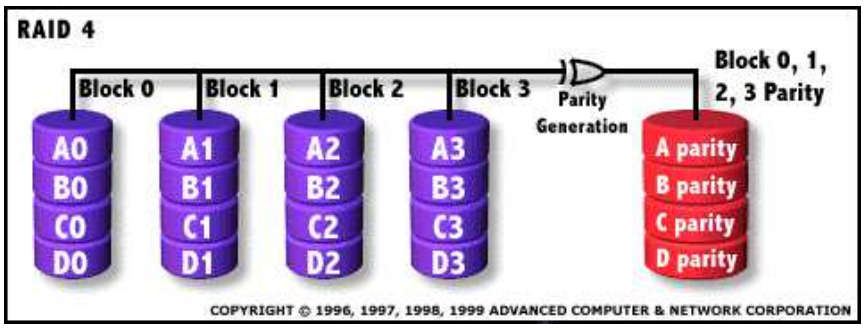

- Principe du RAID 2 : **code correcteur d'erreur ECC**
  - la donnée est l'octet
  - on découpe les données et on en calcule le code ECC que l'on écrit sur des disques dédiés
  - une lecture est vérifiée par le code ECC associé à la donnée ; on corrige ainsi une panne de disque
- Avantages :
  - grand taux de transfert de données
- Inconvénients :
  - inefficace
  - cher ; peu d'offres commerciales



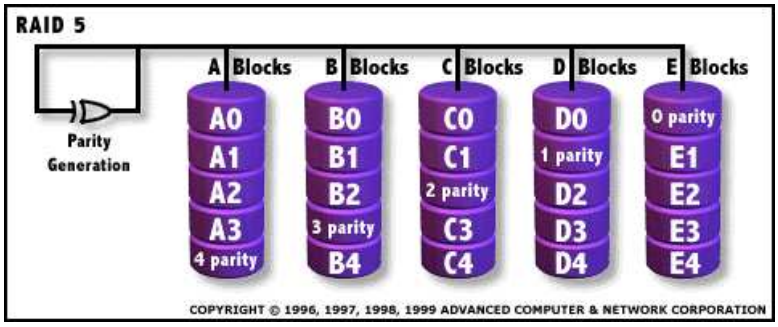

- Principe du RAID 3 (version simplifiée du RAID 2 avec un seul disque de parité) :
  - minimum de 3 disques
  - on découpe les octets en stripes écrits sur les disques
  - on calcule la parité du stripe que l'on écrit sur le disque dédié de parité
  - on vérifie la parité du stripe lors des lectures
- Avantages :
  - taux élevé de lectures ou d'écritures
  - panne disque relativement insensible
- Inconvénients :
  - contrôleur disque complexe







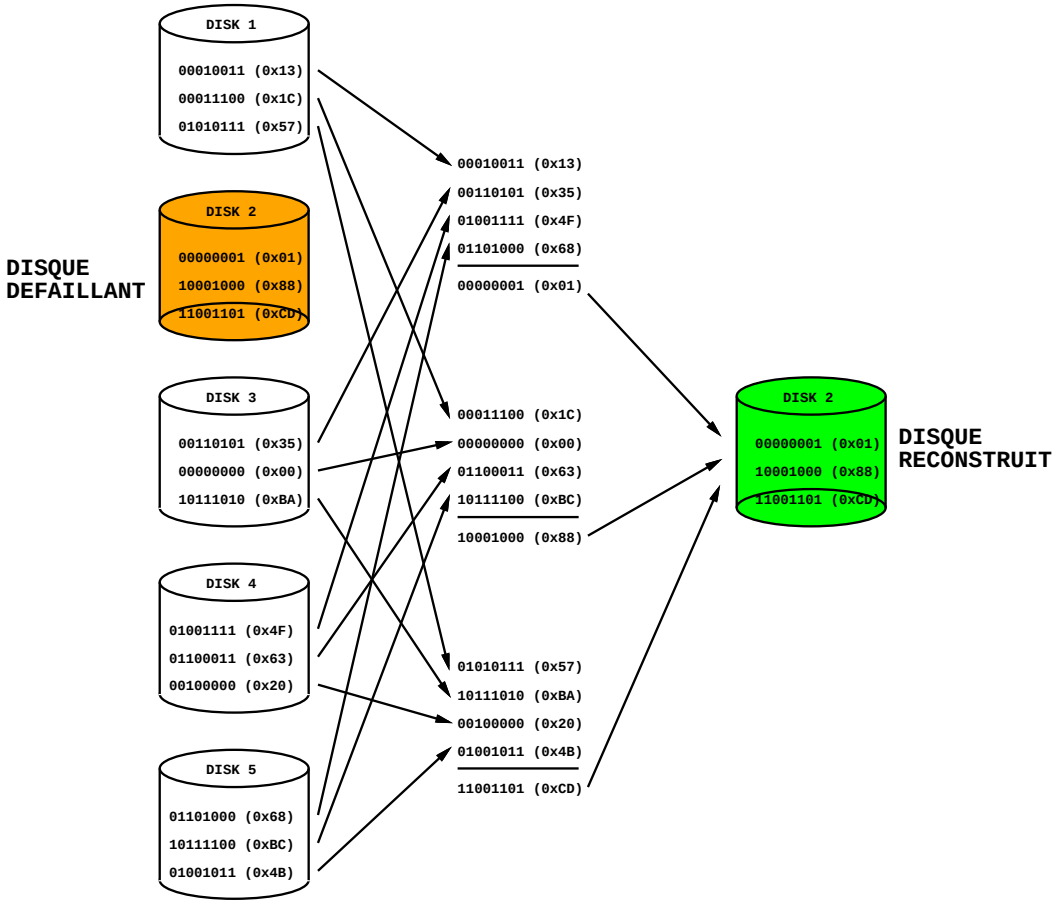

- Principe du RAID 4 :
  - minimum de 3 disques
  - on travaille par bloc d'octets que l'on disperse sur les disques durs
  - on calcule la parité de ces blocs que l'on écrit sur le disque dédié de parité
- Avantages :
  - les disques durs peuvent agir indépendamment
- Inconvénients :
  - reconstruction d'un disque complexe
  - pour écrire un bloc, on doit lire le bloc et sa parité ⇒ globalement plus lent
  - peu d'offres commerciales




- Principe du RAID 5 :
  - minimum de 3 disques
  - chaque bloc d'octets est écrit sur un disque
  - la parité est dispersée sur tous les disques évitant ainsi un goulet d'étranglement du RAID 4.
- Avantages :
  - Fort taux de transfert en lecture, taux moyen en écriture
  - Solution populaire
- Inconvénients :
  - Contrôleur complexe
  - Reconstruction délicate d'un disque







Filesystem ≡ structure performante de gestion des fichiers

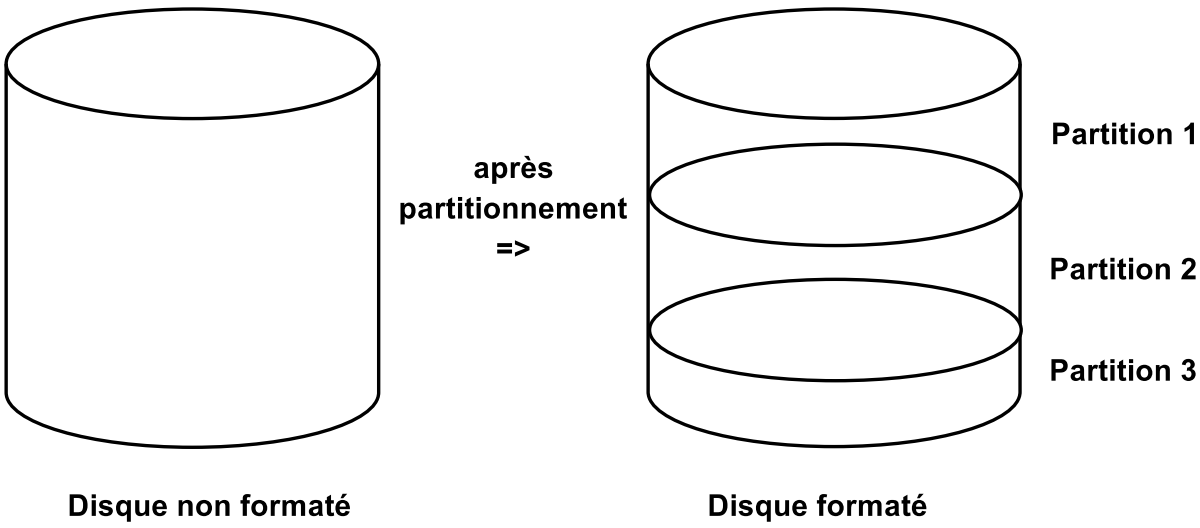
Exemple : *Fast File System* dit FFS, EXT2 ou EXT3 de LINUX, etc.

Brièvement, en voici quelques caractéristiques :

1. Notion d'inode
2. Notion d'écriture géographiquement proche
3. Optimisation de la gestion de la liste des blocs libres
4. Stratégie d'allocation des blocs







Pourquoi partitionner un disque dur en filesystems ?

- pour cloisonner les utilisateurs c'est-à-dire les problèmes
- pour faciliter les sauvegardes
- pour limiter l'expansion de certains utilisateurs au détriment des autres
- pour faciliter l'utilisation de NFS (Network File System)

Problèmes courants avec les partitions :

- Une partition est pleine alors qu'une autre, surdimensionnée est quasi vide.
- Les différents constructeurs divergent sur les implémentations des partitions.

Une partition peut contenir :

- un filesystem
- une zone de swap, sans structure de filesystem plaquée dessus
- rien ; la partition est réservée pour un usage ultérieur





Les informations concernant les partitions sont stockées dans une zone du disque dite **disk label** (sur SunOS dans le premier secteur de la première partition).

Le disk label contient :

- caractéristiques physiques du disque
- paramètres de partitionnement
- secteurs défectueux

Les commandes pour déposer un disk label sont propres à chaque constructeur (« `disklabel` », « `format` » ...) :

- Sur Solaris : « `format` »
- Sur Linux : « `fdisk` »

(en anglais *print virtual table of content*)

Syntaxe : `prtvtoc raw-device`

```
# prtvtoc /dev/rdsk/c1t1d0s2
* /dev/rdsk/c1t1d0s2 partition map
*
* Dimensions:
*     512 bytes/sector
*     417 sectors/track
*       6 tracks/cylinder
*    2502 sectors/cylinder
*   14342 cylinders
*   14340 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       0      8    00          0  35878680  35878679
       2      5    01          0  35878680  35878679   /mnt/ufs/1
```





Syntaxe : `fdisk -l`

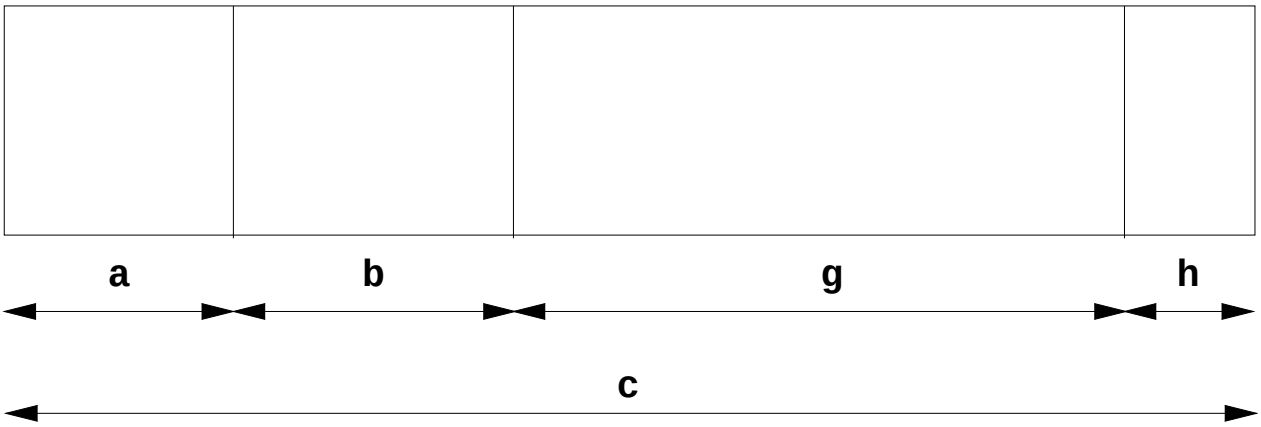
```
# /sbin/fdisk -l /dev/hdc

Disk /dev/hdc: 16 heads, 63 sectors, 148945 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot     Start        End     Blocks   Id  System
/dev/hdc1              1       4162   2097616+  82  Linux swap
/dev/hdc2           4163      75000  35702352   83  Linux
/dev/hdc3          75001     148945  37268280   83  Linux
```

En général, on a 8 partitions par disque au plus.

Leur géographie est conventionnellement quelque chose du genre :








## ◇ Partitions LINUX

Il y aura une différence selon que les disques utiliseront des contrôleurs autres que les contrôleurs IDE de base.
Exemple avec un contrôleur IDE :

```
% df -k -t ext3
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hdd3              1032056    244640    734976  25% /
/dev/hdd1                63413     23066     37073  39% /boot
/dev/hdd8              2064168     62636   1896680   4% /tmp
/dev/hdd6              4128400   1686152   2232536  44% /var
/dev/hdd2              4128432   3069260    849460  79% /usr
/dev/hdd5              4128400    276480   3642208   8% /home
```

Exemple avec un contrôleur RAID SCSI :

```
# df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/cciss/c0d0p2     32890776  24691344   6528672  80% /
/dev/cciss/c0d0p1        98747     10991     82657  12% /boot
/dev/cciss/c0d1p1     35002928  12628172  20596696  39% /share/vmware2
```

## ◇ Partitions SOLARIS

`/dev/dsk/c`$n_1$`t`$n_2$`d`$n_3$`s`$n_4$ (contrôleur, target, disk, slice) :

```
% df -F ufs -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     246167   60541  161010    28%    /
/dev/dsk/c0t0d0s6     962582  419332  485496    47%    /usr
/dev/dsk/c0t1d0s0    1015695  822607  132147    87%    /net/serveur/home
/dev/dsk/c0t1d0s1    7717573 3463222 4177176    46%    /usr/local
/dev/dsk/c0t0d0s5     183687   13017  152302     8%    /opt
```





## ◇ Partitions BSD

Les noms typiques sont de la forme `/dev/`*ddnp*, où :

- *dd* est un code mnémonique du driver ;
- *n* est le numéro du disque ;
- *p* est une lettre entre `a` et `h` identifiant la partition ;

```
% df
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/sd0a       95359    42623    45108    49%    /
```

(en anglais *make file system*)

Commande : `mkfs [options] fichier-spécial`

Par exemple sur Solaris :

```
# mkfs -F ufs /dev/rdiskette
size not specified
ufs usage: mkfs [-F FSType] [-V] [-m] [-o options] special size(sectors)
 -m : dump fs cmd line used to make this partition
 -V :print this command line and return
 -o :ufs options: :nsect=32,ntrack=16,bsize=8192,fragsize=1024
 -o :ufs options: :cgsize=0,free=0,rps=60,nbpi=2048,opt=t
 -o :ufs options: :apc=0,gap=0,nrpos=0,maxcontig=0
NOTE that all -o suboptions: must be separated only by commas so as to
be parsed as a single argument
```

⇒ **la commande nécessite des paramètres complexes**





(en anglais *new file system*)

La commande `newfs` offre des options simples et appelle en fait derrière `mkfs` avec ce qu'il faut.
Syntaxe la plus répandue : `newfs [options] fichier-spécial`

Options intéressantes :

- option « `-m` » : précise la taille de minfree.
- option « `-i` » : précise le nombre d'octets de données pour un inode (2048 octets pour un inode par défaut).
- option « `-v` » : affiche la commande « `mkfs` » qui serait lancée

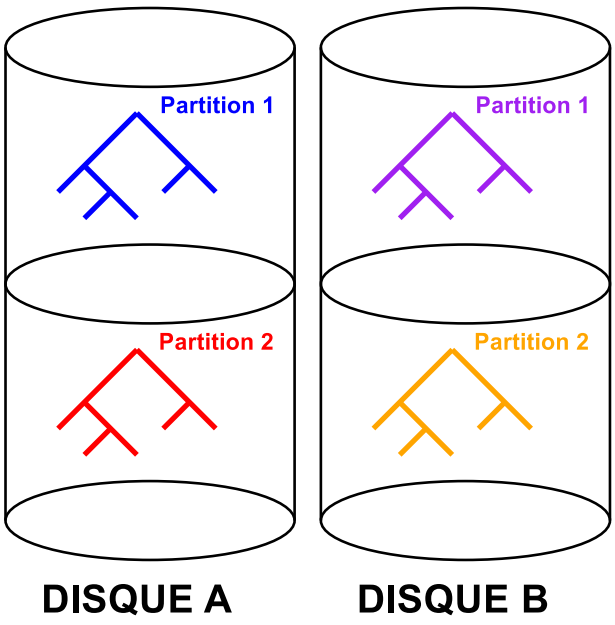
Par exemple sur Solaris :

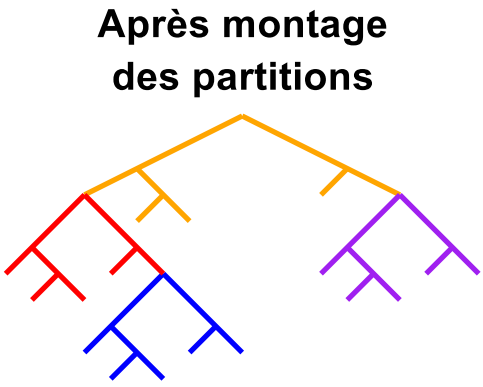
```
# newfs -v /dev/rdiskette
newfs: construct a new file system /dev/rdiskette: (y/n)? y
mkfs -F ufs /dev/rdiskette 2880 18 2 8192 1024 16 10 5 2048 t 0 -1
/dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18 secto
        1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 640, 1184, 1792, 2336,
```





La greffe d'une partition d'un disque sur l'arborescence s'appelle le « **montage** » d'un filesystem.



Syntaxe : `mount [options] fichier-spécial point-de-montage`
Exemples :

```
# mount /dev/sd0g /mnt
# mount -t cdfs -o rrip,ro /dev/rz4c /cdrom
```





Attention :

1. Le point de montage doit exister avant de lancer la commande.
2. Le contenu du point de montage devient invisible après le montage.
3. On ne peut monter un filesystem que si celui-ci a une structure interne marquée comme cohérente (cf la suite sur fsck, page 502) :

```
# mount /users
WARNING: R/W mount of /users denied. Filesystem is not
clean - run fsck.
```

◇ Montage de CDROM/DVDROM

Sous UNIX, un CDROM/DVDROM une fois monté n'est plus éjectable.

Il faut monter un CDROM/DVDROM en mode readonly.





Syntaxe : `umount point-de-montage`

Attention : on ne peut pas démonter une partition

1. si une commande s'exécute dans la partition
2. si des fichiers sont ouverts dans la partition
3. si l'on a un répertoire courant dans un répertoire de la partition

(ce qui se ramène à avoir un file descripteur ouvert sur la partition dans tous les cas)

Message d'erreur classique :

```
# cd /mnt
# umount /mnt
umount: /mnt: Device busy
```

◇ Démontage de CDROM

Sous UNIX, un CDROM une fois démonté redevient éjectable.





Rappel : Traditionnellement un boot en single user monte la partition « / » uniquement et elle est en read-only.

Contexte :

- On boote en single user.
- On veut écrire sur « / » (par exemple pour corriger quelque chose)

Comment faire ? car on ne peut pas démonter « / » car la partition est busy (seule partition montée et on est dedans).

⇒ existence d'une option de « remontage à chaud » de partition :

```
# mount -o remount,rw /
```

(en anglais *file user*)

Contexte : on veut démonter une partition mais elle est busy !

Une solution : commande « `fuser` » (à peu près standard)

```
# cd /mnt
# umount /mnt
umount: /mnt: Device busy
# fuser /mnt
/mnt:     17853c   10237c
# echo $$
17853
```





(en anglais *list of open files*)

Contexte : on veut démonter une partition mais elle est busy !

Une solution : commande « `lsof` »
`ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/`

```
# cd /mnt
# umount /mnt
umount: /mnt: Device busy
# lsof /mnt
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    21372 root  cwd   VDIR   36,2      512    2 /mnt
lsof    21376 root  cwd   VDIR   36,2      512    2 /mnt
lsof    21377 root  cwd   VDIR   36,2      512    2 /mnt
# echo $$
21372
```

(en anglais *display free*)

La commande « **df** » permet de voir la liste des montages :

```
% df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     123231   39164   71744    36%    /
/dev/dsk/c0t0d0s3    1523574  699814  762818    48%    /usr
/dev/dsk/c0t0d0s4     492422   27835  415345     7%    /var
/dev/dsk/c0t0d0s5    3095966 1371773 1662274    46%    /opt
serveur.example.com:/users
                     4375366 2949034 1382579    69%    /users
serveur.example.com:/logiciels
                      650847  343011  242756    59%    /logiciels
```

**Attention** : certains UNIX affichent les informations en blocs de 512 octets.
⇒ utiliser l'option « `-k` » pour préciser d'afficher en kilo-octets.





On peut limiter l'affichage à certains types de filesystems :

- Sur LINUX et la plupart des UNIX traditionnels :
  `df -t type-du-filesystem`
  (par exemple sur LINUX : types « `ext2` », « `ext3` », « `iso9660` », « `proc` », « `nfs` »)
- Sur SUN Solaris :
  `df -F type-du-filesystem`
  (par exemple : types « `ufs` », « `hsfs` », « `proc` », « `lofs` », « `nfs` »)

◇ Exemples sur LINUX :

On peut limiter l'affichage aux disques locaux (c'est-à-dire présents sur la machine même) :

```
% df -k -t ext2
Filesystem       1K-blocks      Used Available Use% Mounted on
/dev/sda3         15330272   6080340   8471184  42% /
/dev/sda1           256667     34981    208434  15% /boot
```





◇ Exemples sur SUN Solaris :

On peut limiter l'affichage aux disques locaux (c'est-à-dire présents sur la machine même) :

```
% df -k -F ufs
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     123231   39164   71744    36%    /
/dev/dsk/c0t0d0s3    1523574  699814  762818    48%    /usr
/dev/dsk/c0t0d0s4     492422   27835  415345     7%    /var
/dev/dsk/c0t0d0s5    3095966 1371773 1662274    46%    /opt
```

On peut limiter l'affichage aux disques distants :

```
% df -k -F nfs
Filesystem            kbytes    used   avail capacity  Mounted on
serveur.example.com:/users
                     4375366 2949034 1382579    69%    /users
serveur.example.com:/logiciels
                      650847  343011  242756    59%    /logiciels
```

**Attention** : « `df` » peut bloquer sur un montage défaillant (disque presque en panne, serveur réseau injoignable).
⇒ Lancer « `df &` » en cas de blocage possible.

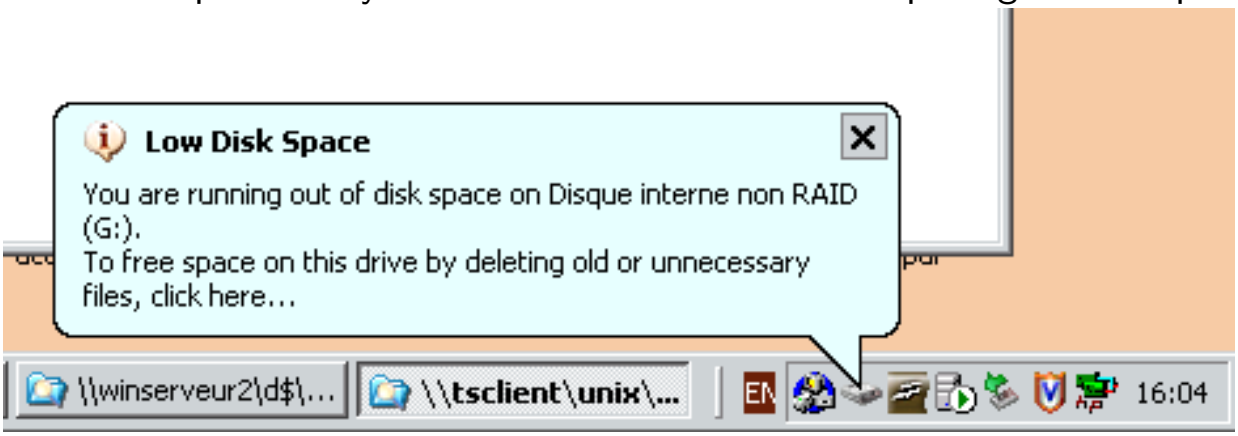




**Attention** : ne pas confondre :

- la commande « `df` » renvoie la liste des partitions montées
- la commande « `fdisk -l` » renvoie la liste des partitions d'un disque, que les partitions soient montées ou pas

WINDOWS peut envoyer une bulle avertissant du remplissage des disques.




Pour s'en débarasser, voici la clef de registres à positionner :

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
\Policies\Explorer]
"NoLowDiskSpaceChecks"=dword:00000001
```





La commande « `mount` » permet aussi d'avoir la liste des filesystems montés :

```
% mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/intr/largefiles/logging/onerror=panic/dev
/usr on /dev/dsk/c0t0d0s3 read/write/setuid/intr/largefiles/logging/onerror=panic/
/proc on /proc read/write/setuid/dev=2d80000 on Tue Oct 10 17:49:11 2000
/dev/fd on fd read/write/setuid/dev=2e40000 on Tue Oct 10 17:49:14 2000
/etc/mnttab on mnttab read/write/setuid/dev=2f40000 on Tue Oct 10 17:49:16 2000
/var on /dev/dsk/c0t0d0s4 read/write/setuid/intr/largefiles/logging/onerror=panic/
/var/run on swap read/write/setuid/dev=1 on Tue Oct 10 17:49:16 2000
/tmp on swap read/write/setuid/dev=2 on Tue Oct 10 17:49:18 2000
/opt on /dev/dsk/c0t0d0s5 read/write/setuid/intr/largefiles/logging/onerror=panic/
/infosystems on /dev/dsk/c0t0d0s6 read/write/setuid/intr/largefiles/logging/onerro
/users on serveur.example.com:/users remote/read/write/setuid/dev=2fc0001  on Tue
/logiciels on serveur.example.com:/logiciels remote/read/write/setuid/dev=2fc0002
```

**Attention** : « `mount` » peut bloquer sur un montage défaillant (disque presque en panne, serveur réseau injoignable).
⇒ Lancer « `mount &` » en cas de blocage possible.





**Attention** : ne pas confondre :

- la commande « `mount` » renvoie la liste des partitions montées
- la commande « `fdisk -l` » renvoie la liste des partitions d'un disque, que les partitions soient montées ou pas

(en anglais *file system table*)

Au démarrage, les disques sont montés automatiquement par la commande « `mount -a` ».

L'option « `-a` » signifie *all*.

La commande « `mount -a` » utilise le contenu du fichier « `/etc/fstab` » et monte tous les volumes mentionnés dedans.

Sur Solaris, le fichier équivalent à « `/etc/fstab` » s'appelle « `/etc/vfstab` » et suit une syntaxe équivalente.





Par exemple sur Linux, le fichier « `/etc/fstab` » contient quelque chose comme :

```
/dev/hda1       /               ext2        defaults            1 1
/dev/cdrom      /mnt/cdrom      iso9660     noauto,owner,ro     0 0
/dev/fd0        /mnt/floppy     auto        noauto,owner        0 0
none            /proc           proc        defaults            0 0
none            /dev/pts        devpts      gid=5,mode=620      0 0
/dev/hda5       swap            swap        defaults            0 0
```

Le format de « `/etc/fstab` » (ou équivalent) :

- champ 1 : nom du fichier spécial
- champ 2 : point de montage
- champ 3 : type de flesystem
- champ 4 : options de montage
- champ 5 : utilisé par la commande « `dump` »
- champ 6 : numéro d'ordre du disque lors d'une vérification de cohérence des disques par la commande « **fsck** »





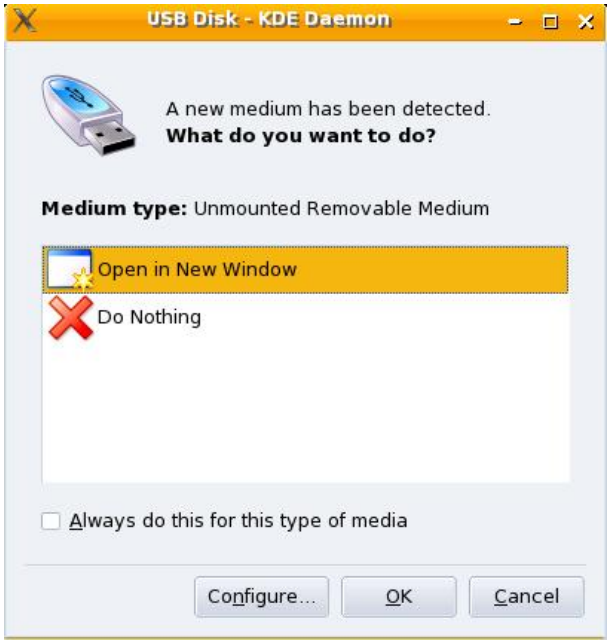
Une fois renseigné le fichier « `/etc/fstab` », on peut :

- monter la totalité des partitions : « `mount -a` »
- monter une certaine partition : « `mount /point/de/montage` » comme par exemple :

```
# mount /mnt/cdrom
```

- démonter la totalité des partitions : « `umount -a` »

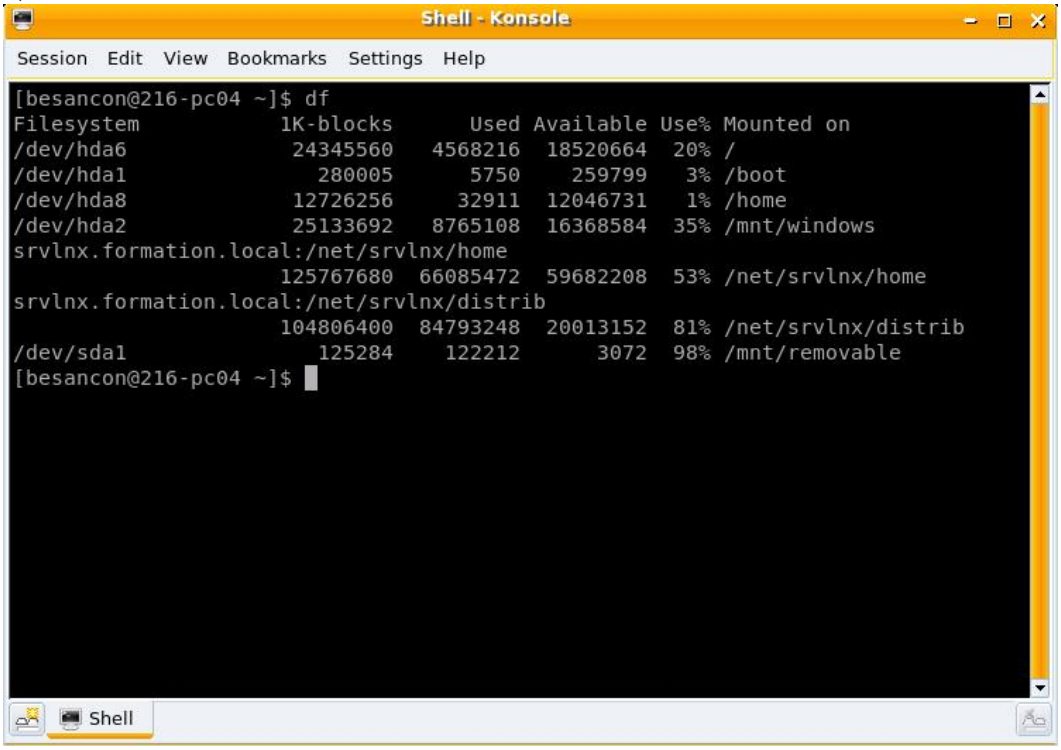
Les clefs USB sont directement reconnues par le bureau de l'environnement de multifenêtrage KDE de la salle de TP de la Formation Permanente :








La clef apparait en tant qu'icone sur le bureau et en tant que « `/mnt/removable` » dans l'arborescence des fichiers.




La clef est montée en tant que « `/mnt/removable` ».

La clef USB doit être formatée en FAT ou FAT32.

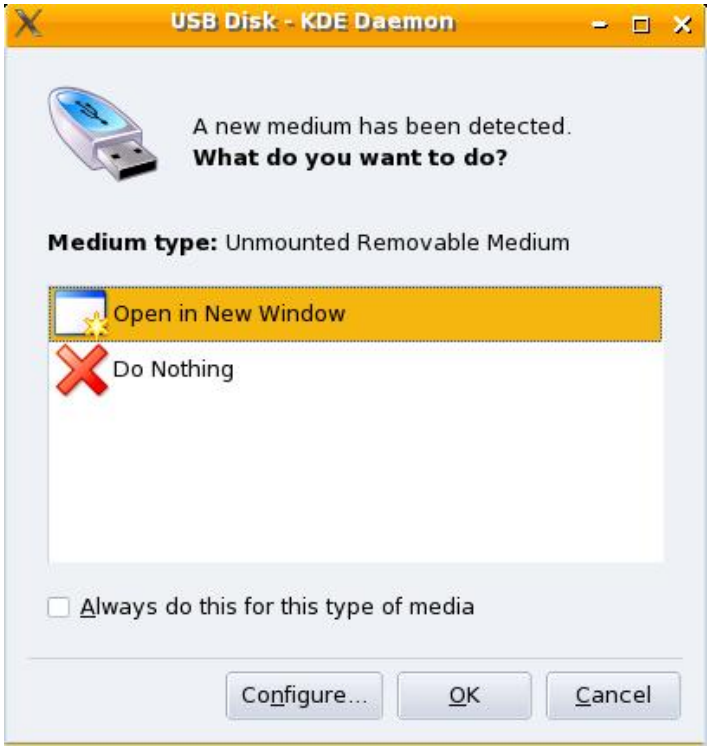
La clef USB doit être formatée en mode disque dur et doit comporter par conséquent une table de partition.

Une clef USB formatée en mode disquette (c'est-à-dire sans table de partition) ne sera pas reconnue par le système LINUX de la salle de TP de la Formation Permanente.

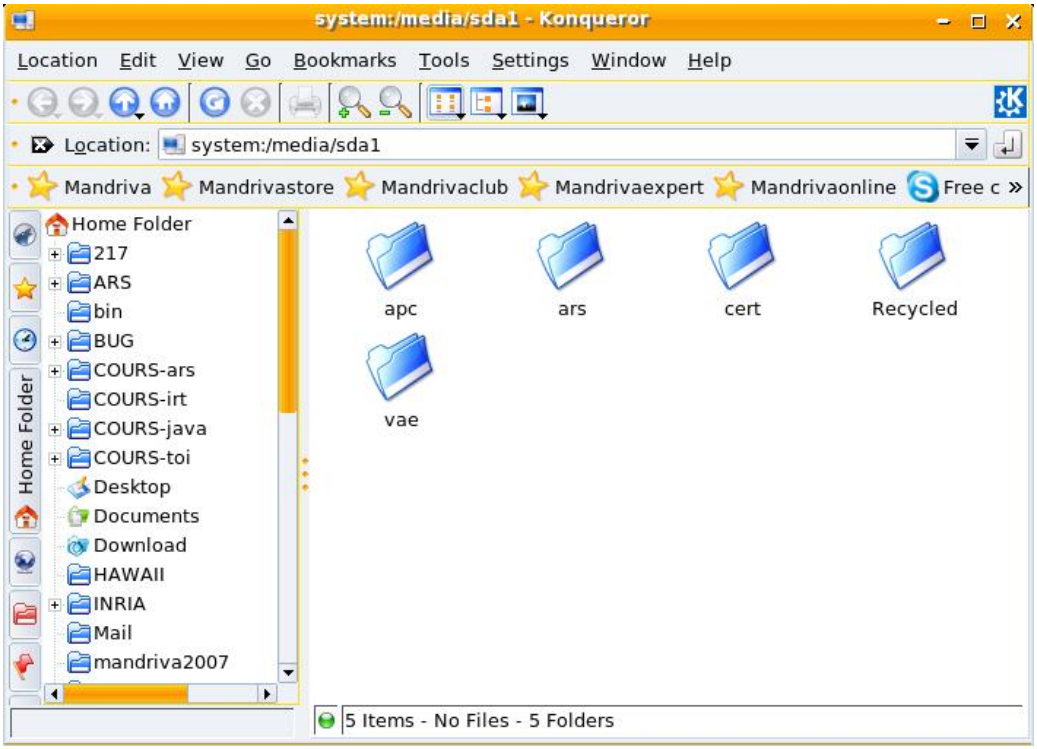




◇ Mandriva 2007 : insertion de la clef



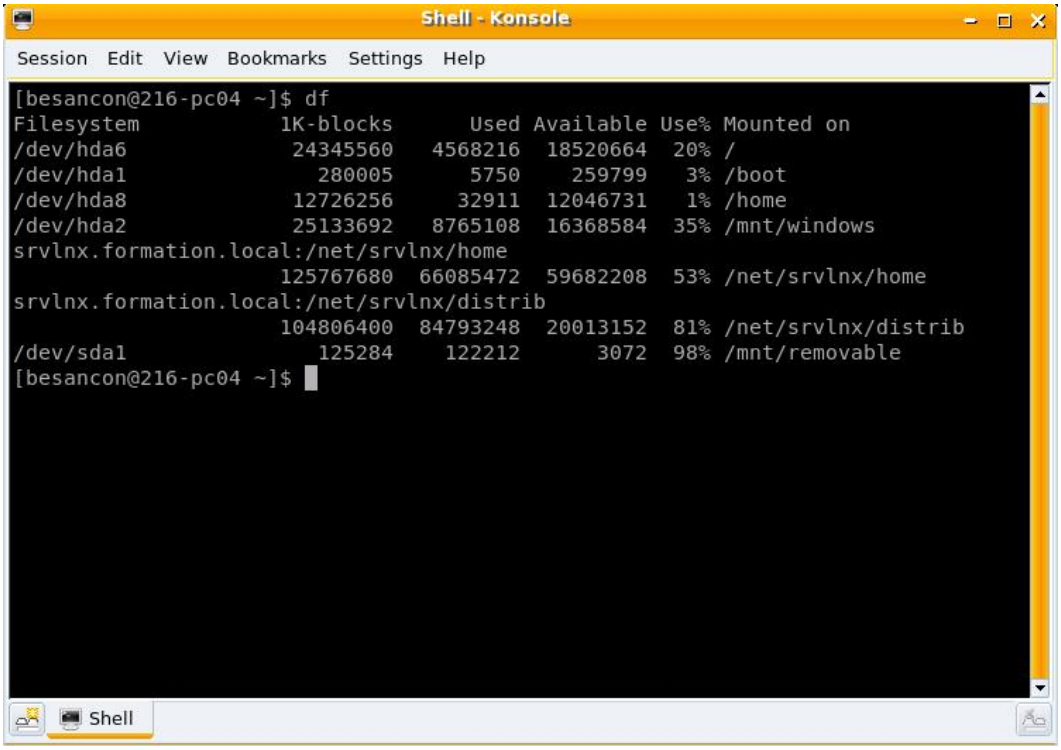
## ◇ Mandriva 2007 : l'explorateur de fichiers s'ouvre



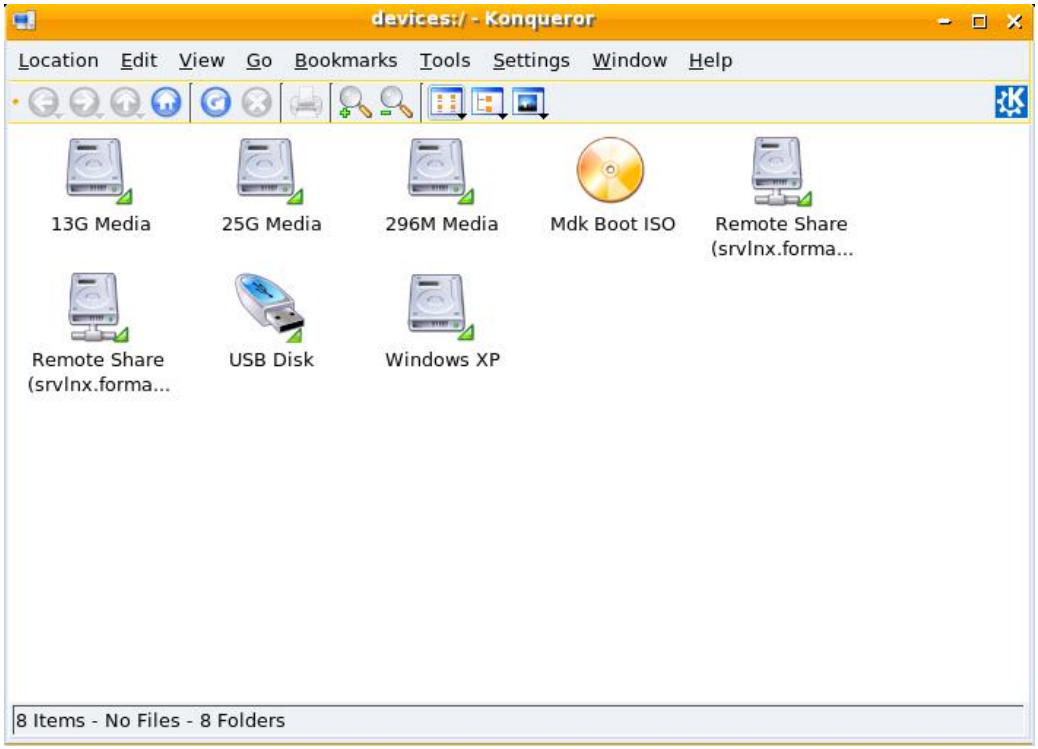





## ◇ Mandriva 2007 : montage automatique de la clef



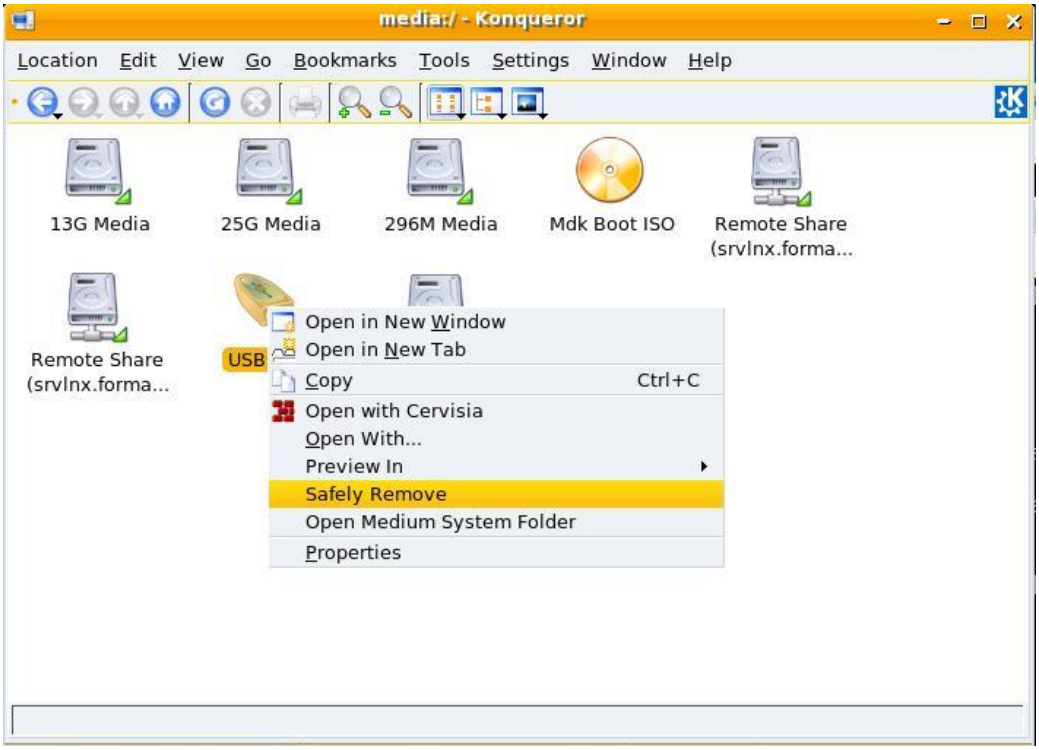
## ◇ Mandriva 2007 : présence de la clef dans les devices



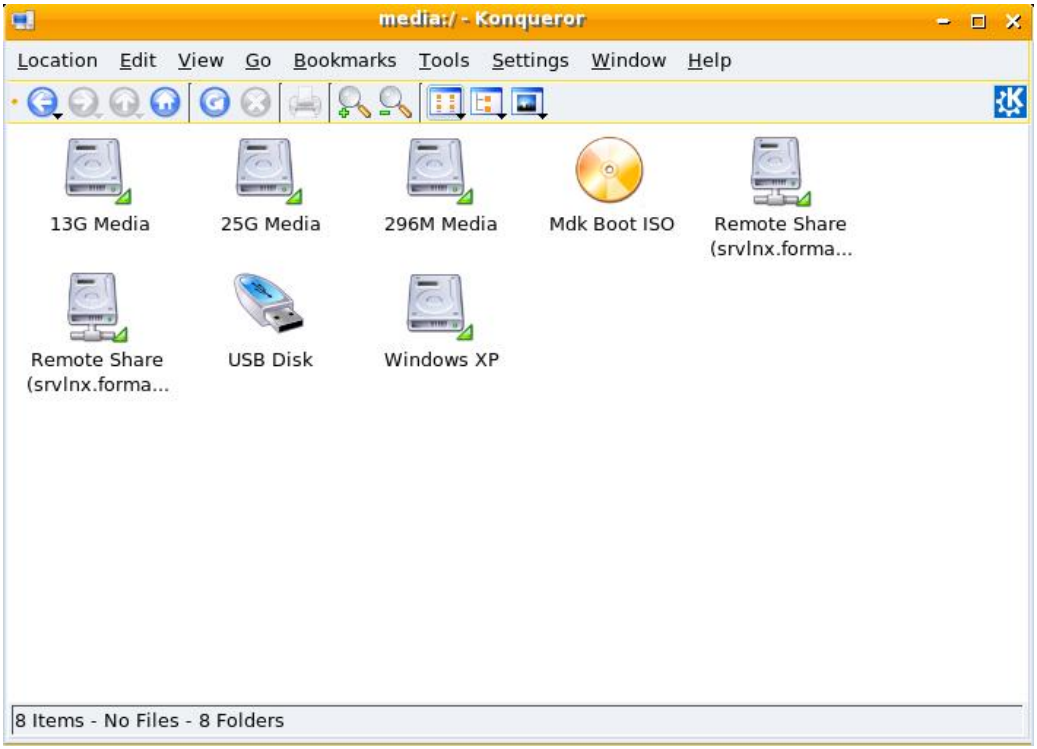





## ◇ Mandriva 2007 : démontage de la clef (1)



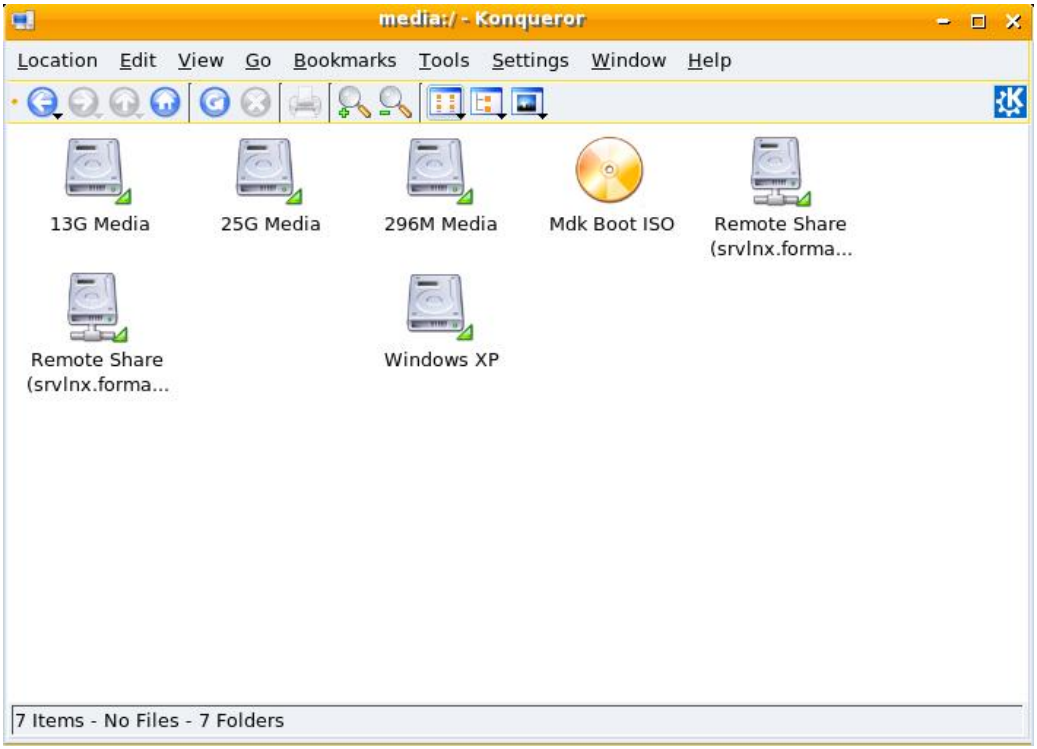
## ◇ Mandriva 2007 : démontage de la clef (2)



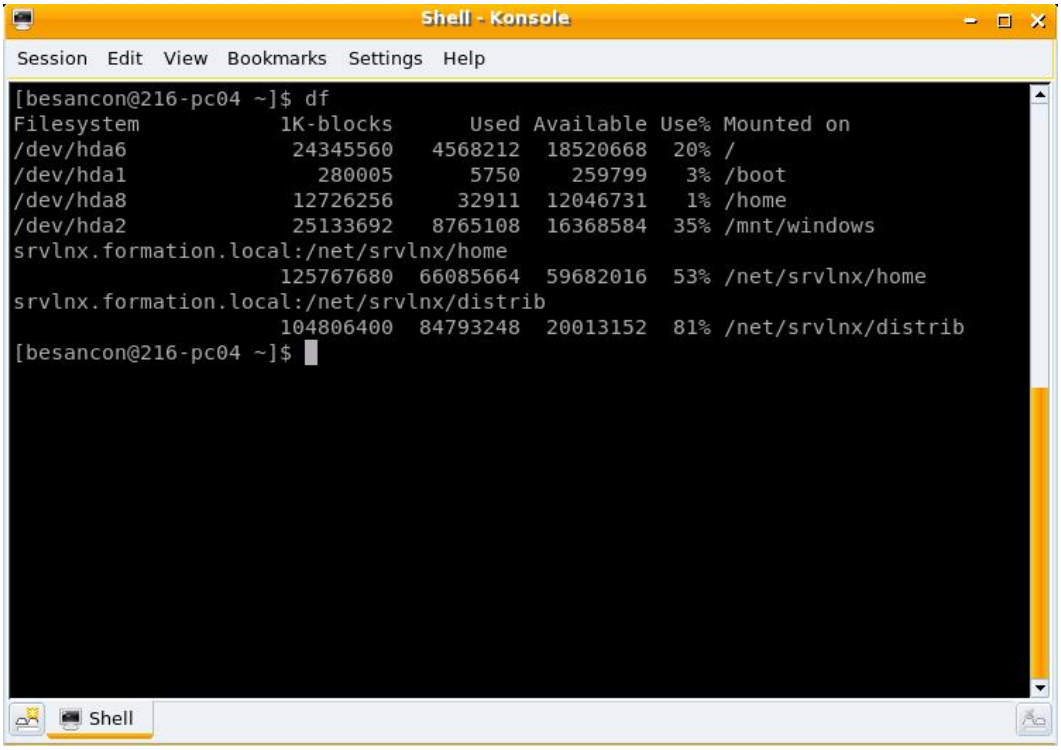





## ◇ Mandriva 2007 : démontage de la clef (3)



## ◇ Mandriva 2007 : démontage de la clef (4)








La clef USB doit être formatée en FAT ou FAT32.

OK mais ça ne marche pas tout le temps sous LINUX alors que cela marche parfaitement sous WINDOWS ! Pourquoi ?

Sous LINUX, la clef USB doit être formatée en mode disque dur et doit comporter par conséquent une table de partition.
Sous LINUX, une clef USB formatée en mode disquette (c'est-à-dire sans table de partition) ne sera pas reconnue par le système LINUX de la salle de TP de la Formation Permanente.

## Exemple d'une clef USB bien formatée :

```
% dmesg
...
usb 5-6: new high speed USB device using address 8
hub 5-6:1.0: USB hub found
hub 5-6:1.0: 1 port detected
usb 5-6.1: new high speed USB device using address 9
scsi3 : SCSI emulation for USB Mass Storage devices
  Vendor: USB 2.0   Model: Flash Disk        Rev: 1.00
  Type:   Direct-Access                      ANSI SCSI revision: 02
SCSI device sdb: 1015808 512-byte hdwr sectors (520 MB)
sdb: Write Protect is off
sdb: Mode Sense: 00 26 00 00
sdb: assuming drive cache: write through
 /dev/scsi/host3/bus0/target0/lun0: p1
Attached scsi removable disk sdb at scsi3, channel 0, id 0, lun 0
USB Mass Storage device found at 9
usb 5-6: USB disconnect, address 8
usb 5-6.1: USB disconnect, address 9
```

Il y a une table de partition avec une unique partition primaire.





## Exemple d'une clef USB mal formatée :

```
% dmesg
...
usb 5-7: new high speed USB device using address 2
hub 5-7:1.0: USB hub found
hub 5-7:1.0: 1 port detected
usb 5-7.1: new high speed USB device using address 3
SCSI subsystem initialized
Initializing USB Mass Storage driver...
scsi0 : SCSI emulation for USB Mass Storage devices
  Vendor: Sony      Model: Storage Media     Rev: PROL
  Type:   Direct-Access                      ANSI SCSI revision: 02
USB Mass Storage device found at 3
usbcore: registered new driver usb-storage
USB Mass Storage support registered.
SCSI device sda: 512000 512-byte hdwr sectors (262 MB)
sda: Write Protect is off
sda: Mode Sense: 00 06 00 00
sda: assuming drive cache: write through
 /dev/scsi/host0/bus0/target0/lun0: unknown partition table
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
usb 5-7: USB disconnect, address 2
usb 5-7.1: USB disconnect, address 3
```

Il n'y a pas de table de partition. La clef n'est pas vue sous LINUX.

Correction du problème de la clef mal formatée :

1. Sauver la clef en la vidant sur une machine WINDOWS (car la clef fonctionnera sous WINDOWS)
2. Lancer « `fdisk` » pour créer une table de partitions correcte. Toutes les données seront perdues, ATTENTION.
3. Déposer un filesystem FAT32 sur la partition primaire.





```
# fdisk /dev/sda

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

```
Command (m for help): l

 0  Empty           1c  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden W95 FAT1 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
 9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 b  W95 FAT32       50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  W95 FAT32 (LBA) 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 e  W95 FAT16 (LBA) 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  W95 Ext'd (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden W95 FAT3 65  Novell Netware  b8  BSDI swap       ff  BBT
```





```
Command (m for help): o  <-- Mise en place d'une table vide de partition
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(

Command (m for help): w  <-- Enregistrement de la table de partitions
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

A ce moment là, « `fdisk` » se termine. Il faut le relancer.

```
[root@216-pc01 z]# fdisk /dev/sda

Command (m for help): p  <-- Affichage de la table de partitions

Disk /dev/sda: 262 MB, 262144000 bytes
9 heads, 56 sectors/track, 1015 cylinders
Units = cylinders of 504 * 512 = 258048 bytes

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n  <-- Création d'une partition
Command action
   e   extended
   p   primary partition (1-4)
p  <-- Création d'une partition primaire
Partition number (1-4): 1  <-- Numéro de la partition primaire
First cylinder (1-1015, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1015, default 1015):
Using default value 1015
```





```
Command (m for help): p  <-- Affichage de la table de partitions

Disk /dev/sda: 262 MB, 262144000 bytes
9 heads, 56 sectors/track, 1015 cylinders
Units = cylinders of 504 * 512 = 258048 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        1015      255779+  83  Linux

Command (m for help): t  <-- Changement du type de la partition
Selected partition 1
Hex code (type L to list codes): b  <-- Type FAT32 de la partition
Changed system type of partition 1 to b (W95 FAT32)

Command (m for help): p  <-- Affichage de la table de partitions

Disk /dev/sda: 262 MB, 262144000 bytes
9 heads, 56 sectors/track, 1015 cylinders
Units = cylinders of 504 * 512 = 258048 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        1015      255779+   b  W95 FAT32
```

```
Command (m for help): w  <-- Enregistrement de la table de partitions
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

A ce moment là, « `fdisk` » se termine mais c'est terminé pour nous aussi. Reste à formater par « `mkdosfs` » la partition primaire créée.





Le mécanisme des quotas impose à l'utilisateur des limites sur :

1. le nombre de blocs disque utilisables **par filesystem**
2. le nombre d'inodes utilisables **par filesystem**

Une limite est en fait composée d'un couple de valeurs :

- **limite soft** :
  Aucune alarme n'est activée tant qu'on reste en dessous de cette limite. Si l'on dépasse ce seuil, il y a une alarme qui est activée. Des messages d'erreur apparaissent. L'utilisateur a 7 jours pour revenir en dessous de ce seuil. Au delà de 7 jours, toute consommation de ressources supplémentaires est bloquée.
- **limite hard** :
  C'est un seuil supérieur indépassable. L'utilisateur s'il dépasse la limite soft peut continuer de consommer des ressources à concurrence de la limite hard.

Par exemple :

```
% quota -v besancon
Disk quotas for besancon (uid 4332):
Filesystem   usage  quota   limit  timeleft  files  quota  limit  timeleft
/adm         98696  80000  100000  7.0 days   4959   4000   5000  7.0 days
```

L'utilisateur « besancon » dépasse son quota soft et a 7 sept jours pour revenir en dessous des 2 limites soft.





Les commandes de gestion des quotas sont :

| Commande | Description |
|---|---|
| quota -v utilisateur | Affiche l'utilisation disque de l'utilisateur et son quota |
| quotacheck | Recalcule les quotas utilisés |
| quotaon | Active les quotas |
| quotaoff | Désactive les quotas |
| repquota | Rapport de situation des quotas |
| edquota | Modification du quota d'un utilisateur |

Pour avoir un rapport sur les quotas, utiliser « repquota » (en anglais *report quota*)

```
# repquota -v -a
/dev/sd3g (/adm):
                        Block limits                        File limits
User            used  soft   hard  timeleft  used  soft  hard  timeleft
besancon  ++  98696 80000 100000  7.0 days  4959  4000  5000  7.0 days
```





L'activation du système de quotas nécessite les points suivants :

1. Le noyau doit supporter le mécanisme des quotas. En effet l'aspect comptabilité incombe au noyau.
2. Le filesystem doit supporter les quotas.
3. Les quotas doivent être activés lors du montage de la partition :
   1. Préciser que le filesystem est monté avec les quotas activés au niveau de « /etc/fstab » ou équivalent :

      ```
      /dev/sd3g  /adm              4.2  rw,quota   1 4
      ```

   2. Lancer « quotaon ». De préférence pendant le boot de la station. Une opération « quotacheck » doit être lancée au préalable afin d'avoir un état cohérent des quotas. Cela peut ralentir le boot.
4. Les quotas doivent être entrés pour chaque utilisateur. C'est une opération automatisable au moyen d'un squelette type et de la commande « edquota ».

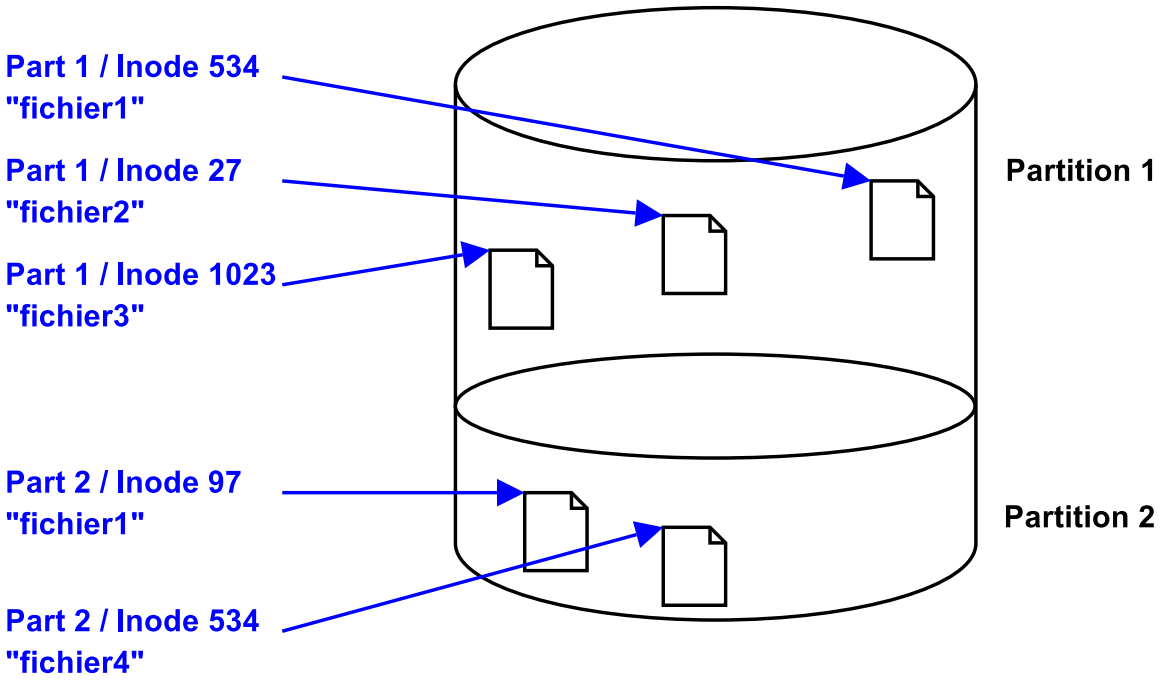
inode ≡ *Index Node*
**Un objet est identifié sur une partition de disque dur par son numéro d'inode (index node) et non pas par son nom.**
L'inode n'est unique qu'au sein d'une même partition.
Le système manipule en fait les objets via un couple *(partition, inode)*.

**ATTENTION : le nom d'un objet n'est pas stocké dans l'inode. Le nom d'un objet est stocké dans le répertoire.**
Un répertoire ≡ liste de couples (nom d'objet, numéro d'inode)





Les inodes sont gérés par partition.
On peut retrouver le même numéro d'inode dans plusieurs partitions :



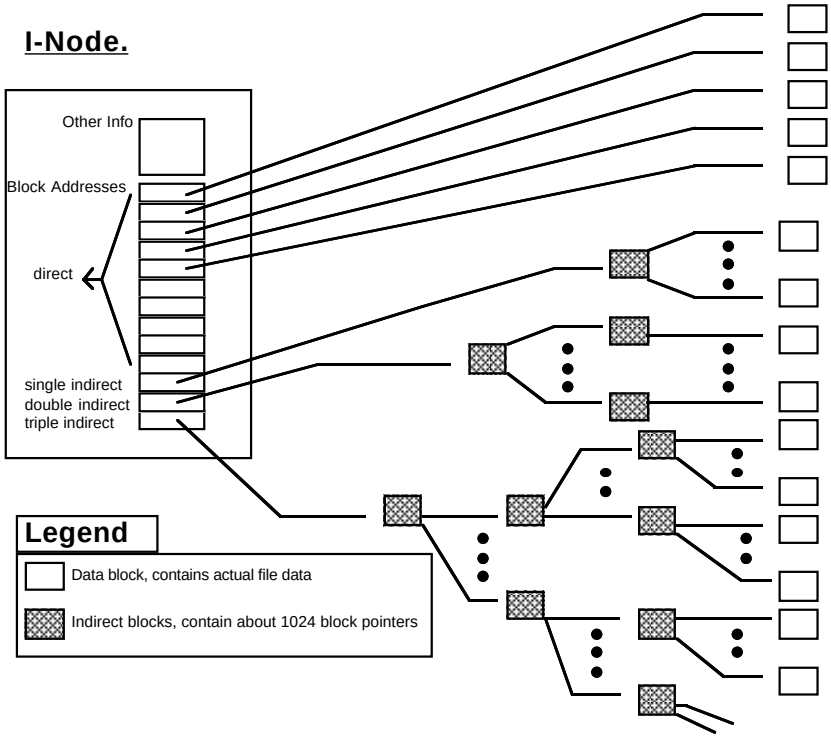
La structure inode (cf « `<sys/ufs/dinode.h>` ») décrit un objet de la façon suivante :

- type de l'objet
- droits d'accès
- nombre de liens ; quand il passe à 0 et que l'objet n'est plus utilisé, l'objet est détruit et la place est récupérée.
- UID, GID
- taille
- dates de dernière modification, consultation etc.
- localisation sur le disque des blocs de l'objet





La structure inode (environ 128 octets) ne peut pas contenir la liste complète de tous les blocs utilisés. On utilise donc le principe de **liste indirecte** :



Un bloc a pour taille 4 ko (4096 octets). Une adresse de bloc occupe 4 octets.

L'inode contient 13 numéros de blocs :

- les 12 premiers sont les numéros des 12 premiers blocs de données du fichier ;
- le suivant indique le numéro d'un bloc **indirect** : il contient les numéros des blocs de données à partir du onzième. Il adresse donc 1024 blocs.
- le douzième numéro est le numéro d'un bloc. Celui-ci contient 1024 numéros de blocs **indirects**, c'est-à-dire que chacun de ces 1024 blocs adressent un bloc de données ;
- le treizième numéro suit cette logique avec une **triple** indirection.

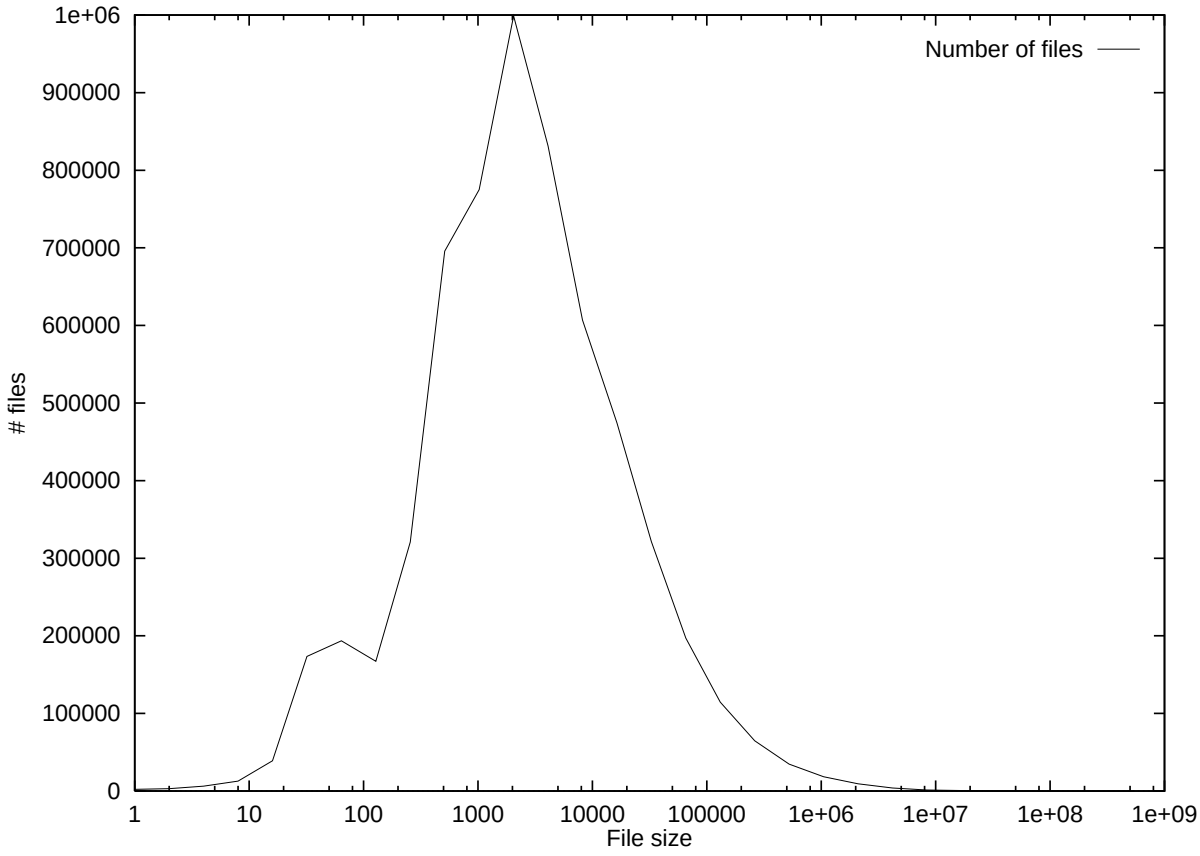




Ce schéma donne une taille limite de fichiers de :

$$(12 + 1024 + 1024^2 + 1024^3) \times 4096 \text{ octets} = 4\,402\,345\,721\,856 \text{ octets}$$

Le troisième niveau d'indirection est quasi inutilisé car le remplissage des blocs indirects et indirects de niveau 1 et 2 assure déjà une taille de fichiers $>$ à 4 Go.








(en anglais *display free*)

La commande « **df** » permet de voir le nombre d'inodes utilisées ou libres.

Syntaxe :
Sur LINUX et la plupart des UNIX traditionnels : « `df -i` »
Sur SUN Solaris : « `df -e` »

## ◇ Exemple sur LINUX :

```
% df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda3            2280320  318523 1961797   14% /
/dev/hda1              66264      50   66214    1% /boot
none                  193904       1  193903    1% /dev/shm
mailhost.example.com:/var/mail
                     7654398   17808 7636590    1% /var/mail
```





## ◇ Exemple sur SUN Solaris :

```
% df -e
Filesystem             ifree
/dev/dsk/c0t0d0s0      442832
/proc                    9788
mnttab                      0
/dev/dsk/c0t0d0s3      498452
swap                    59968
...
```

Points importants dans la logique du fonctionnement du filesystem :
**Certains directories ont des numéros d'inodes bien précis**

```
% /bin/ls -lai /
     2 drwxr-sr-x 22 root          512 Sep 11 15:11 .
     2 drwxr-sr-x 22 root          512 Sep 11 15:11 ..
     3 drwxr-xr-x  2 root         8192 Aug  8 17:38 lost+found
```

Le numéro d'inode 0 sert à marquer les inodes inutilisés.

Le numéro d'inode 1 est réservé.





(en anglais *lost+found* ≡ *bureau des objets trouvés*)

L'arborescence est un objet complexe :

- répertoires « `.` »
- répertoires « `..` »
- liste des blocs utilisés
- liste des blocs non utilisés

⇒ endommagement possible
⇒ vérification de la cohérence de l'arborescence par la commande « `fsck` » (voir page 502)

Le directory « `lost+found` » dans chaque partition stockera les inodes diagnostiqués par « `fsck` » comme présentant un problème.

La taille du directory (vide) est suffisamment grande pour y stocker dans les conditions précaires du boot des inodes endommagés :

```
% ls -l /lost+found
-rw-------  1 root     sys            0 Mar  1  1996 #155782
drwxr-xr-x  2 mailman  101          512 Jun  2  2000 #091584
```





Astuce : garder une trace des inodes pour s'en sortir en cas de problème

Comment : via « `find /partition -ls > MANIFEST.txt` »

```
% find /partition -ls
     2     1 drwxr-xr-x   9 root     root          512 Mar 29 04:30
     3     8 drwx------   2 root     root         8192 Aug  8  2001
  6400     1 drwxr-xr-x   2 besancon root         1024 Nov 28  2007
147204     1 drwxr-xr-x   4 besancon root          512 Dec  5 23:3
1248181    1 drwxr-xr-x   5 besancon adm           512 Dec  5 01:
1248183    4 drwxr-xr-x   2 besancon adm          4096 Nov 28 14:
1248185 2888 -rwxr-xr-x   1 besancon adm       2944062 Nov 28 14:
1248186 3000 -rwxr-xr-x   1 besancon adm       3060976 Nov 28 14:
...
```

Points importants dans la logique du fonctionnement du filesystem : **Le directory « `lost+found` » (inode 3) est crée automatiquement par la commande « `newfs` » dans la racine de la partition**

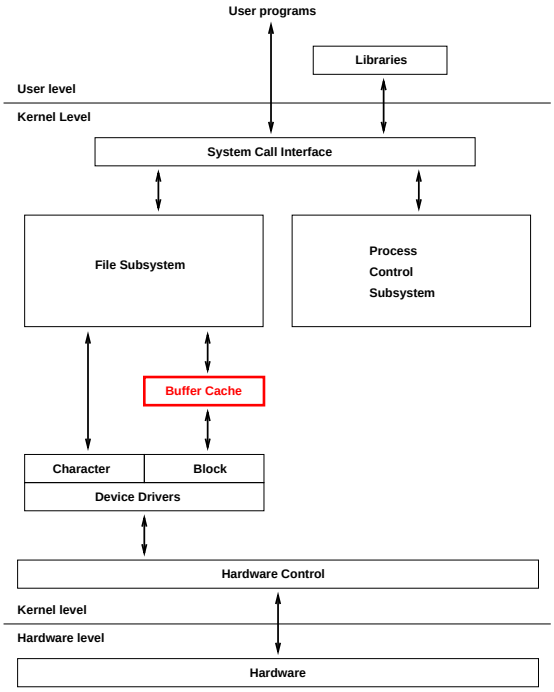
Il ne faut pas détruire ce directory ni essayer de le reconstruire manuellement par « `mkdir` ».

Selon le constructeur, il existe ou pas une commande spéciale pour reconstruire ce directory.





Rappel :



Quelques constatations :

1. Les filesystems sont constamment modifiés.
2. Il peut arriver que des informations se perdent en cas de dysfonctionnement du matériel.
3. Il y a le cache buffer.
   Le buffer cache conserve des données dont on retarde l'écriture sur disques (exception faite pour les répertoires, les superblocks, etc.).

⇒ besoin de provoquer l'écriture sur disque du contenu du buffer cache.
⇒ commande manuelle « **sync** » pour cela.
Lors de l'ecriture manuelle sur disque du buffer cache, le filesystem est marqué FSCLEAN.





La commande manuelle « **sync** » provoque l'écriture sur disque du contenu du buffer cache.

Le système UNIX exécute ce mécanisme d'écriture forcée du contenu du buffer cache **toutes les 30 secondes**.

C'est le programme « `update` » qui réalise cela.
Autres noms possibles : « `syncher` »

Lors de l'écriture chronique sur disque du buffer cache, le filesystem est marqué FSCLEAN.

(en anglais *File System Check Consistency*)

La perte du contenu du buffer cache en cas de dysfonctionnement peut conduire à effectuer des réparations par la commande **fsck (File System Check Consistency)**.

Lors du boot d'une machine, si un filesystem est marqué FSCLEAN, fsck ne vérifie pas ce filesystem. On se trouve dans ces conditions si la machine a été arrêtée ou redémarrée en utilisant des commandes comme `fastboot`, `fasthalt` etc.





Plusieurs vérifications sont effectuées par « `fsck` » :

- le super-block contient le bon nombre de blocs libres ;
- le super-block contient le bon nombre d'inodes libres ;
- le super-block contient la bonne taille de la partition ;
- un bloc de données appartient à un et un seul fichier, ou à la liste des blocs libres ;
- un inode, s'il est référencé *n* fois dans des répertoires, a bien *n* liens physiques ;
- un inode est soit utilisé, soit dans la liste des inodes libres ;
- la taille du fichier correspond au nombre de blocs alloués ;
- tout répertoire contient une entrée pour « . » et « .. » ;
- une entrée dans un répertoire doit référencer un inode non libre.

## Exemple quand tout se passe bien :

```
# fsck /dev/rsd2g
** /dev/rsd2g
** Last Mounted on /users
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
CLEAN FLAG NOT SET IN SUPERBLOCK
FIX? [yn] y

23423 files, 1174684 used, 2946897 free (6641 frags, 367532 blocks,
0.2% fragmentation)

***** FILE SYSTEM WAS MODIFIED *****
```





## Exemple en cas de problème :

```
Automatic reboot in progress...
/dev/rsd0a: clean, 234610 free (9626 frags, 28123 blocks, 2.6% fragmentat
/dev/rsd10c: clean, 842274 free (386 frags, 105236 blocks, 0.0% fragmenta
/dev/rsd12c: clean, 470238 free (822 frags, 58677 blocks, 0.1% fragmentat
/dev/rsd11c: clean, 155358 free (1406 frags, 19244 blocks, 0.1% fragmenta
/dev/rccd0c: clean, 3642368 free (15392 frags, 453372 blocks, 0.2% fragme
cannot alloc 8340482 bytes for lncntp
/dev/rccd1c: CAN'T CHECK FILE SYSTEM.
/dev/rccd1c: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
/dev/rccd2c: clean, 1759345 free (10417 frags, 218616 blocks, 0.1% fragme
cannot alloc 10629122 bytes for lncntp
/dev/rsd5c: CAN'T CHECK FILE SYSTEM.
/dev/rsd5c: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
THE FOLLOWING FILE SYSTEMS HAD AN UNEXPECTED INCONSISTENCY:
        /dev/rccd1c (/people), /dev/rsd5c (/hold)
Automatic file system check failed... help!
Enter root password, or ^D to go multi-user
```

Syntaxe :

`fsck [options] [fichier-spécial]`

Options intéressantes :

- option « `-y` » : Répond « yes » à toutes les questions que posent « fsck »
- option « `-n` » : Répond « no » à toutes les questions que posent « fsck »
- option « `-p` » : Répond « yes » à toutes les questions anodines que posent « fsck ». C'est ce qui est utilisé au moment du boot.

Quelques règles :

- Ne jamais lancer « `fsck` » sur une partition active, montée car cela peut conduire à un état instable du filesystem.
- Si possible, lancer « `fsck` » en étant en single user.
- En cas de `fsck` sur « `/` », redémarrer juste après sans sauver le buffer cache.



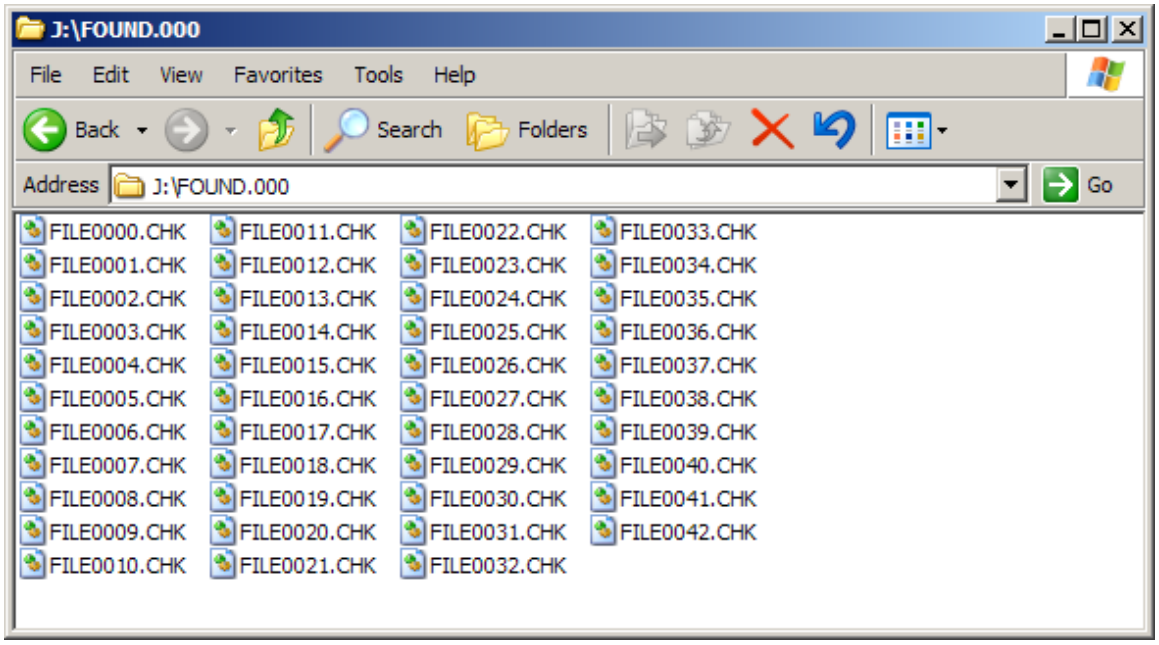


Présence d'un « `fsck` » là où l'on ne s'y attend pas toujours : démarrage d'un switch CISCO récent :

```
C2950 Boot Loader (C2950-HBOOT-M) Version 12.1(11r)EA1, RELEASE SOFTWARE (fc1)
Compiled Mon 22-Jul-02 17:18 by antonino
WS-C2950-24 starting...
Base ethernet MAC Address: 00:0b:46:eb:9b:c0
Xmodem file system is available.
Initializing Flash...
flashfs[0]: 17 files, 2 directories
flashfs[0]: 0 orphaned files, 0 orphaned directories
flashfs[0]: Total bytes: 7741440
flashfs[0]: Bytes used: 3961856
flashfs[0]: Bytes available: 3779584
flashfs[0]: flashfs fsck took 6 seconds.
...done initializing flash.
Boot Sector Filesystem (bs:) installed, fsid: 3
Parameter Block Filesystem (pb:) installed, fsid: 4
Loading "flash:/c2950-i6q4l2-mz.121-11.EA1.bin"...##############################
################################################################################
################################################################################
################################################################################
###########

File "flash:/c2950-i6q4l2-mz.121-11.EA1.bin" uncompressed and installed, entry
point: 0x80010000
executing...
```

Le principe de FSCK existe aussi dans le monde WINDOWS :








Session de correction d'un filesystem de clef USB incohérent :

```
C:\>chkdsk J: /F
The type of the file system is FAT.
Volume 82SU064D32 created 2/25/2002 9:37 AM
Volume Serial Number is 04EC-2BFA
The specified disk appears to be a non-Windows XP disk.
Do you want to continue? (Y/N) y
Windows is verifying files and folders...
\20050401\sun is cross-linked on allocation unit 1156.
Folder truncated.
\20050401\system is cross-linked on allocation unit 1670.
Folder truncated.
\20050401\php is cross-linked on allocation unit 1695.
Folder truncated.
\20050321\52.html is cross-linked on allocation unit 5901.
Cross link resolved by copying.
File and folder verification is complete.
Lost chain cross-linked at cluster 29382.  Orphan truncated.
```

```
Convert lost chains to files (Y/N)? y
3276800 bytes in 43 recovered files.
Windows has made corrections to the file system.

   65 198 080 bytes total disk space.
       59 392 bytes in 55 folders.
   64 208 896 bytes in 331 files.
      929 792 bytes available on disk.

        1 024 bytes in each allocation unit.
       63 670 total allocation units on disk.
          908 allocation units available on disk.
```





La stratégie d'allocation des blocs au sein d'un filesystem ne fonctionne que si le disque contient un espace libre raisonnable. Cet espace est appelé **minfree**. La réserve est de 10% d'ordinaire :

```
% df -t ufs /tmp
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/sd0s1f    127151    22843    94136    20%    /tmp
```

Ceci explique que l'on puisse remplir une partition à plus de 100% :

```
# df -t ufs /mnt
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/sd0s1d   1702685  1638403   -71932   105%    /mnt
```

La commande **tunefs** permet de régler ce pourcentage :

`tunefs -m minfree raw-filesystem`

(démonter le filesystem avant le tunefs puis le remonter après)

Snapshot ≡ cliché instantané du filesystem (terme de photographie)
Le snapshot montre un filesystem figé alors que le filesystem continue de se modifier.

Avantages :

- snapshot rapide à construire
- backup d'une image stable d'un filesystem
- récupération facile et simple d'un fichier





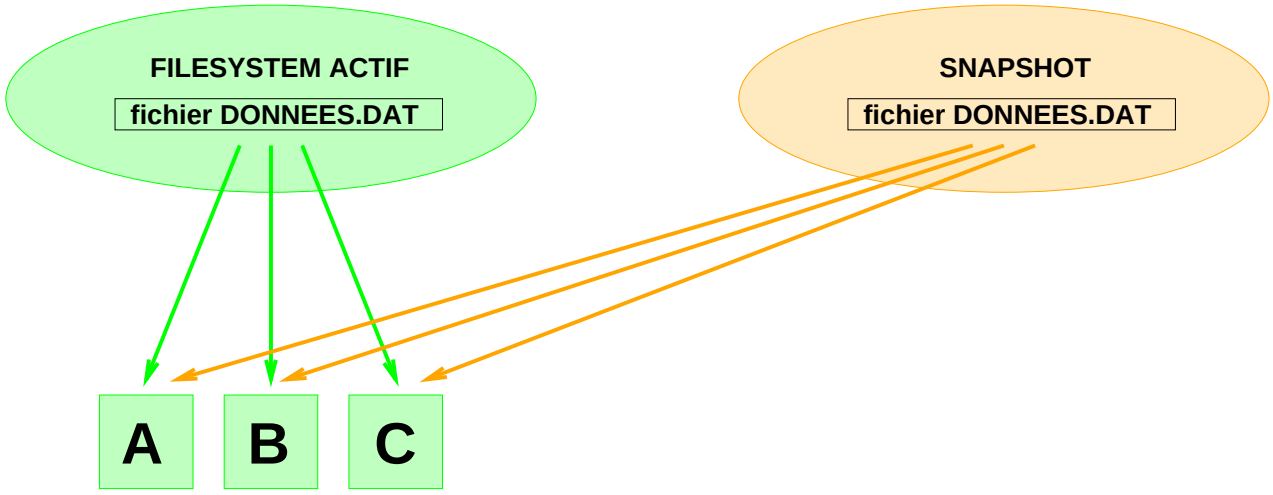
Disponible à partir de Solaris 8 (commande `fssnap`)
Mais ne subsiste pas après un reboot...

Disponible à partir de FreeBSD 5 (option de `/etc/fstab` : `snapshot`)
Mais plante le noyau d'une machine multiprocesseur en 5.0...

Disponible à partir de Linux si on utilise un Logical Volume Manager.

Disponible sur les machines spécialisées en stockage de la marque Network Appliance (`http://www.netapp.com`)

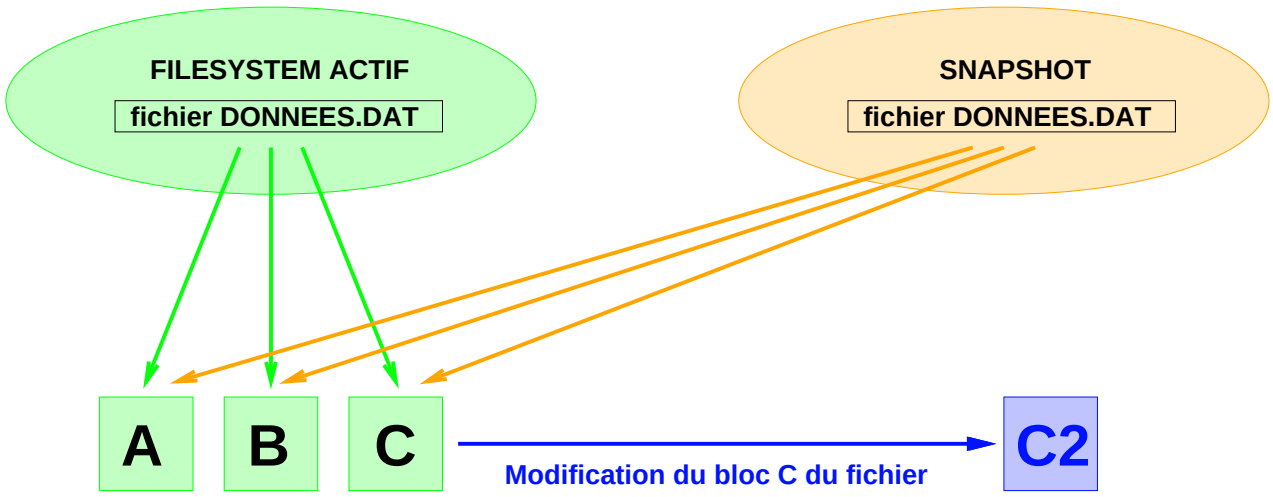
Principe : on construit le snapshot en notant les blocs disques utilisés.



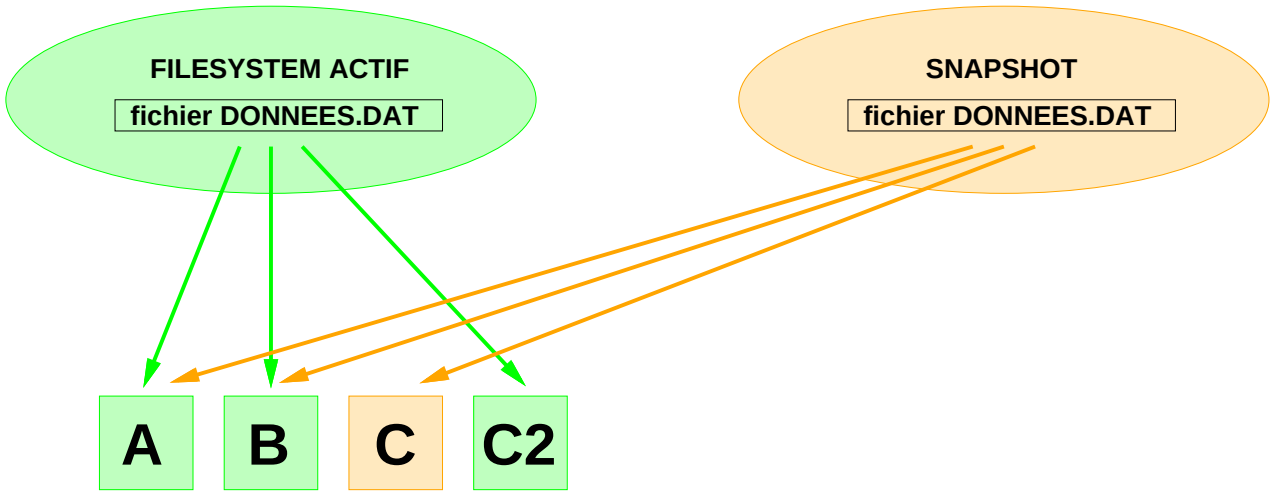





Le filesystem peut continuer à évoluer. Un fichier modifie ainsi ses blocs.



Le filesystem est conçu pour travailler conjointement avec les snapshots. Il va dupliquer le bloc que l'on veut modifier car le filesystem sait que ce bloc est associé à un snapshot. Il modifier la copie du bloc. Le snapshot pointe toujours sur les mêmes blocs.

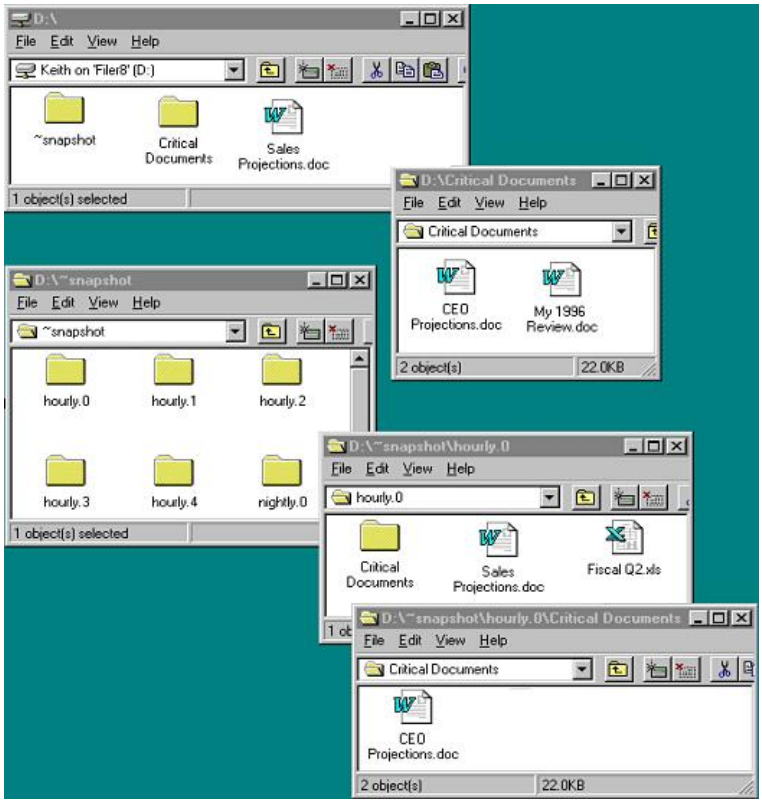



**Nécessité de gérer le nombre et la durée de vie des snapshots !**





Sur les machines Network Appliance, présence d'un répertoire « `.snapshot` » dans chaque répertoire. Il contient les snapshots horaires par défaut :



Disponible en Windows 2003 server : *Volume Shadow-Copy Service* dit VSS

VSS :

- serveur de fichiers Windows 2003 server
- clients Windows XP ou Windows 2003 server
- logiciel Shadow-Copy sur les clients

cf `http://www.laboratoire-microsoft.org/articles/win/volume_shadow_copy/0/`

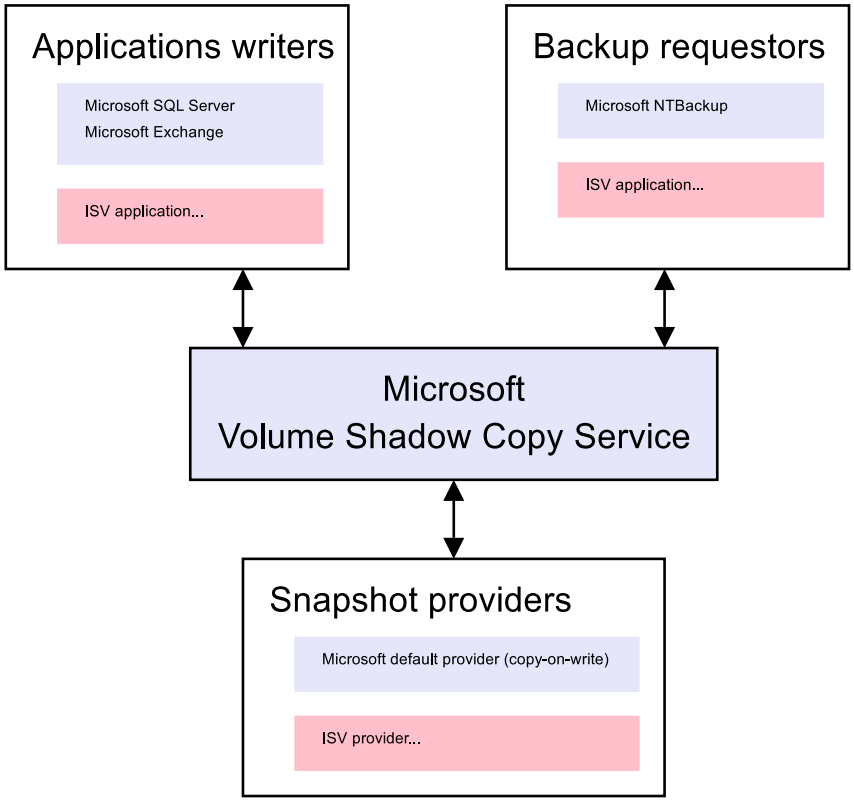




Approche différente des systèmes UNIX :

- sur UNIX : le snapshot est fait indépendamment de l'application ; un snapshot d'une base de données s'il est invalide va conduire la base de données à faire une réparation du snapshot ⇒ délai de retour à la normale. . .
- sur Windows : le snapshot sera toujours « sain » ; une application pourra se fier à un snapshot et l'utiliser comme point de reprise immédiat.

Schéma de principe de VSS :








◇ Application Writers

Les applications sont les utilisatrices des snapshots car elles savent mieux que le filesystem les fichiers et l'importance des fichiers qu'elles utilisent.
Il faut donc rendre une application compatible avec VSS moyennant un kit de développement Microsoft, « Microsoft snapshot SDK » (disponible en NDA).

Les applications ainsi modifiées sont des *Snapshot Writers* :

> *L'application sera responsable de retenir l'activité en écriture le temps de tenter la création du snapshot.*
>
> - *Si l'application n'a pas pu retenir le flux d'écriture, alors le snapshot n'est pas sain mais l'application le sait.*
> - *Si le flux d'écriture a pu être retenu, le snapshot est sain et l'application le sait.*

◇ Backup requestors

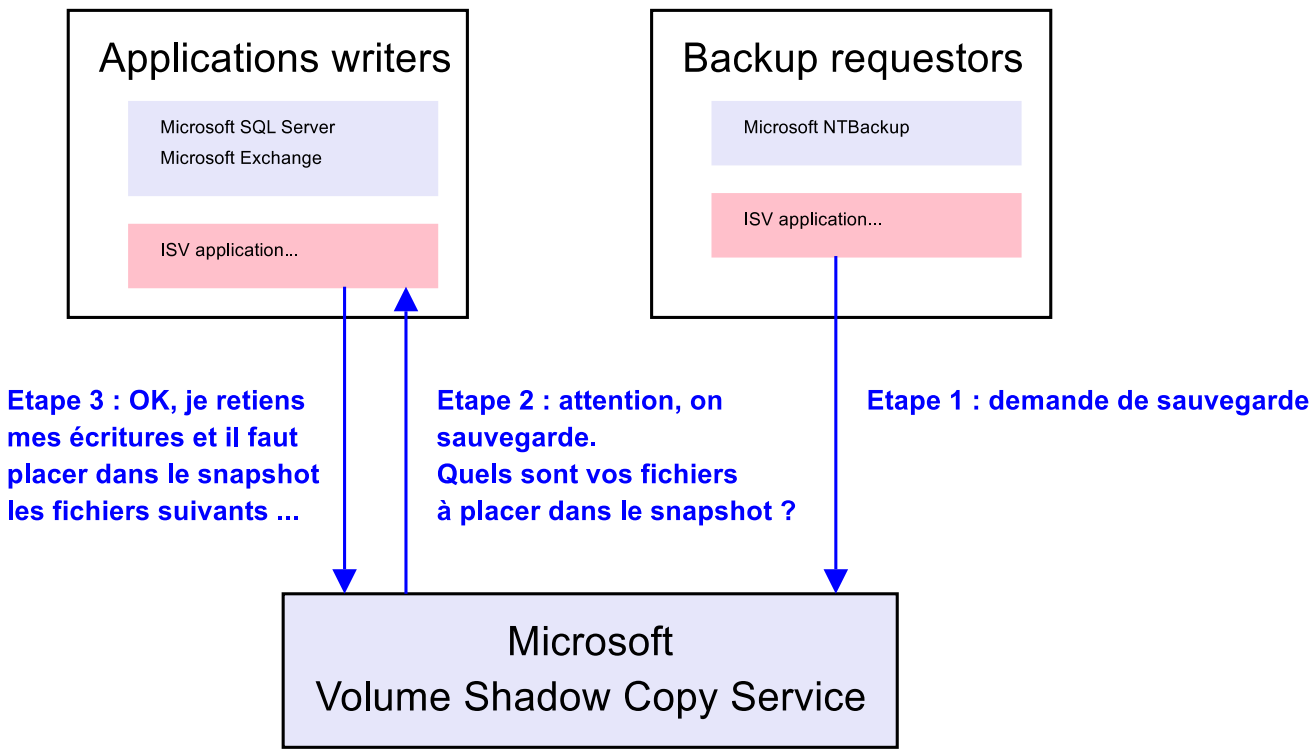
Des programmes externes comme `NTBackup.exe` peuvent demander la sauvegarde ou la restoration de la machine.
Ces programmes de « backup » doivent être modifiés au moyen du « Microsoft snapshot SDK » pour être rendus compatibles avec VSS.

Principe de la sauvegarde : voir pages suivantes
Principe de la restoration : voir pages suivantes

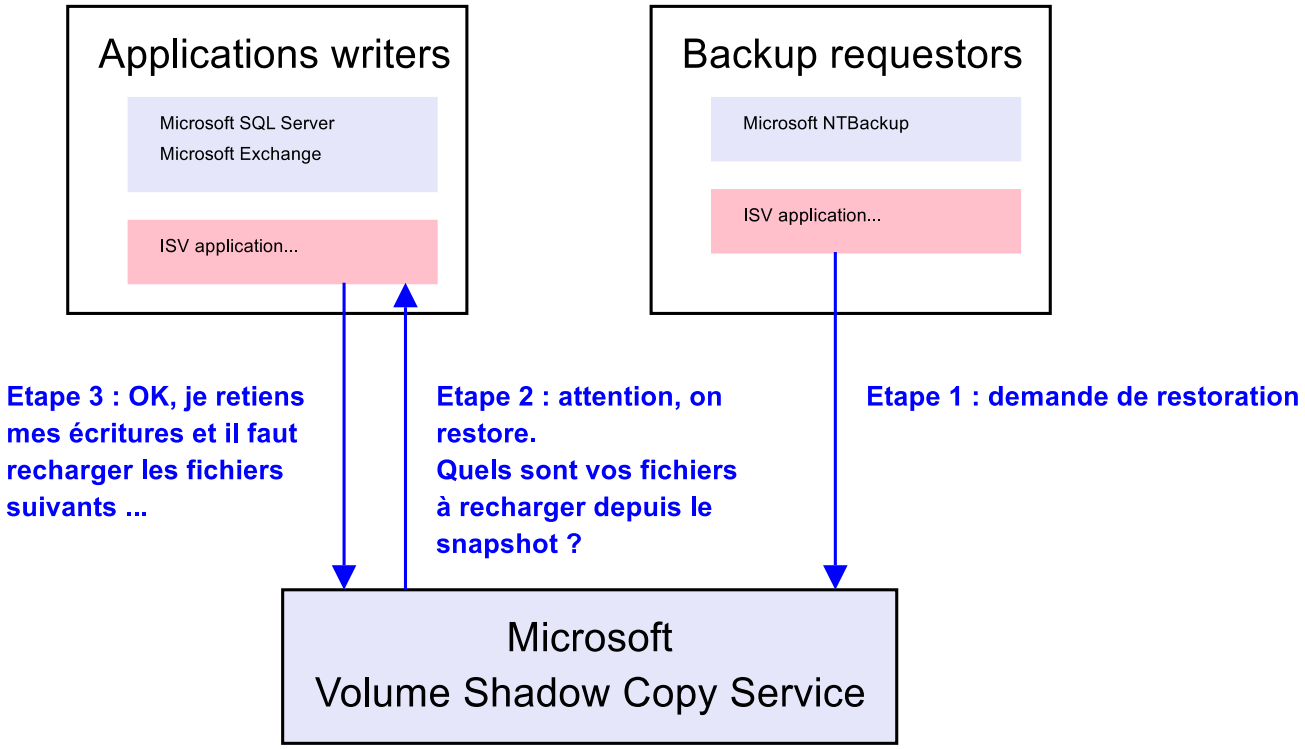




Principe de la sauvegarde :



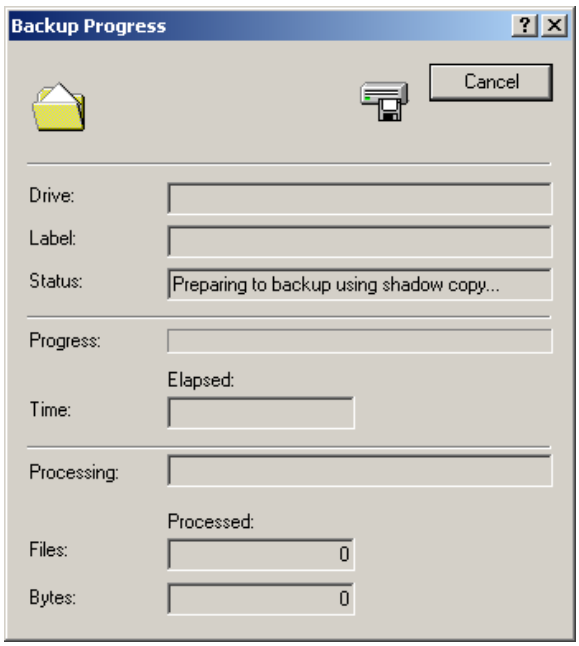
Principe de la restoration :








Mention de VSS dans une fenêtre transitoire de `NTBackup.exe` de Windows XP :



Disponible par exemple sur les UNIX IBM AIX, SUN Solaris, COMPAQ Tru64 Unix, LINUX

Un LVM est un gestionnaire de volumes logiques qui fonctionne le plus souvent dynamiquement :

- création dynamique de filesystems
- supporte davantage que les 8 partitions classiques d'un seul disque dur
- redimensionnement dynamique de filesystems actifs
- suppression de filesystems
- ajout de disques online

Un LVM est un avantage : meilleure flexibilité pour allouer de la place disque.
Un LVM est complexe à mettre en place et délicat à manipuler.
Un LVM est spécifique à chaque UNIX.

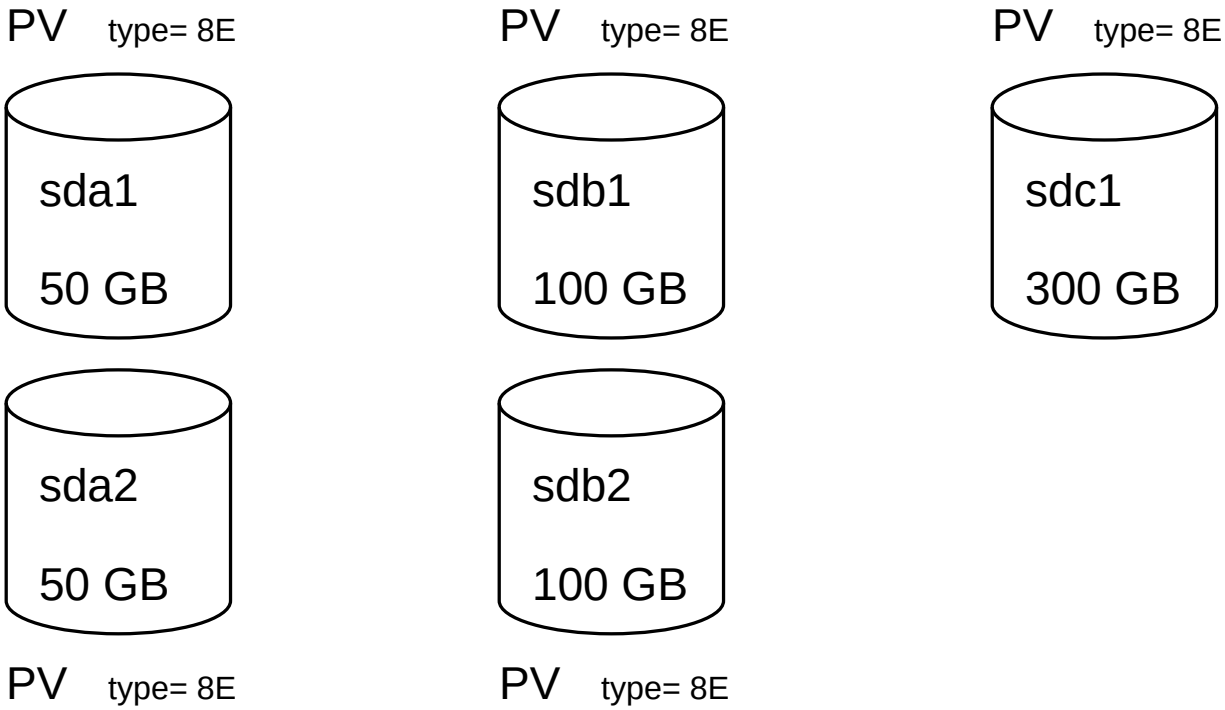




## ◇ LVM sous LINUX : Physical Volume

Notion de **Physical Volume** (PV) : disque entier ou partition

Type de la partition : `8E` (voir page 474)



## Initilisation du LVM : « vgscan »

```
# vgscan
  Reading all physical volumes.  This may take a while...
  No volume groups found
```





## Création de PV : « pvcreate device »

```
# fdisk -l /dev/sdb
Disk /dev/sdb: 536 MB, 536870912 bytes
64 heads, 32 sectors/track, 512 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         256      262128   8e  Linux LVM
/dev/sdb2             257         512      262144   8e  Linux LVM

# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

## Liste des PV : « pvdisplay »

```
# pvdisplay
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               255.81 MB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               zb89Gr-oeOv-Vm0Q-P2nh-y8PW-0udm-eunT0d

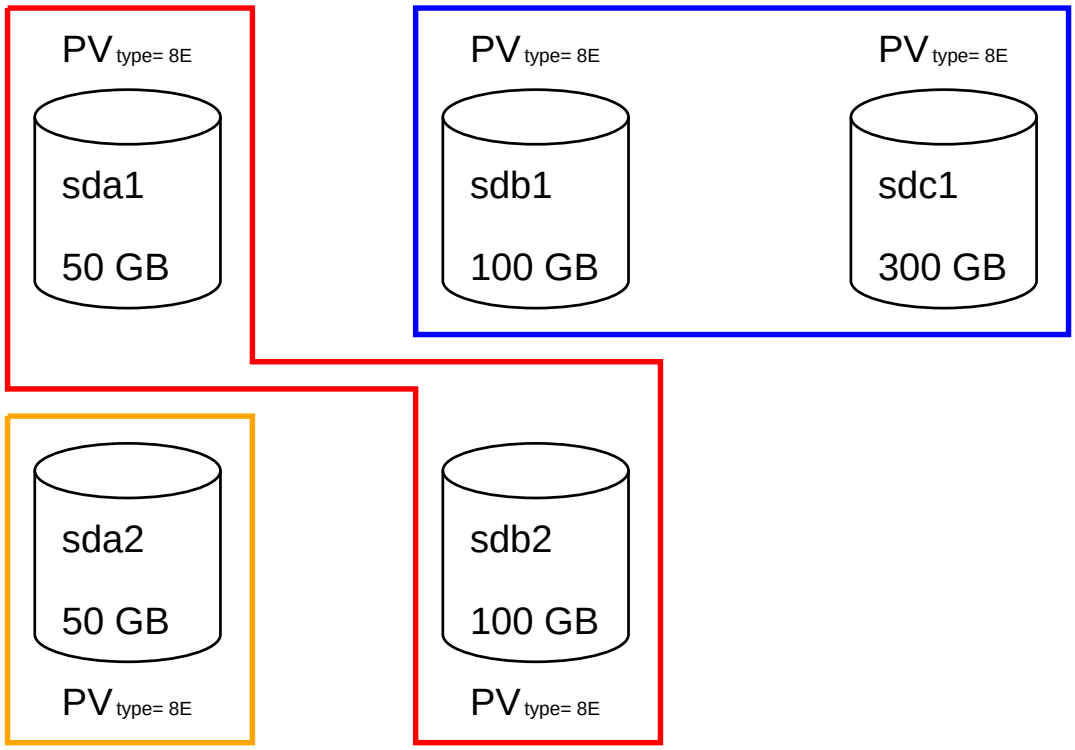
# pvdisplay -s
  Device "/dev/sdb1" has a capacity of 255.81 MB
```

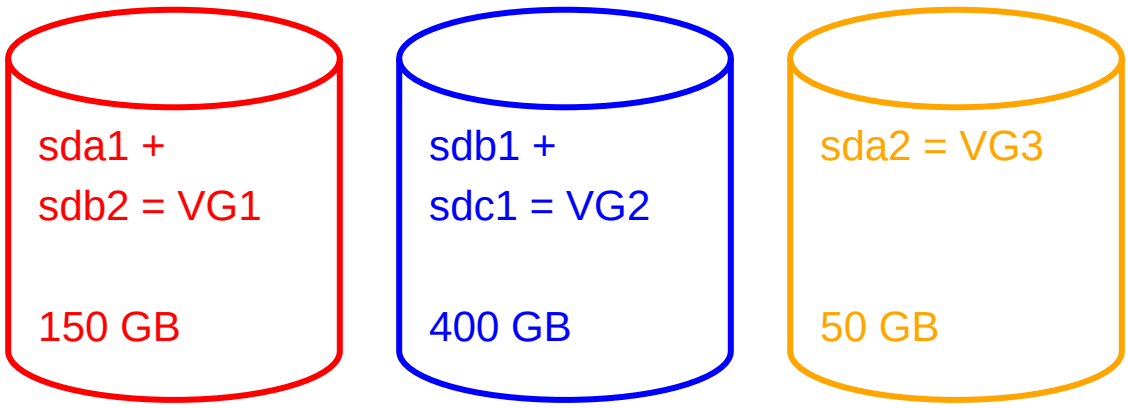




## ◇ LVM sous LINUX : Volume Group

On aggrége plusieurs PV pour donner un **Volume Group** (VG).










## Création de VG : « `vgcreate VG PV` »

```
# vgcreate JARDIN /dev/sdb1
  Volume group "JARDIN" successfully created
```

## Liste des VG : « vgdisplay »

```
# vgdisplay
  --- Volume group ---
  VG Name               JARDIN
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               252.00 MB
  PE Size               4.00 MB
  Total PE              63
  Alloc PE / Size       0 / 0
  Free  PE / Size       63 / 252.00 MB
  VG UUID               ZBRKTD-FD75-FuQC-cb1M-NtyI-Hw8x-rUQBM6
```



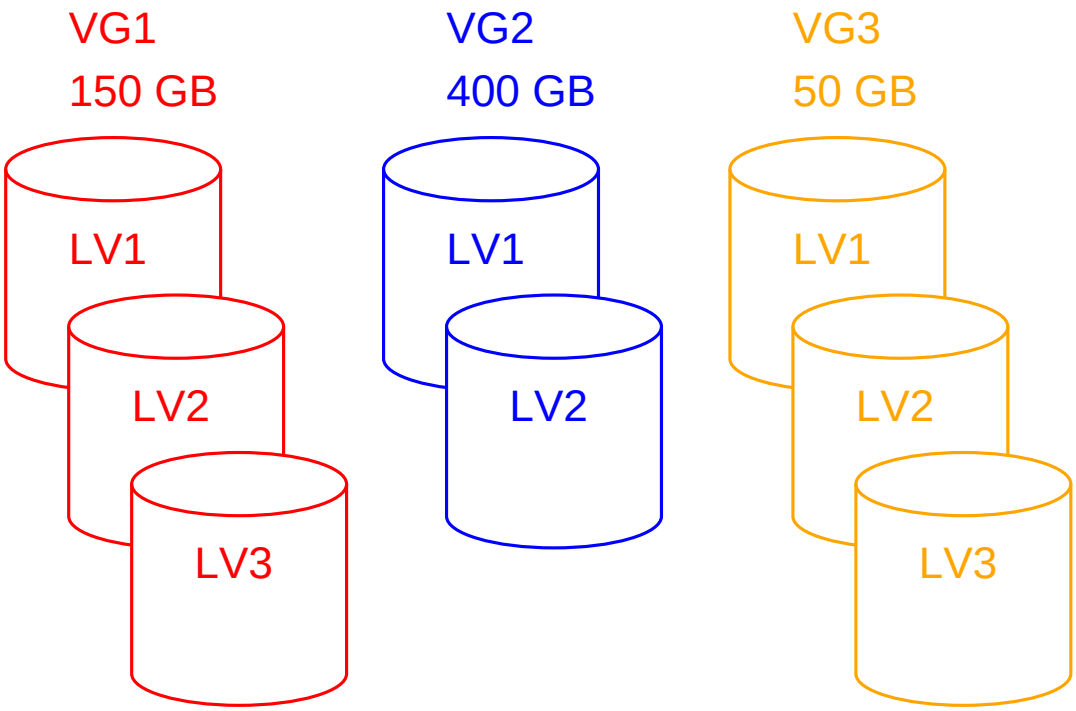


## Avec option « -s » :

```
# vgdisplay -s
  "JARDIN" 252.00 MB [0         used / 252.00 MB free]
```

## ◇ LVM sous LINUX : Logical Volume

On crée des partitions ou **Logical Volume** (LV) au sein des VG.








## Création de LV : « `lvcreate -L taille -n LV VG` »

```
# lvcreate -L 128M -n ANANAS JARDIN
  Logical volume "ANANAS" created
```

Variante :

```
# lvcreate -v -L 128M -n ANANAS JARDIN
    Finding volume group "JARDIN"
    Archiving volume group "JARDIN" metadata (seqno 1).
    Creating logical volume ANANAS
    Creating volume group backup "/etc/lvm/backup/JARDIN" (seqno 2).
    Found volume group "JARDIN"
    Creating JARDIN-ANANAS
    Loading JARDIN-ANANAS table
    Resuming JARDIN-ANANAS (253:0)
    Zeroing start of logical volume "ANANAS"
    Creating volume group backup "/etc/lvm/backup/JARDIN" (seqno 2).
  Logical volume "ANANAS" created
```

## Liste des LV : « lvdisplay »

```
# lvdisplay
  --- Logical volume ---
  LV Name                /dev/JARDIN/ANANAS
  VG Name                JARDIN
  LV UUID                QmFxba-zGw1-v1bw-ZbQo-FBbF-Zeai-SwJ2Er
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                128.00 MB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0
```

On notera le nom du LV : « /dev/JARDIN/ANANAS » ce qui indique la forme générique « /dev/VG/LV ».





## ◇ LVM sous LINUX : redimensionnement des VG

```
# vgdisplay
  --- Volume group ---
  VG Name               JARDIN
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               252.00 MB
  PE Size               4.00 MB
  Total PE              63
  Alloc PE / Size       32 / 128.00 MB
  Free  PE / Size       31 / 124.00 MB
  VG UUID               ZBRKTD-FD75-FuQC-cb1M-NtyI-Hw8x-rUQBM6
```

## L'ajout de PV est possible : « vgextend VG PV »

```
# vgextend JARDIN /dev/sdb2
  Volume group "JARDIN" successfully extended

# vgdisplay
  --- Volume group ---
  VG Name               JARDIN
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  5
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               504.00 MB
  PE Size               4.00 MB
  Total PE              126
  Alloc PE / Size       32 / 128.00 MB
```





## La suppression de PV est possible : « vgreduce VG PV »

```
# vgreduce JARDIN /dev/sdb2
  Removed "/dev/sdb2" from volume group "JARDIN"

# vgdisplay
  --- Volume group ---
  VG Name               JARDIN
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  6
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               252.00 MB
  PE Size               4.00 MB
  Total PE              63
  Alloc PE / Size       32 / 128.00 MB
```

## ◇ LVM sous LINUX : redimensionnement des LV

Les LV peuvent être dimensionnées à chaud :

« `lvextend -L taille LV` »

Ici passage de 128 MB à 384 MB :

```
# lvextend -L 384m /dev/JARDIN/ANANAS
  Extending logical volume ANANAS to 384.00 MB
  Logical volume ANANAS successfully resized

# lvdisplay
  --- Logical volume ---
  LV Name                /dev/JARDIN/ANANAS
  VG Name                JARDIN
  LV UUID                QmFxba-zGw1-v1bw-ZbQo-FBbF-Zeai-SwJ2Er
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                384.00 MB
  Current LE             96
  Segments               2
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:0
```





Les LV peuvent être redimensionnés à chaud :

« `lvextend -L -taille LV` »

Ici passage de 384 MB à 256 MB :

```
# lvreduce -L -128m /dev/JARDIN/ANANAS
  WARNING: Reducing active logical volume to 256.00 MB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce ANANAS? [y/n]: y
  Reducing logical volume ANANAS to 256.00 MB
  Logical volume ANANAS successfully resized
```

Attention : Réduire la taille de LV fait perdre les données.

Les LV peuvent être redimensionnés à chaud :
« `lvextend -L +taille LV` »

Ici passage de 256 MB à 384 MB :

```
# lvextend -L +128m /dev/JARDIN/ANANAS
  Extending logical volume ANANAS to 384.00 MB
  Logical volume ANANAS successfully resized
```





## ◇ LVM sous LINUX : redimensionnement des filesystems sur les LV

Les filesystems déposés sur les LV peuvent être redimensionnées à chaud s'ils le permettent.

Pour un filesystem EXT3 :

- démonter le filesystem
- faire « `e2fsck -f LV` »
- faire « `resize2fs LV` »

Ici le LV passe de 128 MB à 384 MB :

```
# mount /dev/JARDIN/ANANAS /mnt
# df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda3              1786792   1291048    404980  77% /
/dev/sda1                15521      3206     11514  22% /boot
none                    127984         0    127984   0% /dev/shm
/dev/mapper/JARDIN-ANANAS
                        126931      1550    118828   2% /mnt

# umount /mnt
# lvextend -L +256m /dev/JARDIN/ANANAS
  Extending logical volume ANANAS to 384.00 MB
  Logical volume ANANAS successfully resized
```





```
# e2fsck -f /dev/JARDIN/ANANAS
e2fsck 1.35 (28-Feb-2004)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/JARDIN/ANANAS: 11/32768 files (9.1% non-contiguous), 5691/131072 blc

# resize2fs /dev/JARDIN/ANANAS
resize2fs 1.35 (28-Feb-2004)
Resizing the filesystem on /dev/JARDIN/ANANAS to 393216 (1k) blocks.
The filesystem on /dev/JARDIN/ANANAS is now 393216 blocks long.

# mount /dev/JARDIN/ANANAS /mnt
# df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda3              1786792   1291052    404976  77% /
/dev/sda1                15521      3206     11514  22% /boot
none                    127984         0    127984   0% /dev/shm
/dev/mapper/JARDIN-ANANAS
                        380807      2054    363025   1% /mnt
```

Pour un filesystem REISERFS[1] :

- inutile de démonter le filesystem
- faire « `resize_reiserfs LV` »

```
# resize_reiserfs /dev/JARDIN/ananas
XXXXXXXXXX
```

**A completer...**

[1] Faut-il encore utiliser le filesyetme ReiserFS ? : fin 2006, Hans Reiser, le développeur américain de ce filesystem, est soupçonné par la justice américaine d'avoir fait disparaitre son épouse Nina dont il était en train de divorcer.





La commande « `fsck` » peut se révéler non pratique :

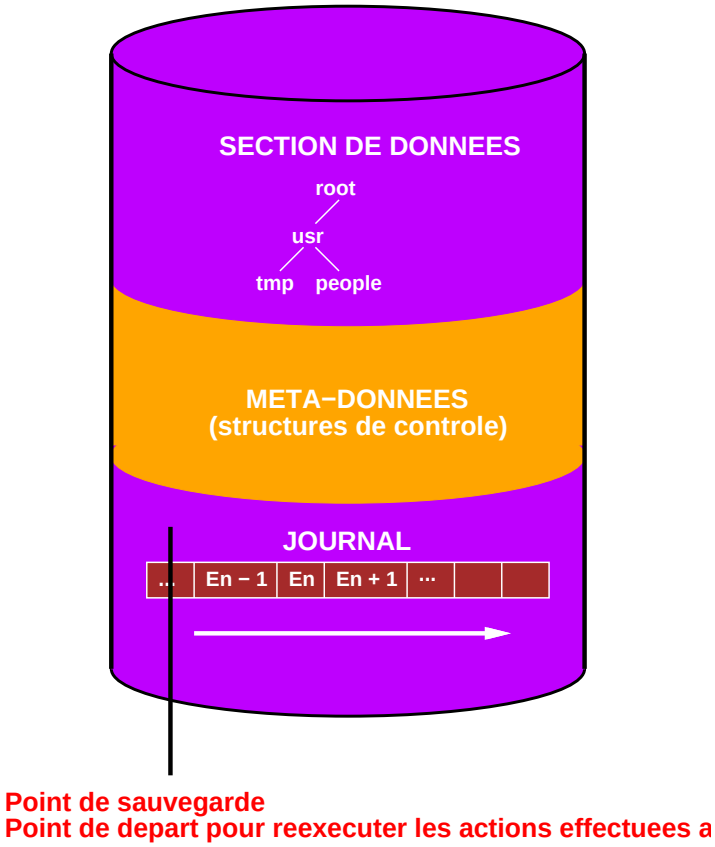
- lorsque l'on a beaucoup de disques
- lorsque les disques sont gros

⇒ solution du filesystem journalisé :

- garantit la fiabilité des données
- grande rapidité de recouvrement lors d'un crash machine grâce à la lecture complète des opérations stockées dans un journal

Le journal correspond à une zone disque réservée dans laquelle sont rangées séquentiellement toutes les mises à jour de données, et ce, avant leur mise à jour effective dans le système de fichiers.

Lors d'un crash machine, la relecture du journal permet de reconstituer la totalité des systèmes de fichiers en quelques secondes. Il n'y a plus besoin de recourir à `fsck`.








Filesystems journalisés disponibles sur :

- DIGITAL UNIX : filesystem de nom ADVFS
- LINUX : filesystems de nom EXT3 et REISERFS. Plusieurs filesystems journalisés sont disponibles.
- SOLARIS (depuis SOLARIS version 8) : option « `logging` » à indiquer au niveau de « `/etc/vfstab` ». Très simple à activer comme à désactiver.

Un CDR ou CDRW peut être mono-session ou multi-session.
Il ne sera envisagé ici que le cas de gravure mono-session.

Un graveur CDR dit `1X` grave la capacité de 650 Mo en 1 heure.
Appliquer une règle de trois pour un lecteur `N X`.

La gravure de CD/DVD se fait en deux étapes sous UNIX :

1. construction d'une image de l'arborescence à graver ; le format est dit ISO 9660 avec les extensions Rock Ridge pour gérer les noms longs des fichiers et des répertoires ainsi que des choses spéciales comme les liens symboliques
2. gravure de l'image iso

La commande la plus connue pour faire cela est « `mkisofs` », disponible à l'URL `ftp://ftp.berlios.de/pub/cdrecord/` dans le package « `cdrtools` ».





◇ Exemple 1

On veut une image ISO utilisable sur plateforme UNIX et PC Windows.
⇒ employer les options « `-r` » et «`-J` » :

```
% mkisofs -r -J -o image.iso arborescence
Using UNIX_000.EPS;1 for  /unix-fork.pdf (unix-fork-exec.pdf)
Using VANGO000.;1 for  arborescence/poubelle/vangogh-ps-ef (vangogh-ps-e)
Using TASKM000.EPS;1 for  arborescence/windows/taskmgr-07.pdf (taskmgr-06.pdf)
Using TASKM001.EPS;1 for  arborescence/windows/taskmgr-06.pdf (taskmgr-05.pdf)
Using TASKM002.EPS;1 for  arborescence/windows/taskmgr-05.pdf (taskmgr-03.pdf)
Using TASKM003.EPS;1 for  arborescence/windows/taskmgr-03.pdf (taskmgr-02.pdf)
Using TASKM004.EPS;1 for  arborescence/windows/taskmgr-02.pdf (taskmgr-01.pdf)
Using TASKM000.GIF;1 for  arborescence/windows/taskmgr-07.gif (taskmgr-06.gif)
Using TASKM001.GIF;1 for  arborescence/windows/taskmgr-06.gif (taskmgr-05.gif)
Using TASKM002.GIF;1 for  arborescence/windows/taskmgr-05.gif (taskmgr-03.gif)
Using TASKM003.GIF;1 for  arborescence/windows/taskmgr-03.gif (taskmgr-02.gif)
Using TASKM004.GIF;1 for  arborescence/windows/taskmgr-02.gif (taskmgr-01.gif)
Using CHROO000.DIA;1 for  arborescence/chroot/chroot.dia~ (chroot.dia)
 61.28% done, estimate finish Sat Oct 23 13:03:57 2004
Total translation table size: 0
Total rockridge attributes bytes: 4284
Total directory bytes: 10240
Path table size(bytes): 64
Max brk space used 10000
8184 extents written (15 Mb)
```

◇ Exemple 2

On veut une image ISO à partir d'un CDROM que l'on a (CDROM de fichiers informatiques, pas CD musical).
⇒ employer la commande « `dd` » :

```
% dd if=/dev/cdrom of=/tmp/image.iso bs=128k
```





◇ Exemple 3

**ATTENTION : pour des backups système, ne pas utiliser l'option `-r` mais lui préférer l'option `-R`. Sinon les droits enregistrés seront incorrects et ne refléteront pas les vrais droits.**

Soit des fichiers protégés :

```
% ls -l
drwxr-xr-x   2 besancon adm          512 Sep 27 19:18 backup/
drwx------   6 pgsql    pgsql        512 Oct 23 11:20 data/
```

Si l'on fait :

```
# mkisofs -r -J -o /tmp/image.iso .
```

on obtient après gravure un CDROM où les droits et les propriétaires des fichiers sont :

```
% ls -l /mnt/cdrom
dr-xr-xr-x   2 root     root         2048 Sep 27 19:18 backup
dr-xr-xr-x   6 root     root         2048 Oct 23 11:20 data
```

Tous les fichiers dans `data` sont lisibles ! ! !
Tous les fichiers sont propriété de `root` ! ! !
⇒ Ce n'est pas une sauvegarde « système » réutilisable.





Si l'on fait :

```
# mkisofs -R -J -o /tmp/image.iso .
```

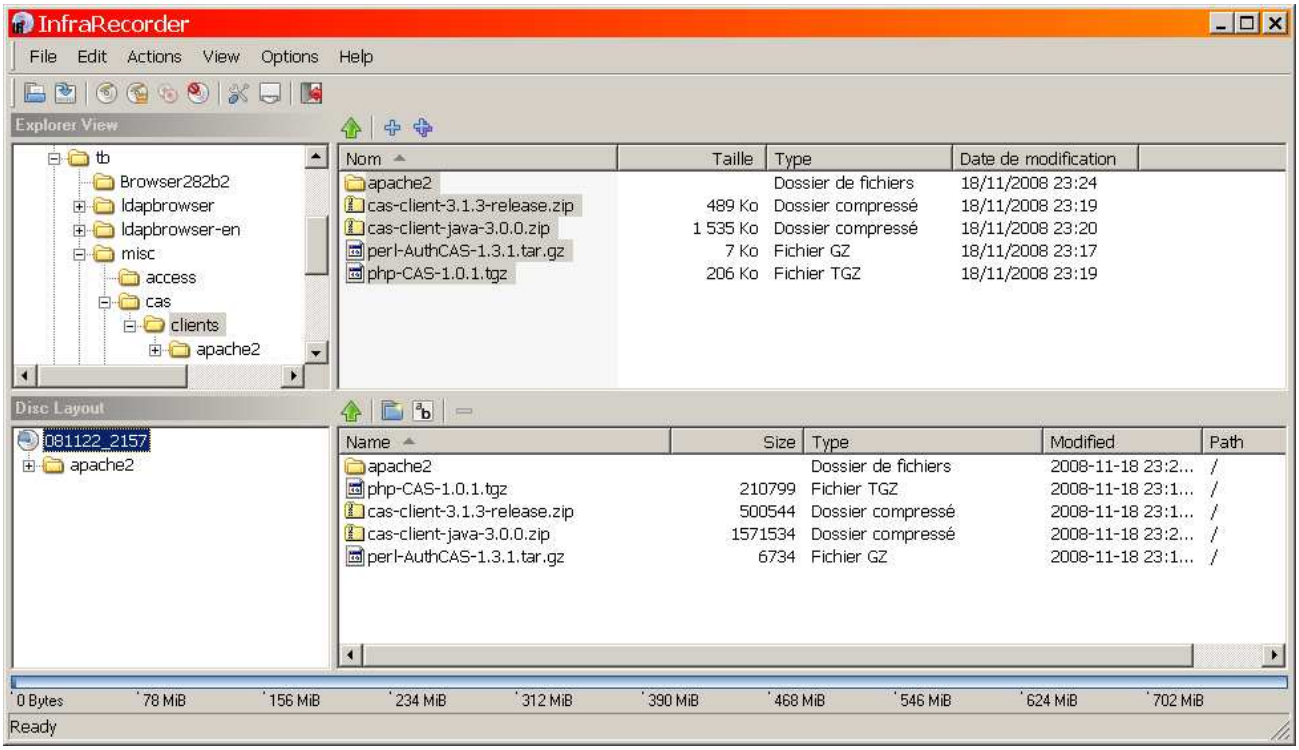
on obtient après gravure un CDROM où les droits et les propriétaires des fichiers sont :

```
% ls -l /mnt/cdrom
drwxr-xr-x   2 besancon adm          2048 Sep 27 19:18 backup
drwx------   6 pgsql    pgsql        2048 Oct 23 11:20 data
```

Tous les fichiers ont conservé leurs droits originaux. La sauvegarde « système » est utilisable.

## ◇ Création d'une image ISO 9660 sous WINDOWS (1)

Façon la plus simple : INFRARECORDER disponible sur `http://infrarecorder.org/`

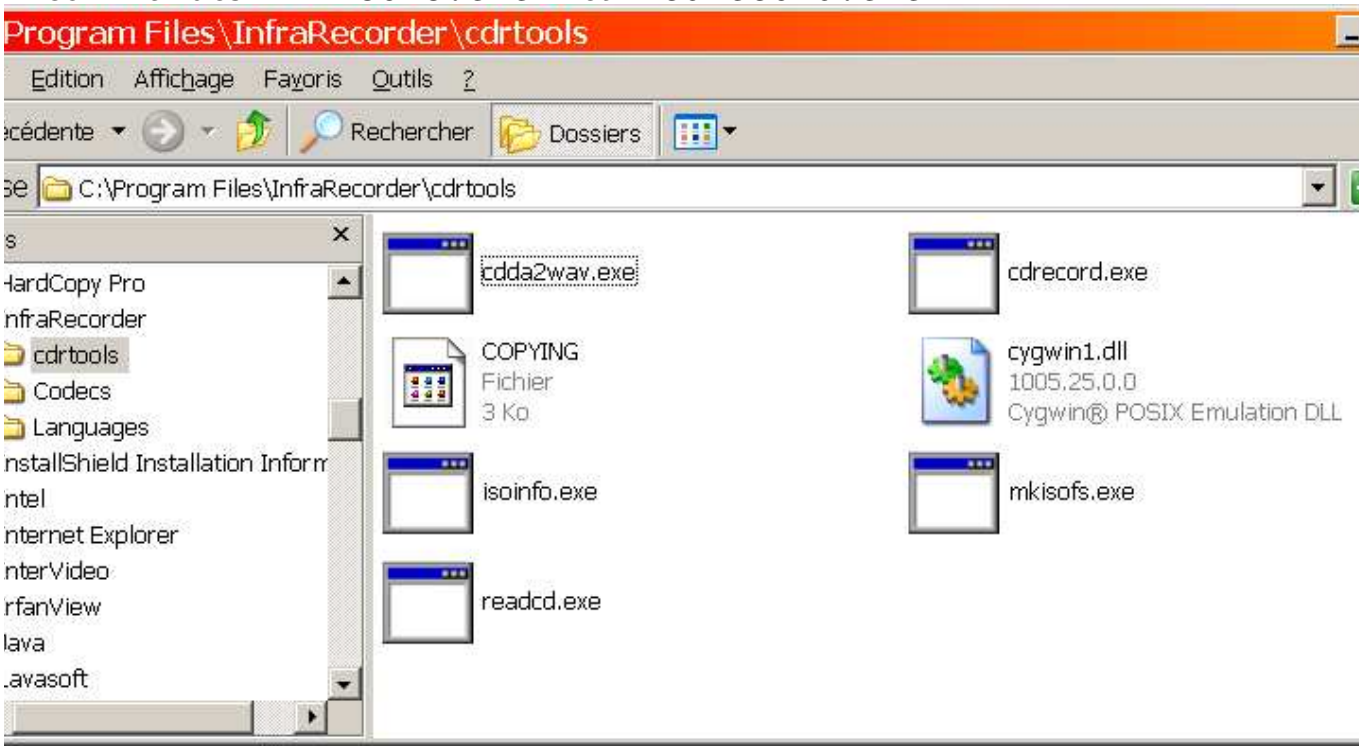







## ◇ Création d'une image ISO 9660 sous WINDOWS (2)

Commande « `mkisofs.exe` » portée d'UNIX à WINDOWS.
En fait, disponible dans INFRARECORDER = interface graphique aux commandes « `mkisofs.exe` » et « `cdrecord.exe` ».



On peut vouloir vérifier le contenu d'une image ISO avant de la graver. Certains systèmes UNIX permettent de « monter une image ISO 9660 ».

Avantages :

- Permet de tester une image avant de la graver peut-être pour rien
- Gain de temps si l'on peut monter l'image ISO car on évite la phase de gravure sur le support CDROM, support altérable (rayures, cassabilité, etc.)
- L'image ISO n'est pas rayable, cassable

Défauts :

- L'image ISO peut-être corrompue lors d'un transfert réseau par exemple. Nécessité de faire un checksum de l'image et de la vérifier.





## ◇ Montage d'une image ISO 9660 sous LINUX

```
# mount -o loop -t iso9660 /var/tmp/image.iso /mnt/iso
# ls /mnt/iso
20041013                factures                pgsql
__THIS_IS_SAE5_DATA__   html                    rr_moved
cisco                   pamo                    shares
# umount /mnt/iso
```

### ◇ Montage d'une image ISO 9660 sous SOLARIS

Sur SOLARIS, il faut :

1. associer l'image ISO avec un device de type block ⇒ commande « `lofiadm` »
2. monter ensuite le device de type block ⇒ commande « `mount` » traditionnelle
3. démonter le device de type block apès utilisation ⇒ commande « `umount` » traditionnelle
4. dissocier le device de type block ⇒ commande « `lofiadm` »

Exemple d'une session complète page suivante.





Exemple d'une session complète :

```
# lofiadm -a /var/tmp/image.iso
/dev/lofi/1
# mount -r -F hsfs /dev/lofi/1 /mnt/iso
# ls /mnt/iso
20041013                __THIS_IS_SAE5_DATA__   cisco
factures                html                    pamo
pgsql                   rr_moved                shares
# umount /mnt/iso
# lofiadm -d /dev/lofi/1
```

## ◇ Montage d'une image ISO 9660 sous FREEBSD

Sur FREEBSD, il faut :

1. associer l'image ISO avec un device de type block ⇒ commande « `mdconfig` » (anciennement « `vnconfig` »)
2. monter ensuite le device de type block ⇒ commande « `mount` » traditionnelle
3. démonter le device de type block apès utilisation ⇒ commande « `umount` » traditionnelle
4. dissocier le device de type block ⇒ commande « `mdconfig` » (anciennement « `vnconfig` »)

Exemple d'une session complète page suivante.





Exemple d'une session complète :

```
# mdconfig -a -t vnode -f /var/tmp/image.iso
md0
# mount -t cd9660 /dev/md0 /mnt/iso
# df -k /mnt/iso
Filesystem 1K-blocks   Used Avail Capacity  Mounted on
/dev/md0      354240 354240     0   100%    /mnt/iso
# ls /mnt/iso
20041013               factures              pgsql
__THIS_IS_SAE5_DATA__  html                  rr_moved
cisco                  pamo                  shares
# umount /mnt/iso
# mdconfig -d -u md0
```

◇ Montage d'une image ISO 9660 sous WINDOWS

On peut monter une image ISO 9660 via le logiciel gratuit DAEMONTOOLS disponible à l'URL `http://www.daemon-tools.cc`.

AUTRES COMMANDES

**A completer...**





La gravure d'une image ISO 9660 est une opération spécifique à chaque OS.

◇ Gravure d'une image ISO sous LINUX

La gravure d'une image ISO sous LINUX se fait par la commande suivante :

- pour graveur IDE : commande non standard : « `cdrecord` » disponible à l'URL
  `http ://www.fokus.fhg.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html`.
- logiciel avec interface graphique : K3B, site officiel
  `http://www.k3b.org`





Liste des graveurs IDE par CDRECORD : « `cdrecord -scanbus` »

Exemple :

```
% cdrecord -scanbus
...
scsibus1:
        1,0,0   100) 'TSSTcorp' 'CDW/DVD SH-M522C' 'TS01' Removable CD-RC
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) *
        1,6,0   106) *
        1,7,0   107) *
```

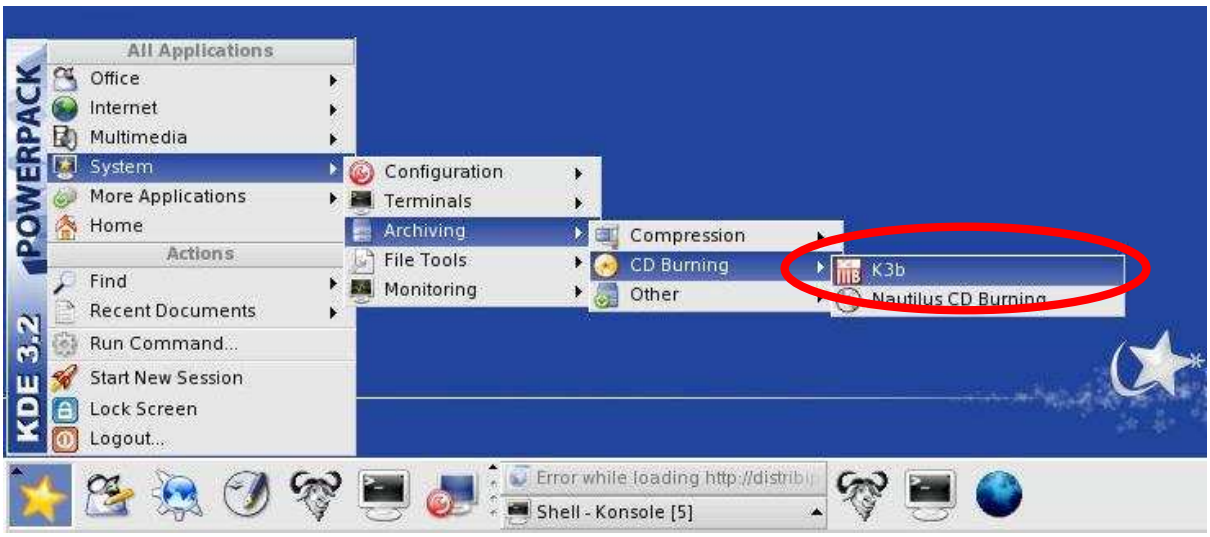
## Exemple de gravure d'une image ISO sur un graveur IDE par CDRECORD :

```
% cdrecord -v dev=ATA:1,0,0 xendemo-2.0.6.iso
...
scsidev: 'ATA:1,0,0'
devname: 'ATA'
scsibus: 1 target: 0 lun: 0
...
Track 01: data   697 MB
...
Track 01: Total bytes read/written: 731414528/731414528 (357136 sectors).
Writing  time:  152.504s
Average write speed  33.2x.
Min drive buffer fill was 13%
Fixating...
Fixating time:   23.329s
cdrecord: fifo had 11521 puts and 11521 gets.
cdrecord: fifo was 50 times empty and 2066 times full, min fill was 0%.
```





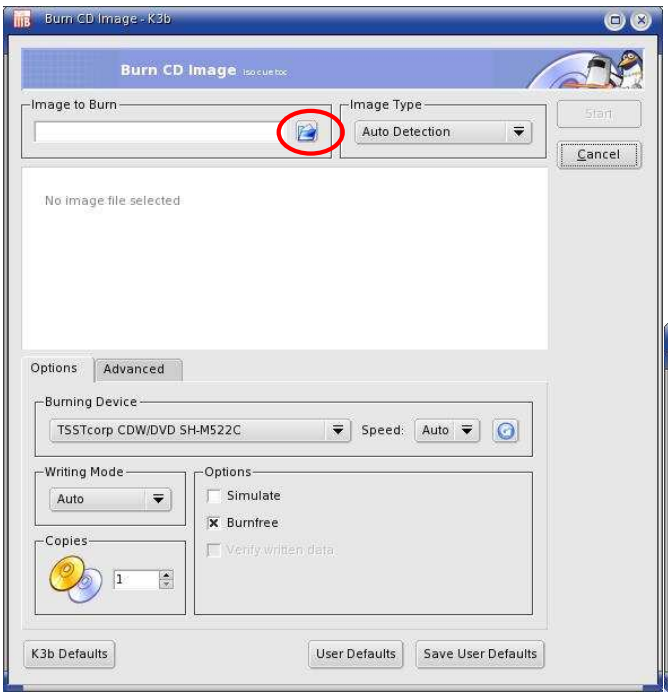
## Exemple de gravure d'une image ISO sur un graveur IDE par K3B :

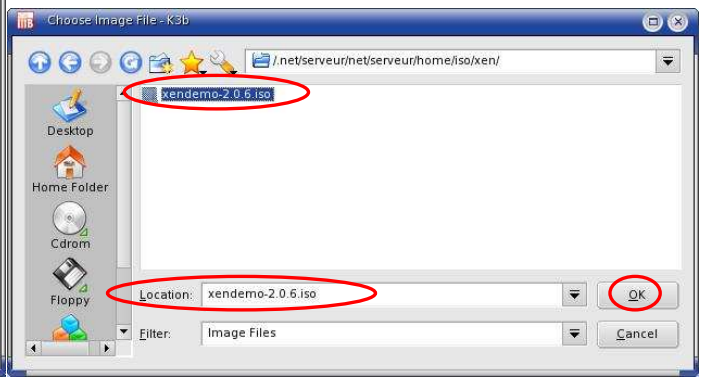


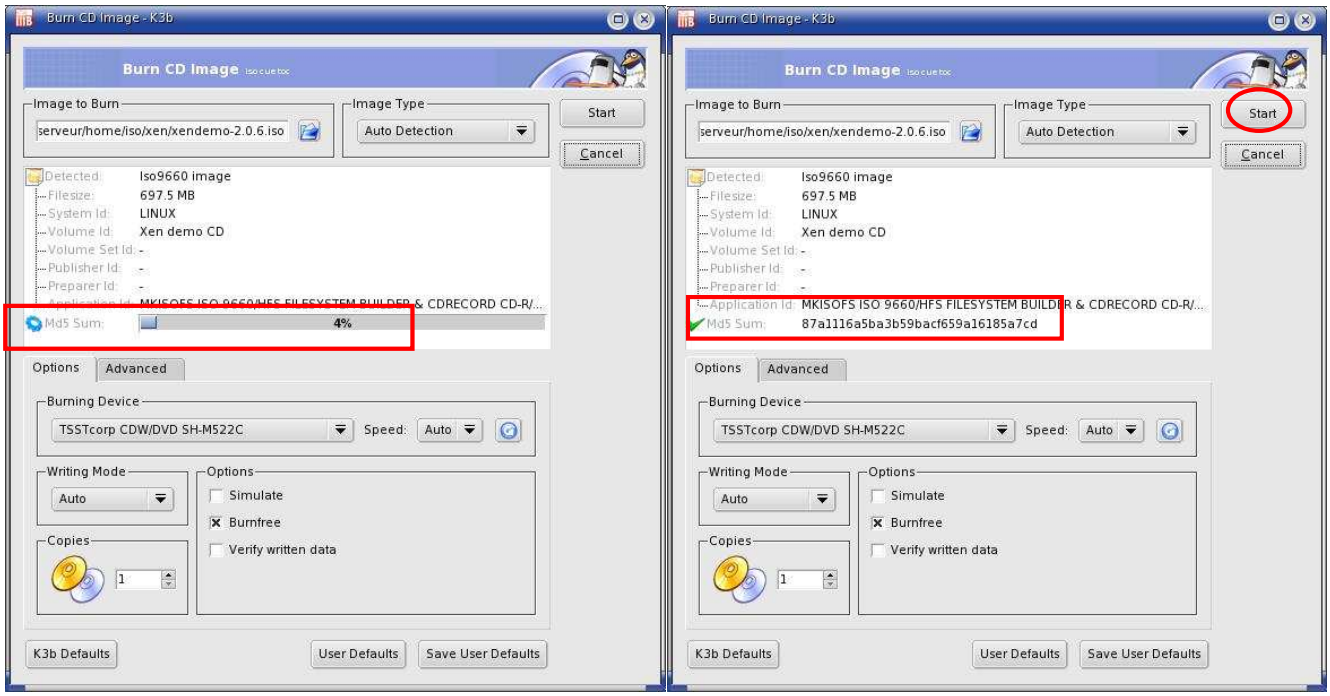

K3b - The CD Kreator
File Project Plugins Tools Settings Help
TSSTcorp - CDW/DV...
Home
Root
xendemo-2.
0.6.iso
CD
DVD
Encode Video...
Copy CD...
Burn CD Image...
New Audio CD Project
New Data CD Project
New Data DVD Project
Copy CD...
Temp: 18.3 GB/23.2 GB
K3b 0.11.16





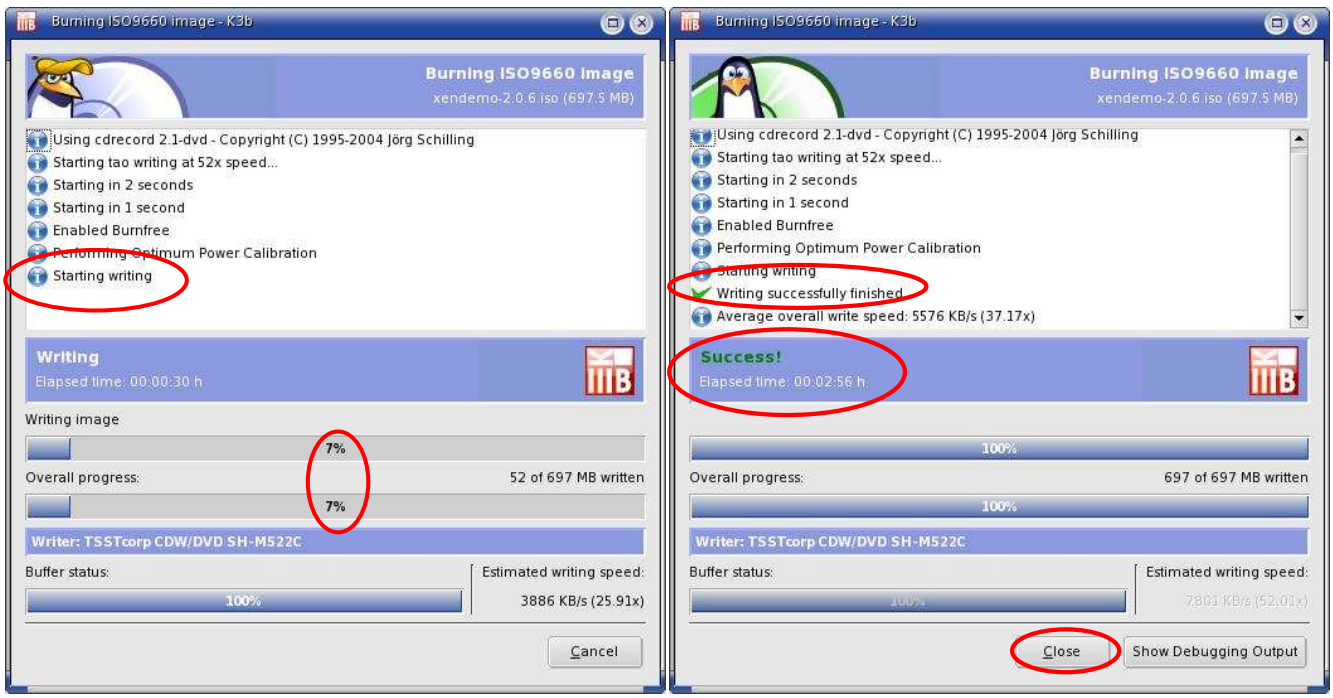

Burn CD Image - K3b
Burn CD Image
Image to Burn
Image Type
Auto Detection
Start
Cancel
No image file selected
Options
Advanced
Burning Device
TSSTcorp CDW/DVD SH-M522C
Speed:
Auto
Writing Mode
Auto
Options
Simulate
Burnfree
Copies
K3b Defaults
User Defaults
Save User Defaults


Choose Image File - K3b
/net/serveur/net/serveur/home/iso/xen/
xendemo-2.0.6.iso
Desktop
Home Folder
Cdrom
Floppy
Location:
xendemo-2.0.6.iso
OK
Filter:
Image Files
Cancel


Burn CD Image - K3b
Burn CD Image
Image to Burn
serveur/home/iso/xen/xendemo-2.0.6.iso
Image Type
Auto Detection
Start
Cancel
Detected: Iso9660 image
Filesize: 697.5 MB
System Id: LINUX
Volume Id: Xen demo CD
Volume Set Id: -
Publisher Id: -
Preparer Id: -
Application Id: MKISOFS ISO 9660/HFS FILESYSTEM BUILDER & CDRECORD CD-R/...
Md5 Sum: 4%
Options
Advanced
Burning Device
TSSTcorp CDW/DVD SH-M522C
Speed: Auto
Writing Mode
Auto
Options
Simulate
Burnfree
Verify written data
Copies
1
K3b Defaults
User Defaults
Save User Defaults
Burn CD Image - K3b
Burn CD Image
Image to Burn
serveur/home/iso/xen/xendemo-2.0.6.iso
Image Type
Auto Detection
Start
Cancel
Detected: Iso9660 image
Filesize: 697.5 MB
System Id: LINUX
Volume Id: Xen demo CD
Volume Set Id: -
Publisher Id: -
Preparer Id: -
Application Id: MKISOFS ISO 9660/HFS FILESYSTEM BUILDER & CDRECORD CD-R/...
Md5 Sum: 87a1116a5ba3b59bacf659a16185a7cd
Options
Advanced
Burning Device
TSSTcorp CDW/DVD SH-M522C
Speed: Auto
Writing Mode
Auto
Options
Simulate
Burnfree
Verify written data
Copies
1
K3b Defaults
User Defaults
Save User Defaults






Burning ISO9660 image - K3b
Burning ISO9660 image
xendemo-2.0.6.iso (697.5 MB)
Using cdrecord 2.1-dvd - Copyright (C) 1995-2004 Jörg Schilling
Starting tao writing at 52x speed...
Starting in 2 seconds
Starting in 1 second
Enabled Burnfree
Performing Optimum Power Calibration
Starting writing
Writing
Elapsed time: 00:00:30 h
Writing image
7%
Overall progress:
52 of 697 MB written
7%
Writer: TSSTcorp CDW/DVD SH-M522C
Buffer status:
100%
Estimated writing speed:
3886 KB/s (25.91x)
Cancel
Burning ISO9660 image - K3b
Burning ISO9660 image
xendemo-2.0.6.iso (697.5 MB)
Using cdrecord 2.1-dvd - Copyright (C) 1995-2004 Jörg Schilling
Starting tao writing at 52x speed...
Starting in 2 seconds
Starting in 1 second
Enabled Burnfree
Performing Optimum Power Calibration
Starting writing
Writing successfully finished
Average overall write speed: 5576 KB/s (37.17x)
Success!
Elapsed time: 00:02:56 h
100%
Overall progress:
697 of 697 MB written
100%
Writer: TSSTcorp CDW/DVD SH-M522C
Buffer status:
Estimated writing speed:
Close
Show Debugging Output

◇ Gravure d'une image ISO sous SOLARIS

La gravure d'une image ISO sous LINUX se fait par les commandes suivantes :

- pour graveur CDRW IDE : commande fournie par SOLARIS : « `cdrw` »
- pour graveur DVD+-RW IDE : commande fournie par SOLARIS : « `cdrw` » (selon niveau de patch)





Liste des graveurs IDE et SCSI par CDRW : « `cdrw -l` »

Exemple :

```
# cdrw -l
Looking for CD devices...
    Node                    Connected Device                Device type
----------------------+--------------------------------+-----------------
 /dev/rdsk/c0t1d0s2   | _NEC     DVD_RW ND-2500A  1.06 | CD Reader/Writer
 /dev/rdsk/c1t5d0s2   | HP       CD-Writer+ 9200  1.0e | CD Reader/Writer
```

Exemple de gravure d'une image ISO sur un graveur IDE par CDRW :

```
# cdrw -i -v -d /dev/rdsk/c0t1d0s2 image.iso
Initializing device...done.
Writing track 1...12 %
Writing track 1...done.
Finalizing (Can take several minutes)...done.
```





Effacement d'un CDRW sur un graveur IDE par CDRW :
« `cdrw -b all` »

Exemple :

```
# cdrw -b -v -d /dev/rdsk/c0t1d0s2 session
Initializing device...done.
Blanking the media (Can take several minutes)...done.
```

```
# cdrw -b -v -d /dev/rdsk/c0t1d0s2 all
Blanking the media (Can take several minutes)...done.
```

Exemple de gravure d'une image ISO sur un graveur IDE par CDRW :

```
# cdrw -i -v -d /dev/rdsk/c0t1d0s2 image.iso
Initializing device...done.
Writing track 1...12 %
Writing track 1...done.
Finalizing (Can take several minutes)...done.
```





◇ Gravure d'une image ISO sous FREEBSD

La gravure d'une image ISO sous LINUX se fait par la commande suivante :

- pour graveur CDRW IDE : commande fournie par FREEBSD : « `burncd` »
- pour graveur DVD+-RW IDE : commande fournie par FREEBSD : « `burncd` »

## Effacement d'un CDRW : « burncd blank »

Exemple :

```
# burncd blank
blanking CD - 100 % done
```





## Gravure d'une image ISO 9660 en mono-session : « burncd data image.iso fixate »

Exemple :

```
% burncd data image.iso fixate
next writeable LBA 0
writing from file image.iso size 354240 KB
written this track 354240 KB (100%) total 354240 KB
fixating CD, please wait..
```

◇ Gravure d'une image ISO sous WINDOWS

La gravure d'une image ISO sous WINDOWS se fait par les commandes suivantes :

- pour graveur CDRW IDE : commande fournie par MICROSOFT dans le resource kit Windows 2003 : « `cdburn.exe` »
- pour graveur DVD+-RW IDE : commande fournie par MICROSOFT dans le resource kit Windows 2003 : « `dvdburn.exe` »
- pour graveur CDRW / DVD+-RW IDE : commande non standard : « `INFRARECORDER` » disponible sur `http://infrarecorder.org/`
- pour graveur CDRW / DVD+-RW IDE : commande non standard : « `cdrecord.exe` » disponible via INFRARECORDER `http://infrarecorder.org/`

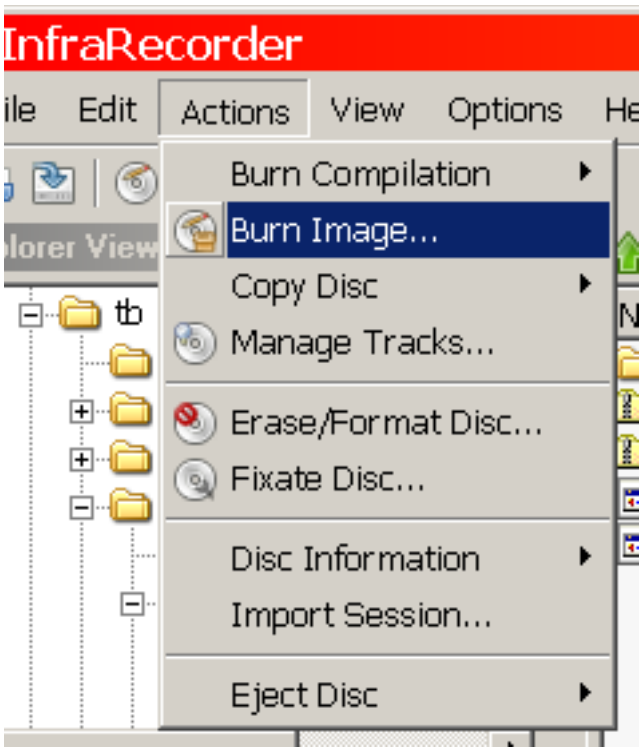


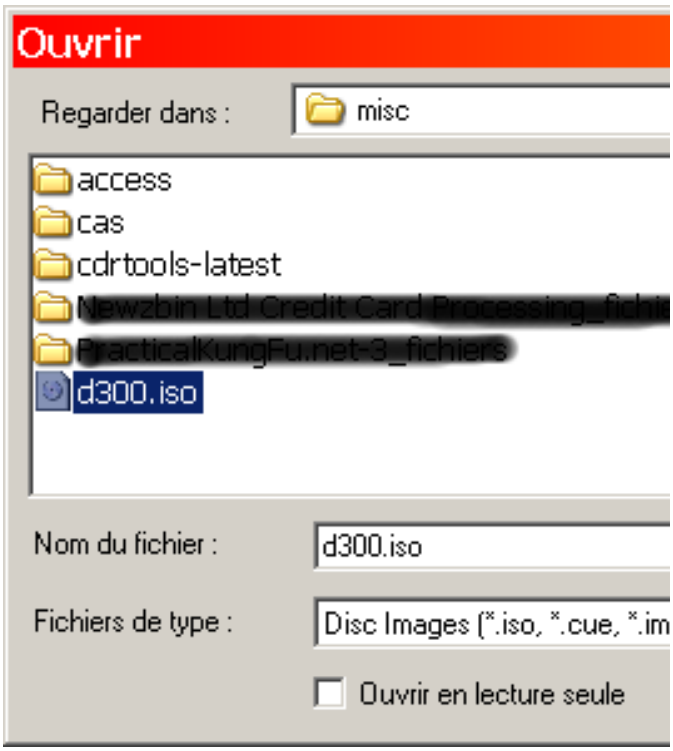


Gravure d'une image ISO 9660 en mono-session :
« `cdburn.exe drive -erase [image [options]]` » ou
« `cdburn.exe drive image [-speed max]` »
Exemple :

```
C:\Documents and Settings\root\Desktop>cdburn e: -erase
Erasing media before burning
Number of blocks in ISO image is ffffffff
Erasing target media
Media erased

C:\Documents and Settings\root\Desktop>cdburn e: cd040818.iso -spe
Requesting burn at maximum speed
Number of blocks in ISO image is 613
Finished Writing
Synchronizing Cache: Error ejecting/reinserting disc
```

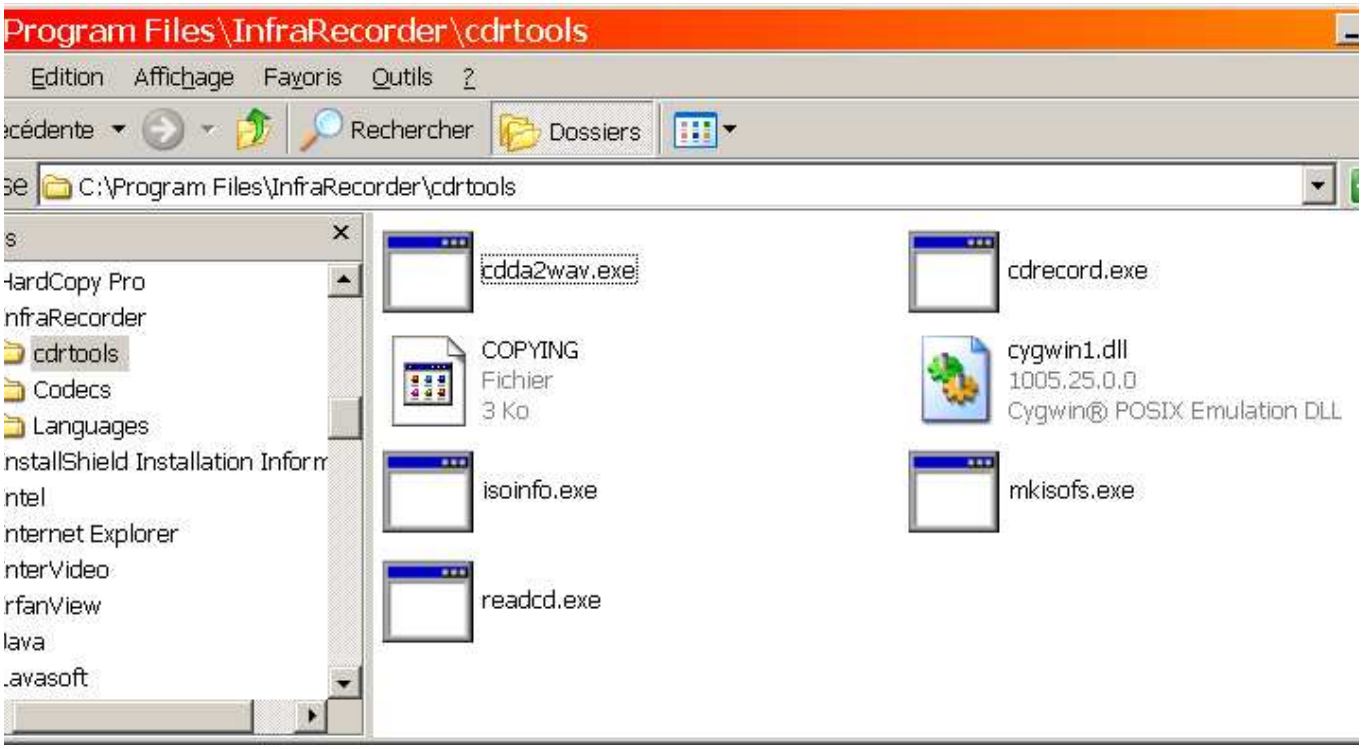







Commande « `cdrecord.exe` » portée d'UNIX à WINDOWS.
En fait, disponible dans INFRARECORDER = interface graphique aux commandes « `mkisofs.exe` » et « `cdrecord.exe` ».



Utilité :

- 5 formations
- 2 projets par formation
- 20 élèves par formation
- travail en binôme

Au total : 5 * 2 * 20 / 2 = 100 groupes UNIX à gérer que seul l'administrateur peut gérer et que le modèle des droits UNIX rendra très difficile à mettre en place.

⇒ création d'un autre mécanisme connu sous le nom de *Access Control Lists* ou ACL.
Disponible sur LINUX, SOLARIS, FREEBSD, WINDOWS

Attention aux différentes implémentations non compatibles des ACL. L'unification des différentes implémentations n'a pas encore eu lieu en pratique.





Le mécanisme des ACL sous LINUX nécessite :

- un noyau 2.4 plus des patches
- un noyau 2.6 standard
- un filesystem supportant les ACL (ext2, ext3, jfs, xfs)

(package `acl-2.2.31-1mdk.i586.rpm` sur le CDROM 6 de Mandriva 2006)

◇ Activation du mécanisme des ACL

Il faut indiquer l'option « `acl` » au niveau du fichier « `/etc/fstab` » ou de la commande « `mount` » :

Exemple avec création d'un filesystem pour l'occasion :

```
# dd if=/dev/zero of=/tmp/exemple.raw bs=1M count=128
# mkfs /tmp/exemple.raw
...
# mount -o loop,acl -t ext2 /tmp/exemple.raw /mnt/exemple
# mount
/dev/sda5 on / type ext3 (rw)
...
/exemple.raw on /mnt/exemple type ext2 (rw,loop=/dev/loop0,acl)
```



## Chapitre 10 • Systèmes de fichiers
### §10.46 • ACL sous LINUX : lister les ACL / `getfacl`

(en anglais *get file ACL*)

Syntaxe : `getfacl [options] objets`

## ◇ Exemple : fichier sans ACL

Rien de visible au niveau de « ls » :

```
% /bin/ls -l
-rw-r--r--   1 besancon adm           49 Oct 31 14:43 ananas.txt
```

Au niveau de « getfacl » :

```
% getfacl ananas.txt
# file: ananas.txt
# owner: besancon
# group: adm
user::rw-
group::r--
other:r--
```





## ◇ Exemple : fichier avec ACL

Au niveau de ls : le symbole « + » indique la présence d'ACL :

```
% ls -l
-rw-r--r--   1 besancon adm           49 Oct 31 14:43 ananas.txt
-rw-r--r--+  1 besancon adm           49 Oct 31 14:43 cerise.txt
```

Au niveau de getfacl :

```
% getfacl cerise.txt
# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-
group::r--
mask:rw-
other:r--
```

◇ Démonstration de l'effet de l'ACL

```
% id
uid=6000(mysql) gid=6000(mysql)
% cat /etc/hosts > cerise.txt
% ls -l
-rw-r--r--+  1 besancon adm         2408 Oct 31 15:17 cerise.txt
```





ATTENTION : il y a la notion de *mask* qui se greffe par dessus les droits individuels pour donner les droits effectifs :

```
% getfacl cerise.txt
# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-
group::r--
mask:rw-
other:r--
```

Ici les droits effectifs ne sont pas explicités car ils sont égaux à ceux de l'utilisateur « `mysql` ».

(en anglais *set file ACL*)

Syntaxe : `setfacl [options] objets`





◇ Exemple

```
% setfacl -m user:mysql:rw- cerise.txt
% getfacl cerise.txt
# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-
group::r--
mask:rw-
other:r--
```

◇ Exemple de modification du mask

Interdiction du droit en écriture donné individuellement via les ACL :

```
% setfacl -m mask:r-- cerise.txt
% getfacl exemple.txt
# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-          #effective:r--
group::r--
mask:r--
other:r--
```

Le droit effectif est bien « `rw- & r-- = r--` »





◇ A propos des commandes de manipulation de fichiers :

**Il faut vérifier que les commandes de manipulation des objets fonctionnent correctement avec les ACL.**

Par exemple, la commande « `cp` » ne conserve pas les ACL sous LINUX Mandriva 2006.

◇ A propos de tar

A priori le « tar » de LINUX ne supporte pas les ACL.

2 alternatives :

- lors d'un backup faire :
  - « getfacl -R objets > /tmp/acl.txt »
  - « tar cvf archive.tar objets »

  et lors de la restoration faire :
  - « tar xvf archive.tar »
  - « setfacl --restore=/tmp/acl.txt »
- utiliser la commande star disponible à l'adresse http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/star.html





Le mécanisme des ACL sous SOLARIS est toujours activé contrairement à LINUX Mandriva 2006 par exemple.

(en anglais *get file ACL*)

Syntaxe : `getfacl [options] objets`





◇ Exemple : fichier sans ACL

Rien de visible au niveau de « `ls` » :

```
% /bin/ls -l
total 2
-rw-r--r--   1 besancon adm           49 Oct 31 14:43 ananas.txt
```

Au niveau de « `getfacl` » :

```
% getfacl ananas.txt

# file: ananas.txt
# owner: besancon
# group: adm
user::rw-
group::r--              #effective:r--
mask:r--
other:r--
```

ATTENTION : il y a la notion de **mask** qui se greffe par dessus les droits individuels pour donner les droits effectifs. Ici les droits effectifs sont explicités car ils ce sont pas égaux à ceux de l'utilisateur « mysql ».

```
% getfacl ananas.txt

# file: ananas.txt
# owner: besancon
# group: adm
user::rw-
group::r--              #effective:r--
mask:r--
other:r--
```

**ATTENTION : différence par rapport à LINUX Mandriva 2006 : le mask par défaut est « r-- » et non pas « rw- ».**





◇ Exemple : fichier avec ACL

Au niveau de « ls » : le symbole « + » indique la présence d'ACL :

```
% ls -l
total 4
-rw-r--r--   1 besancon adm          49 Oct 31 14:43 ananas.txt
-rw-r--r--+  1 besancon adm          49 Oct 31 14:43 cerise.txt
```

Au niveau de « getfacl » :

```
% getfacl cerise.txt

# file: exemple.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-          #effective:rw-
group::r--              #effective:r--
mask:rw-
other:r--
```

◇ Démonstration de l'effet de l'ACL :

```
% id
uid=6000(mysql) gid=6000(mysql)
% cat /etc/hosts > cerise.txt
% ls -l
-rw-r--r--+  1 besancon adm         2408 Oct 31 15:17 cerise.txt
```





(en anglais *set file ACL*)

Syntaxe : `setfacl [options] objets`

◇ Exemple

```
% setfacl -m user:mysql:rw- cerise.txt
% getfacl cerise.txt

# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-          #effective:r--
group::r--               #effective:r--
mask:r--
other:r--
```

A cause du mask par défaut, l'utilisateur « mysql » n'aura pas son droit en écriture même.
Cf l'indication « #effective:r-- ».





◇ Exemple de modification du mask

Autoriser les droits donnés individuellement via les ACL :

```
% setfacl -m mask:rw- cerise.txt
% setfacl -m user:mysql:rw- cerise.txt
% getfacl cerise.txt

# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-          #effective:rw-
group::r--               #effective:r--
mask:rw-
other:r--
```

Sur SOLARIS, les commandes fournies par le système intégrent le support des ACL.

Exemple : commande « cp » compatible avec les ACL

```
% /bin/ls -l
-rw-r--r--+  1 besancon adm           49 Oct 31 14:43 cerise.txt
% cp cerise.txt fraise.txt
% ls -l
-rw-r--r--+  1 besancon adm           49 Oct 31 14:43 cerise.txt
-rw-r--r--+  1 besancon adm           49 Oct 31 15:13 fraise.txt
```





◇ A propos de tar

La commande « tar » de SOLARIS supporte les ACL via l'option « -p ».

Sauvegarde des fichiers avec leurs ACL dans une archive TAR :

```
% tar cvfp /tmp/archive.tar cerise.txt
a cerise.txt 1K
```

On retrouve bien dans l'archive TAR la présence d'ACL :

```
% strings /tmp/archive.tar | grep mysql
user::rw-,user:mysql:rw-,group::r--,mask:rw-,other:r--
```

Restoration des fichiers avec leurs ACL à partir de l'archive TAR :

```
% cd /chemin/vers/ailleurs
% tar xvpf /tmp/archive.tar
tar: blocksize = 6
x cerise.txt, 49 bytes, 1 tape blocks

% ls -l
-rw-r--r--+  1 besancon adm           49 Oct 31 14:43 cerise.txt

% getfacl cerise.txt

# file: cerise.txt
# owner: besancon
# group: adm
user::rw-
user:mysql:rw-          #effective:rw-
group::r--               #effective:r--
mask:rw-
other:r--
```





**A completer...**

Ci joint dans la version imprimée de ce cours, un document sur la technologie SMART.

Adresse web du document fourni :
« http://www.???.com/???/Monitoring_Hard_Disks_with_smartmont



## Monitoring Hard Disks with smartmontools

# Crash Prevention

Hard disks don't always die as suddenly as you might think. The right tools can help you detect hard disk issues before they become critical.

BY GABRIELE POHL

Hard disks have rotating moving parts – disks that rotate at 5400 or 7200 or more revolutions per minute – and the heads are subject to extreme acceleration and deceleration. Because they have moving parts, hard disks are subject to wear and tear. Manufacturers typically estimate a mean failure time for their products – a purely statistical value, and they provide no guarantee that the disk will not die on you within the first month. And Murphy's law says that your hard disk is more likely to die when you are not expecting it to do so, that is, you will probably lose it when you do not have a recent backup or when spare parts are hard to come by.

If you are lucky, your computer might output a message like the following when you boot it:

```
SMART Failure Predicted on
 Primary Master: Maxtor 34098H4
Warning! Immediately back-up
your data and replace your hard
 disk drive. A Failure may
 be imminent.
```

The message is clear. Your hard disk looks likely to fail, and you should back up your data as soon as possible. This is a friendly warning from your computer BIOS's SMART error detection feature.

## SMART Helper

SMART is the acronym for **S**elf-**M**onitoring, **A**nalysis and **R**eporting **T**echnology, a system that all modern ATA and SCSI hard disks, as well as SCSI tape drives, should have. Besides logging measured values and errors, SMART has device testing features. Of course it is a good think to know about an impending disaster well in advance. And this is exactly what the Smartmontools package [1] does for you. It accesses the SMART feature provided by your hard disks and runs a daemon called *smartd* to provide automated controls.

The package is available for the current versions of the Linux, FreeBSD, NetBSD, Solaris, Darwin, and even Microsoft Windows. You can download the Linux version as a binary RPM or source code version from [1], and an installation guide is available from the same address. Smartmontools supports versions 3 through 7 of the ATA/ATAPI standard. Its predecessor was a software package called Smartsuite. Development of Smartsuite was discontinued back in September 2001.

Listing 1 shows you an interactive approach to displaying basic information for a device using *smartctl*. The failed hard disk in our example is attached to the IDE bus as the primary master; it is (or should be) accessible as */dev/hda*. These commands all require *root* privileges, since non-privileged users do not have access to device files.

Listing 1 tells you that the drive is a 34097H4 model by Maxtor; its serial number is L4101EJC; it has version YAH814Y0 of the factory software; and it complies to version 6 of the ATA/ATAPI standard. The serial number can be very important when it comes to claiming a replacement drive within the warranty period.

SMART support is enabled, as you can see in the last line of the report, which reflects the BIOS setting. If the last line tells you that SMART support is disabled, you need to enter *smartctl -s on* to enable the SMART support feature.

## And How Are We Feeling Today?

Setting the *smartctl -H* flag queries the health status of a device. As you can see in Listing 2, this is FAILED for our hard disk, and that explains the warning displayed on booting the machine. The hard disk's embedded SMART logic has

**Listing 1: Device Information**

```
# <B>smartctl -i /dev/hda<B>
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
=== START OF INFORMATION SECTION ===
Device Model:     Maxtor 34098H4
Serial Number:    L4101EJC
Firmware Version: YAH814Y0
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   6
ATA Standard is:  ATA/ATAPI-6 T13 1410D revision 0
Local Time is:    Tue Aug 10 12:17:11 2004 CEST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

already reallocated 637 defective sectors on the disk. This is an automatic feature that remaps damaged sectors to a small number of reserved sectors. Unfortunately, the supply of reserved sectors is now exhausted. If the hard disk had checked out okay, the result would have been PASSED.



**Figure 1: SMART capability report for a disk.**

The ATA standard defines a whole bunch of attributes that describe a device's technical properties [3], and manufacturers can add proprietary extensions. There may be some differences in the way the data are stored; attribute #9, which specifies a device's operating hours, in particular. To avoid complications smartmontools has a database with the attributes for a number of models. If your drive model is not in the latest version of the smartmontools database, you can check the smartmontools-database [2] mailing list for an update. The smartmontools homepage tells you how to do so.

The *-A* option tells *smartctl* to output the values of a device's attributes (see Listing 3). You can then compare the latest values with the manufacturer's threshold values. The *VALUE* column has the current value, *WORST* has the worst measured value so far, and *THRESH* the factory threshold value. A current value equal to, or below, the threshold value indicates an error condition for the attribute in question. In our example (Listing 2), the threshold value for the *Reallocated_Sector_Ct* attribute is 63. The current value of 1 is way below that value, and this is what causes the *FAILING_NOW* message in the *WHEN_FAILED* column.

The attribute type is also important to evaluating the health state of a device. The attribute type is displayed in the *TYPE* column in Listing 3, for example. *Old_age* attributes are characteristic of a normal aging process. Attributes with a *Pre-fail* tag are more serious, as they indicate an imminent failure! In this case, you should look to replace the disk in the near future.

## Online and Offline Tests

SMART-enabled hard disks can collect values for specified attributes and store them without affecting performance. There are a few examples of this in Listing 3; the attributes with the *Always* label in the *UPDATED* column. The smartmontools manpage refers to this approach as an online test.

You can run *smartctl* with the *-c* flag to discover which SMART methods your device supports and how much time the tests take (see Figure 1). The same command indicates the test progress in the *Self-test execution status* section.

The so-called offline test can affect disk performance, although you might not notice this in real-life conditions, as the test is interrupted if the operating

### Listing 2: Querying the health status

```
# <B>smartctl -H /dev/hda<B>
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: FAILED!
Drive failure expected in less than 24 hours. SAVE ALL DATA.
Failed Attributes:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE
UPDATED  WHEN_FAILED RAW_VALUE
  5 Reallocated_Sector_Ct   0x0033   001   001   063    Pre-fail  Always
FAILING_NOW 637
```

### Listing 3: Querying Attributes

```
# smartctl -A /dev/hda
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen
=== START OF READ SMART DATA SECTION ===
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x000a   253   252   000    Old_age   Always       -       41
  3 Spin_Up_Time            0x0027   222   222   063    Pre-fail  Always       -       4458
  4 Start_Stop_Count        0x0032   253   253   000    Old_age   Always       -       35
  5 Reallocated_Sector_Ct   0x0033   001   001   063    Pre-fail  Always   FAILING_NOW 637
  6 Read_Channel_Margin     0x0001   253   253   100    Pre-fail  Offline      -       0
  7 Seek_Error_Rate         0x000a   253   252   000    Old_age   Always       -       0
  8 Seek_Time_Performance   0x0027   252   246   187    Pre-fail  Always       -       38203
  9 Power_On_Minutes        0x0032   253   253   000    Old_age   Always       -       16h+46m
  [...]
```

system needs regular access to the disk. To launch the offline test, type *smartctl -t offline <device>*.

To start periodical offline testing – usually every four hours – you specify the *-o on* option, assuming that the manufacturer supports this feature. You can check your device's SMART capabilities to find out (see Figure 1). The *Offline data collection capabilities* should have an entry for *Auto Offline data collection on/off support*. No matter whether you run the offline test explicitly or automatically, it will update the attributes labeled as offline in the *UPDATED* column (Listing 3).

While online and offline tests only collect data, the self-diagnosis test checks the electrical and mechanical characteristics of the device and the read throughput. You can run the self-diagnosis feature while working on a machine; again, the test is interrupted if the operating system needs regular access to the device. If you prefer to disable the device for the duration of the test, you can specify the *-C* option to do so. This so-called captive mode should only be used for media that are not mounted.

A short self-diagnosis (*-t short*) takes a few minutes; a detailed self-diagnosis can take an hour or more depending on the size of the disk you are testing. To launch the extended test, launch *smartctl* with the *-t long* option.

The device stores the test results in a self-test logfile. You can tell *smartctl* to display the logfile (see Listing 4) by specifying the *-l selftest* option. Details of recent tests are displayed at the top of the list. The test automatically stops if it detects an error. The *Remaining* column then tells you how much of the test was not processed as a percentage. If the test detects a hard disk error, it stores the logical address of the first defective block in the *LBA* column. As you can see in line #2 of Listing 4, this test was aborted by my typing *smartctl -X /dev/hdb*.

Besides the self-test logfile, there is also an error log that gives you details on the last five errors that have occurred. The *smartctl -l error* option displays the error log.

## Automated Monitoring with *smartd*

All of the *smartctl* commands we have looked at thus far need to be launched explicitly by the admin user, and that makes them unsuitable for long-term monitoring tasks. Enter *smartd*, a daemon that automatically queries SMART attributes and performs self-tests as specified by the admin. *smartd* logs status and error messages in the syslog and optionally sends email warnings if critical errors are found.

It makes sense to launch *smartd* when you boot your machine and stop the daemon when you shutdown. To support this, smartmontools has the *smartd* shell script besides the */usr/sbin/smartctl* and */usr/sbin/smartd* executables. The script is typically stored below */etc/rc.d/init.d*. You can run *chkconfig --add smartd* to set links at the required runlevels. Suse Linux users may need to move the *smartd* shell script from */etc/rc.d/init.d* to */etc/init.d* before doing so!

To launch *smartd* with specific options, you need to modify the *smartd_opts* variable in the script to reflect your requirements. For example, the *--interval N* option specifies the interval for launching *smartd*. The default is 1800 seconds, that is 30 minutes. You can run *smartd --help* to display your options and the manpage (*man smartd*) has more details.

*smartd* can be configured via entries in */etc/smartd.conf*. If this file does not exist, or if it contains an entry for *DEVICESCAN*, *smartd* will attempt to detect the ATA and SCSI devices in your system when launched, that is, it will look for */dev/hda* through */dev/hdt* and */dev/sda* through */dev/sdz*. It then enables SMART for any devices it finds and launches into the monitoring process.

## Fine Tuning with *smartd.conf*

Entries below the *DEVICESCAN* keyword specify the kind of monitoring to be performed for each device. If you do not add specific entries, the defaults apply for each device type (*ata, scsi, 3ware.*) Run *man smartd.conf* for details of the defaults.

It makes sense not to monitor attributes that are subject to frequent changes, such as the temperature (attributes #194 and #231), or the operating hours (attribute #9); this ensures that your syslog stays readable. To disable these attributes, add a *-I 194 -I 231 -I 9* entry to the *smartd.conf* file. For an email alert type *-m* followed by your email address.

## Mitigating the Danger of Failures

Of course you cannot prevent system failures simply by using *smartd*, as a disk can fail without any warning. But even then, hard disk monitoring can still be useful. It is not always easy to detect a failure. In a worst case scenario, the failure might go unnoticed until your next regular hard disk check, and that would mean even more loss – all the more reason to use smartmontools. ■

### Listing 4: Querying the Self-Test Logfile

```
# smartctl -l selftest /dev/hdb
smartctl version 5.32 Copyright (C) 2002-4 Bruce Allen

=== START OF READ SMART DATA SECTION ===
SMART Self-test log structure revision number 1
Num Test_Description Status                  Remaining LifeTime LBA_of_
                                                       (hours)  first_error
# 1 Extended offline Completed without error 00%    453      -
# 2 Extended offline Aborted by host         90%    451      -
# 3 Short offline    Completed without error 00%    451       -
```

### INFO


[1] smartmontools homepage: *http://smartmontools.sourceforge.net/*

[2] smartmontools-database mailing list homepage: *https://lists.sourceforge.net/lists/listinfo/smartmontools-database*

[3] Guide to SMART attributes: *http://freepgs.com/smart/attributes.php*



**THE AUTHOR**

*Gabriele Pohl is an Oracle DBA, Linux administrator, IT coach, and consultant (http://www.dipohl.com/).*



*Thanks to Steffen Grunewald and Bruce Allen for their support!*

Ci joint dans la version imprimée de ce cours, l'article fondateur de la technologie RAID.

Adresse web du document fourni :
« http://www.csse.monash.edu.au/~rajkumar/superstorage/chap1

## Chapter 1

# A Case for Redundant Arrays of Inexpensive Disks (RAID)

DAVID A. PATTERSON, GARTH GIBSON, AND RANDY H. KATZ

Computer Science Division
Department of Electrical Engineering and Computer Sciences
571 Evans Hall
University of California
Berkeley, CA 94720

## Abstract

Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O. While the capacity of Single Large Expensive Disk (SLED) has grown rapidly, the performance improvement of SLED has been modest. Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability. This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAIDs to an IBM 3380 and a Fujitsu Super Eagle.

## 1.1 Background: Rising CPU and Memory Performance

The users of computers are currently enjoying unprecedented growth in the speed of computers. Gordon Bell said that between 1974 and 1984, single chip computers improved in performance by 40% per year, about twice the rate of minicomputers [Bell 84]. In the following year Bill Joy predicted an even faster growth [Joy 85]

$$MIPS = 2^{Year-1984}$$

Mainframe and supercomputer manufacturers, having difficulty keeping pace with this rapid growth predicted by "Joy's Law", cope by offering multiprocessors as their top-of-the-line product.

But a fast CPU does not a fast system make. Gene Amdahl related CPU speed to main memory size using this rule [Siewiorek 82]

*Each CPU instruction per second requires one byte of main memory.*

If computer system costs are not to be dominated by the cost of memory, then Amdahl's constant suggests that memory chip capacity should grow at the same rate. Gordon Moore predicted that growth rate over 20 years ago

$$transistors/chip = 2^{Year-1964}$$

As predicted by Moore's Law, RAMs have quadrupled in capacity every two [Moore 75] to three years [Moore 86].

Recently this ratio of megabytes of main memory to MIPS has been defined as alpha [Garcia 84], with Amdahl's constant meaning alpha = 1. In part because of the rapid drop of memory prices, main memory sizes have grown faster than CPU speeds and many machines are shipped today with alphas of 3 or higher.

To maintain the balance of costs in computer systems, secondary storage must match the advances in other parts of the system. A key measure of disk technology is the growth in the maximum number of bits that can be stored per square inch, or the bits per inch in a track times the number of tracks per inch Called M A D, for maximal areal density, the "First Law in Disk Density" predicts [Frank87]

$$MAD = 10^{(Year-1971)/10}$$

Magnetic disk technology has doubled capacity and halved price every three years, in line with the growth rate of semiconductor memory, and in practice between 1967 and 1979 the disk capacity of the average IBM data processing system more than kept up with its main memory [Stevens81].

Capacity is not the only memory characteristic that must grow rapidly to maintain system balance, since the speed with which instructions and data are delivered to a

Reprint from Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.109-116, June 1988

CPU also determines its ultimate performance. The speed of main memory has kept pace for two reasons:
(1) the invention of caches, showing that a small buffer can be managed automatically to contain a substantial fraction of memory references;
(2) and the SRAM technology, used to build caches, whose speed has improved at the rate of 40% to 100% per year.

In contrast to primary memory technologies, the performance of single large expensive magnetic disks (SLED) has improved at a modest rate. These mechanical devices are dominated by the seek and the rotation delays: from 1971 to 1981, the raw seek time for a high-end IBM disk improved by only a factor of two while the rotation time did not change [Harker81]. Greater density means a higher transfer rate when the information is found, and extra heads can reduce the average seek time, but the raw seek time only improved at a rate of 7% per year. There is no reason to expect a faster rate in the near future.

To maintain balance, computer systems have been using even larger main memories or solid state disks to buffer some of the I/O activity. This may be a fine solution for applications whose I/O activity has locality of reference and for which volatility is not an issue, but applications dominated by a high rate of random requests for small pieces of data (such as transaction-processing) or by a low number of requests for massive amounts of data (such as large simulations running on supercomputers) are facing a serious performance limitation.

## 1.2 The Pending I/O Crisis

What is the impact of improving the performance of some pieces of a problem while leaving others the same? Amdahl's answer is now known as Amdahl's Law [Amdahl67]

$$S = \frac{1}{(1-f) + f/k}$$

where:
S = the effective speedup,
f = fraction of work in faster mode, and
k = speedup while in faster mode.

Suppose that some current applications spend 10% of their time in I/O. Then when computers are 10X faster - according to Bill Joy in just over three years--then Amdahl's Law predicts effective speedup will be only 5X. When we have computers 100X faster - via evolution of uniprocessors or by multiprocessors - this application will be less than 10X faster, wasting 90% of the potential speedup.

While we can imagine improvements in software file systems via buffering for near term I/O demands, we need innovation to avoid an I/O crisis [Boral 83].

## 1.3 A Solution: Arrays of Inexpensive Disks

Rapid improvements in capacity of large disks have not been the only target of disk designers, since personal computers have created a market for inexpensive magnetic disks. These lower cost disks have lower performance as well as less capacity. Table 1.1 below compares the top-of-the-line IBM 3380 model AK4 mainframe disk, Fujitsu M2361A "Super Eagle" minicomputer disk, and the Conner Peripherals CP 3100 personal computer disk.

| *Characteristics* | *IBM 3380* | *Fujitsu M2361A* | *Conners CP3100* | *3380 v 3100 (>1 means 3100 is better)* | *2361 v 3100* |
|---|---|---|---|---|---|
| Disk diameter (inches) | 14 | 10 5 | 3 5 | 4 | 3 |
| Formatted Data Capacity (MB) | 7500 | 600 | 100 | 01 | 2 |
| Price/MB(controller incl ) | $18-$10 | $20-$17 | $10-$7 | 1-2 5 | 1 7-3 |
| MTTF Rated (hours) | 30,000 | 20,000 | 30,000 | 1 | 1 5 |
| MTTF in practice (hours) | 100,000 | ? | ? | ? | ? |
| No Actuators | 4 | 1 | 1 | 2 | 1 |
| Maximum I/O's/second/Actuator | 50 | 40 | 30 | 6 | 8 |
| Typical I/O's/second/Actuator | ,30 | 24 | 20 | 7 | 8 |
| Maximum I/O's/second/box | 200 | 40 | 30 | 2 | 8 |
| Typical I/O's/second/box | 120 | 24 | 20 | 2 | 8 |
| Transfer Rate (MB/sec) | 3 | 2 5 | 1 | 3 | 4 |
| Power/box (W) | 6,600 | 640 | 10 | 660 | 64 |
| Volume (cu ft ) | 24 | 3 4 | 03 | 800 | 110 |

**Table 1.1** Comparison of IBM 3380 disk model AK4 for mainframe computers, the Fujitsu M2361A "Super Eagle" disk for minicomputers, and the Conners Peripherals CP 3100 disk for personal computers. By "Maximum I/O's/second" we mean the maximum number of average seeks and average rotates for a single sector access. Cost and reliability information on the 3380 comes from widespread experience [IBM 87] [Gawlick87] and the information on the Fujitsu from the manual [Fujitsu 87], while some numbers on the new CP3100 are based on speculation. The price per megabyte is given as a range to allow for different prices for volume discount and different mark-up practices of the vendors. (The 8 watt maximum power of the CP3100 was increased to 10 watts to allow for the inefficiency of an external power supply (since the other drives contain their own power supplies).

One surprising fact is that the number of I/Os per second per actuator in an inexpensive disk is within a factor of two of the large disks. In several of the remaining metrics, including price per megabyte, the inexpensive disk is superior or equal to the large disks.

The small size and low power are even more impressive since disks such as the CP3100 contain full track buffers and most functions of the traditional mainframe controller. Small disk manufacturers can

provide such functions in high volume disks because of the efforts of standards committees in defining higher level peripheral interfaces, such as the ANSI X3.131-1986 Small Computer Synchronous Interface (SCSI). Such standards have encouraged companies like Adaptec to offer SCSI interfaces as single chips, in turn allowing disk companies to embed mainframe controller functions at low cost. Figure 1.1 compares the traditional mainframe disk approach and the small computer disk approach. The same SCSI interface chip embedded as a controller in every disk can also be used as the direct memory access (DMA) device at the other end of the SCSI bus.

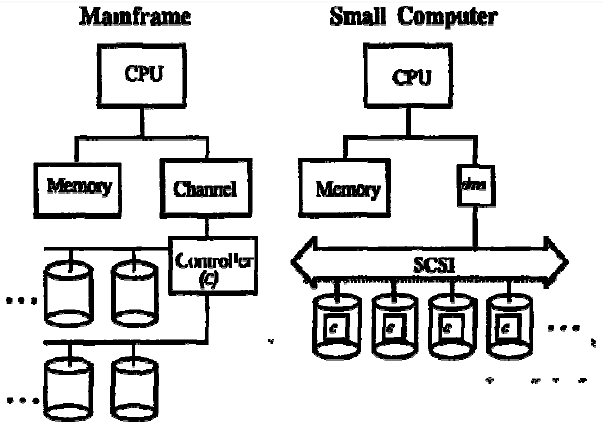



**Figure 1.1** Comparison of organizations for typical mainframe and small computer disk interface. Single chip SCSI interfaces such as the Adaptec AIC-6250 allow the small computer to use a single chip to be the DMA interface as well as provide an embedded controller for each disk [Adaptec87] (The price per megabyte in Table 1.1 includes everything in the shaded boxes above).

Such characteristics lead to the proposal of building I/O systems as arrays of inexpensive disks, either interleaved for the large transfers of supercomputers [Kim 86][Livny 87][Salem 86] or independent for the many small transfers of transaction processing. Using the information in Table I, 75 inexpensive disks potentially have 12 times the I/O bandwidth of the IBM 3380 and the same capacity, with lower power consumption and cost.

## 1.4 Caveats

We cannot explore all issues associated with such arrays in the space available for this paper, so we concentrate on the price-performance and reliability. Our reasoning is that there are no advantages in price-performance or terrible disadvantages in reliability, then there is no need to explore further. We characterize the transaction-processing workload to evaluate performance of a collection of inexpensive disks, but remember that such a collection is just one hardware component of a complete transaction-processing system. While designing a complete TPS based on these ideas is enticing, we will resist that temptation in this paper. Cabling and packaging, certainly an issue in the cost and reliability of an array of many inexpensive disks, is also beyond this paper's scope.

## 1.5 And Now The Bad News: Reliability

The unreliability of disks forces computer systems managers to make backup versions of information quite frequently in case of failure. What would be the impact on reliability of having a hundredfold increase in disks? Assuming a constant failure rate--that is, an exponentially distributed time to failure - and that failures are independent--both assumptions made by disk manufacturers when calculating the Mean Time To Failure (MTTF) - the reliability of an array of disks is:

$$MTTF\ of\ a\ Disk\ Array = \frac{MTTF\ of\ a\ Single\ Disk}{Number\ of\ Disks\ in\ the\ Array}$$

Using the information in Table 1.1, the MTTF of 100 CP 3100 disks is 30,000/100 = 300 hours, or less than 2 weeks. Compared to the 30,000 hour (> 3 years) MTTF of the IBM 3380, this is dismal. If we consider scaling the array to 1000 disks, then the MTTF is 30 hours or about one day, requiring an adjective worse than dismal.

Without fault tolerance, large arrays of inexpensive disks are too unreliable to be useful.

## 1.6 A Better Solution: RAID

To overcome the reliability challenge, we must make use of extra disks containing redundant information to recover the original information when a disk fails. Our acronym for these Redundant Arrays of Inexpensive Disks is RAID. To simplify the explanation of our final proposal and to avoid confusion with previous work, we give the taxonomy of five different organizations of disk arrays, beginning with mirrored disks and progressing through a variety of alternatives with differing performance and reliability. We refer to each organization as a RAID level.

The reader should be forewarned that we describe all levels as if implemented in hardware solely to simplify the presentation, for RAID ideas are applicable to software implementations as well as hardware.

***Reliability*** Our basic approach will be to break the arrays into reliability groups, with each group having extra "check" disks containing the redundant information. When a disk fails we assume that within a short time the failed disk will be replaced and the information will be reconstructed on to the new disk using the redundant information. This time is called the mean time to repair

(MTTR). The MTTR can be reduced if the system includes extra disks to act as "hot" standby spares; when a disk fails, a replacement disk is switched in electronically. Periodically the human operator replaces all failed disks. Here are some other terms that we use:

D = total number of disks with data (not including the extra check disks);

G = number of data disks in a group (not including the extra check disks);

C = number of check disks in a group;

$^{n}G$ = D/G = number of groups.

As mentioned above we make the same assumptions that the disk manufacturers make--that the failures are exponential and independent. (An earthquake or power surge is a situation where an array of disks might not fail independently.) Since these reliability predictions will be very high, we want to emphasize that the reliability is only of the disk-head assembles with this failure model, and not the whole software and electronic system. In addition, in our view the pace of technology means extremely high MTTF are "overkill" - for, independent of expected lifetime, users will replace obsolete disks. After all, how many people are still using 20 years old disks?

The general MTTF calculation for single-error repairing RAID is given in two steps. First, the group MTTF is

$$MTTF_{Group} = \frac{MTTF_{Disk}}{G+C} * \frac{1}{\Pr obability\ of\ another\ failure\ in\ a\ group\ before\ repairing\ the\ dead\ disk}$$

As more formally derived in the appendix, the probability of a second failure before the first has been repaired is

$$\Pr obability\ of\ Another\ Failure = \frac{MTTR}{MTTR_{Disk}/(No\ Disks - 1)} = \frac{MTTR}{MTTF_{Disk}/(G+C-1)}$$

The intention behind the formal calculation in the appendix comes from trying to calculate the average number of second disk failure during the repair time for X single disk failures. Since we assume that disk failure occur at a uniform rate, this average number of second failure during the repair time for X first failure is

$$\frac{X * MTTR}{MTTF\ of\ remaining\ disks\ in\ the\ group}$$

The average number of second failure for a single disk is then

$$\frac{MTTR}{MTTF_{Disk} / No\ of\ remaining\ disks\ in\ the\ group}$$

The MTTF of the remaining disks is just the MTTF of a single disk divided by the number of good disks in the group, giving the result above.

The second step is the reliability of the whole system, which is approximately (since $MTTF_{Group}$ is not quite distributed exponentially)

$$MTTF_{RAID} = \frac{MTTF_{Group}}{^{n}G}$$

Plugging it all together, we get:

$$MTTF_{RAID} = \frac{MTTF_{Disk}}{G+C} * \frac{MTTF_{Disk}}{(G+C-1)*MTTR} * \frac{1}{^{n}G} = \frac{(MTTF_{Disk})^2}{(G+C)*^{n}G*(G+C-1)*MTTR}$$

$$MTTF_{RAID} = \frac{(MTTF_{Disk})^2}{(D+C*^{n}G)*(G+C-1)*MTTR}$$

Since the formula is the same for each level, we make the abstract numbers concrete using these parameters as appropriate: $D$=100 total data disks, $G$=10 data disks per group, $MTTF_{Disk}$ = 30,000 hours, $MTTR$ = 1 hour, with the check disks per group $C$ determined by the RAID level.

***Reliability Overhead Cost*** This is simply the extra check disks, expressed as a percentage of the number of data disks $D$. As we shall see below, the cost varies with RAID level from 100% down to 4%.

***Useable Storage Capacity Percentage*** Another way to express this reliability overhead is in terms of the percentage of the total capacity of data disks and check disks that can be used to store data. Depending on the organization, this varies from a low of 50% to a high of 96%.

***Performance*** Since supercomputer applications and transaction-processing systems have different access patterns and rates, we need different metrics to evaluate both. For supercomputers we count the number of reads and writes per second for large blocks of data, with large defined as getting at least one sector from each data disk in a group. During large transfers all the disks in a group act as a single unit, each reading or writing a portion of the large data block m parallel.

A better measure for transaction-processing systems is the number of individual reads or writes per second. Since transaction-processing systems (e.g., debits/credits) use a read-modify-write sequence of disk accesses, we include that metric as well. Ideally during small transfers each disk in a group can act independently, either reading or writing independent information. In summary supercomputer applications need a high data rate while transaction-processing need a high I/O rate.

For both the large and small transfer calculations we assume the minimum user request is a sector, that a sector is small relative to a track, and that there is enough work to keep every device busy. Figure 1.2 shows the ideal operation of large and small disk accesses in a RAID.

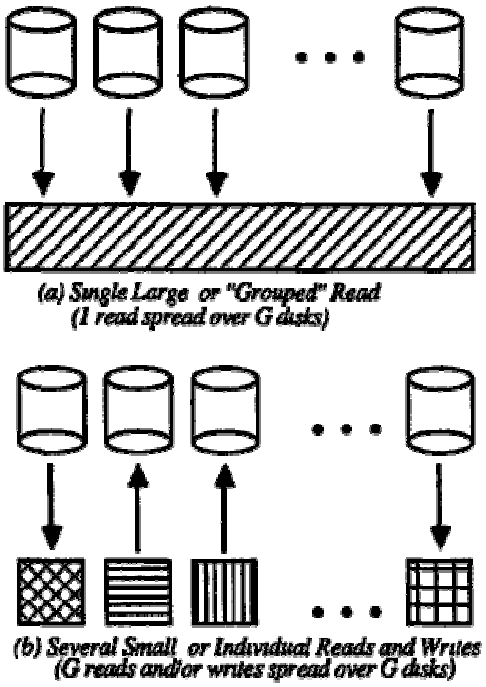



**Figure 1.2** Large transfer vs. small transfer in a group of G disks

The six performance metrics are then the number of reads, writes, and read-modify-writes per second for both large (grouped) or small (individual) transfers. Rather than give absolute numbers for each metric, we calculate efficiency the number of events per second for a single disk. (This is Boral's I/O bandwidth per gigabyte [Boral 83] scaled to gigabytes per disk). In this paper we are after fundamental differences so we use simple, deterministic throughput measures for our performance metric rather than latency.

***Effective Performance Per Disk*** The cost of disks can be a large portion of the cost of a database system, so the I/O performance per disk--factoring in the overhead of the check disks--suggests the cost/performance of a system. This is the bottom line for a RAID.

## 1.7 First Level RAID: Mirrored Disks

Mirrored disks are a traditional approach for improving reliability of magnetic disks. This is the most expensive option since all disks are duplicated (G=l and C=l), and every write to a data disk is also a write to a check disk. Tandem, doubles the number of controllers for fault tolerance, allowing an optimized version of mirrored disks that let reads occur in parallel. Table 1.2I shows the metrics for a Level 1 RAID assuming this optimization.

| *MTTF* | Exceeds Useful Product Lifetime (4,500,000 hrs or > 500 years) | |
|---|---|---|
| *Total Number of Disks* | 2D | |
| *Overhead Cost* | 100% | |
| *Useable Storage Capacity* | 50% | |
| *Events/Sec vs Single Disk* | *Full RAID* | *Efficiency Per Disk* |
| *Large (or Grouped) Reads* | 2D/S | 1 00/S |
| *Large (or Grouped) Writes* | D/S | 50/S |
| *Large (or Grouped) R-M-W* | 4D/3S | 67/S |
| *Small (or Individual) Reads* | 2D | 1 00 |
| *Small (or Individual) Writes* | D | 50 |
| *Small (or Individual) R-M-W* | 4D/3 | 67 |

**Table 1.2** Characteristics of Level 1 RAID. Here we assume that writes are not slowed by waiting for the second write to complete because the slowdown for writing 2 disks is minor compared to the slowdown S for writing a whole group of 10 to 25 disks. Unlike a "pure" mirrored scheme with extra disks that is invisible to the software, we assume an optimized scheme with twice as many controllers allowing parallel reads to all disks, giving full disk bandwidth for large reads and allowing the reads of the read-modify-writes can occur in parallel

When individual accesses are distributed across multiple disks, average queueing, seek, and rotate delays may differ from the single disk case. Although bandwidth may be unchanged, it is distributed more evenly, reducing variance in queueing delay and, if the disk load is not too high, also reducing the expected queueing delay through parallelism [Livny 87]. When many arms seek to the same track then rotate to the described sector, the average seek and rotate time will be larger than the average for a single disk, tending toward the worst case times. This affect should not generally more than double the average access time to a single sector while still getting many sectors in parallel. In the special case of mirrored disks with sufficient controllers, the choice between arms that can read any data sector will reduce the time for average read seek by up to 45% [Bitton 88].

To allow for these factors but to retain our fundamental emphasis we apply a slowdown factor, S, when there are more than two disks in a group. In general, $1 \le S \le 2$ whenever groups of disk work in parallel. With synchronous disks the spindles of all disks in the group are synchronous so that the corresponding sectors of a group disks pass under the heads simultaneously, [Kurzweil 88] so for synchronous disks there is no slowdown and S=1. Since a Level 1 RAID has only one data disk in its group, we assume that the large transfer requires the same

number of disks acting in concert as found in groups of the higher level RAIDs: 10 to 25 disks.

Duplicating all disks can mean doubling the cost of the database system or using only 50% of the disk storage capacity. Such largess inspires the next levels of RAID.

## 1.8 Second Level RAID: Hamming Code for ECC

The history of main memory organizations suggests a way to reduce the cost of reliability. With the introduction of 4K and 16K DRAMs, computer designers discovered that these new devices were subject to losing information due to alpha particles. Since there were many single bit DRAMs in a system and since they were usually accessed in groups of 16 to 64 chips at a time, system designers added redundant chips to correct single errors and to detect double errors in a group. This increased the number of memory chips by 12% to 38%--depending on the size of the group--but it significantly improved reliability.

As long as all the data bits in a group are read or written together, there is no impact on performance. However, reads of less than the group size require reading the whole group to be sure the information is correct, and writes to a portion of the group mean three steps:

1) a read step to get all the rest of the data;

2) a modify step to merge the new and old information;

3) a write step to write the full group, including the check information.

Since we have scores of disks in a RAID and since some accesses are to groups of disks, we can mimic the DRAM solution by bit-interleaving the data across the disks of a group and then add enough check disks to detect and correct a single error. A single parity disk can detect a single error, but to correct an error we need enough check disks to identify the disk with the error. For a group size of 10 data disks (G) we need 4 check disks (C) in total, and if G = 25 then C = 5 [Hamming50]. To keep down the cost of redundancy, we will assume the group size will vary from 10 to 25.

Since our individual data transfer unit is just a sector, bit-interleaved disks mean that a large transfer for this RAID must be at least G sectors. Like DRAMs, reads to a smaller amount still implies reading a full sector from each of the bit-interleaved disks in a group, and writes of a single unit involve the read-modify-write cycle to all the disks. Table 1.3 shows the metrics of this Level 2 RAID.

For large writes, the level 2 system has the same performance as level 1 even though it uses fewer check disks, and so on a per disk basis it outperforms level 1. For small data transfers the performance is dismal either for the whole system or per disk, all the disks of a group must be accessed for a small transfer, limiting the maximum number of simultaneous accesses to D/G. We also must include the slowdown factor S since the access must wait for all the disks to complete.

| *MTTF* | | *Exceeds Useful Lifetime* | | | |
|---|---|---|---|---|---|
| | | *G=10* (494,500 hrs or >50 years) | | *G=25* (103,500 hrs or 12 years) | |
| *Total Number of Disks* | | 1.40D | | 1.20D | |
| *Overhead Cost* | | 40% | | 20% | |
| *Useable Storage Capacity* | | 71% | | 83% | |
| *Events/Sec (vs Single Disk)* | *Full RAID* | *Efficiency Per Disk* *L2* | *L2/L1* | *Efficiency Per Disk* *L2* | *L2/L1* |
| *Large Reads* | *D/S* | 71/S | 71% | 86/S | 86% |
| *Large Writes* | *D/S* | 71/S | 143% | 86/S | 172% |
| *Large R-M-W* | *D/S* | 71/S | 107% | 86/S | 129% |
| *Small Reads* | *D/SG* | 07/S | 6% | 03/S | 3% |
| *Small Writes* | *D/2SG* | 04/S | 6% | 02/S | 3% |
| *Small R-M-W* | *D/SG* | 07/S | 9% | 03/S | 4% |

**Table 1.3** Characteristics of a Level 2 RAID. The L2/LI column gives the % performance of level 2 in terms of level 1 (>100% means L2 is faster). As long as the transfer unit is large enough to spread over all the data disks of a group, the large I/Os get the full bandwidth of each disk, divided by S to allow all disks in a group to complete. Level I large reads are faster because data is duplicated and so the redundancy disks can also do independent accesses. Small I/Os still require accessing all the disks in a group, so only D/G small I/Os can happen at a time, again divided by S to allow a group of disks to finish. Small Level 2 writes are like small R-M-W because the full sectors must be read before new data can be written onto part of each sector.

Thus level 2 RAID is desirable for supercomputers but inappropriate for transaction processing systems, with increasing group size increasing the disparity in performance per disk for the two applications. In recognition of this fact, Thinking Machines Incorporated announced a Level 2 RAID this year for its Connection Machine supercomputer called the "Data Vault," with G = 32 and C = 8, including one hot standby spare [Hillis 87].

Before improving small data transfers, we concentrate once more on lowering the cost.

## 1.9 Third Level RAID: Single Check Disk Per Group

Most check disks in the level 2 RAID are used to determine which disk failed, for only one redundant parity disk is needed to detect an error. These extra disks are truly “redundant” since most disk controllers can already detect if a disk a failed either through special signals provided in the disk interface or the extra checking information at the end of a sector to detect and correct soft errors. So information on the failed disk can be

reconstructed by calculating the parity of the remaining good disks and then comparing bit-by-bit to the parity calculated for the original full group. When these two parities agree, the failed bit was a 0; otherwise it was a 1. If the check disk is the failure, just read all the data disks and store the group parity in the replacement disk.

Reducing the check disks to one per group (C=l) reduces the overhead cost to between 4% and 10% for the group sizes considered here. The performance for the third level RAID system is the same as the Level 2 RAID, but the effective performance per disk increases since it needs fewer check disks. This reduction in total disks also increases reliability, but since it is still larger than the useful lifetime of disks, this is a minor point. One advantage of a level 2 system is that the extra check information associated with each sector to correct soft errors is not needed, increasing the capacity per disk by perhaps 10%. Level 2 also allows all soft errors to be corrected "on the fly" without having to reread a sector. Table 1.4 summarizes the third level RAID characteristics and Figure 1.3 compares the sector layout and check disks for levels 2 and 3.

| *MTTF* | | *Exceeds Useful Lifetime* | | | | | |
|---|---|---|---|---|---|---|---|
| | | *G=l0* (820,000 hrs or >90 years) | | | *G=25* (346,000 hrs or 40 years) | | |
| *Total Number of Disks* | | 1 10D | | | 1 04D | | |
| *Overhead Cost* | | 10% | | | 4% | | |
| *Useable Storage Capacity* | | 91% | | | 96% | | |
| *Events/Sec (vs Single Disk)* | *Full RAID* | *Efficiency Per Disk L3* | *L3/L2* | *L3/L1* | *Efficiency Per Disk L3* | *L3/L2* | *L3/L1* |
| *Large Reads* | *D/S* | 91/S | 127% | 91% | 96/S | 112% | 96% |
| *Large Writes* | *D/S* | 91/S | 127% | 182% | 96/S | 112% | 192% |
| *Large R-M-W* | *D/S* | 91/S | 127% | 136% | 96/S | 112% | 142% |
| *Small Reads* | *D/SG* | 09/S | 127% | 8% | 04/S | 112% | 3% |
| *Small Writes* | *D/2SG* | 05/S | 127% | 8% | 02/S | 112% | 3% |
| *Small R-M-W* | *D/SG* | 09/S | 127% | 11% | 04/S | 112% | 5% |

**Table 1.4** Characteristics of a Level 3 RAID. The L3/L2 column gives the % performance of L3 in terms of L2 and the L3/L1 column gives it in terms of L1 (>100% means L3 is faster). The performance for the full systems is the same in RAID levels 2 and 3, but since there are fewer check disks the performance per disk improves.

Park and Balasubramanian proposed a third level RAID system without suggesting a particular application [Park86]. Our calculations suggest it is a much better match to supercomputer applications than to transaction processing systems. This year two disk manufacturers have announced level 3 RAIDs for such applications using synchronized 5.25 inch disks with G=4 and C=l: one from Maxtor and one from Micropolis [Maginnis 87].

This third level has brought the reliability overhead cost to its lowest level, so in the last two levels we improve performance of small accesses without changing cost or reliability.

## 1.10 Fourth Level RAID: Independent Reads/Writes

Spreading a transfer across all disks within the group has the following advantage:

- Large or grouped transfer time is reduced because transfer bandwidth of the entire array can be exploited.

But it has the following disadvantages as well:

- Reading/writing to a disk in a group requires reading/writing to all the disks in a group; levels 2 and 3 RAIDs can perform only one I/O at a time per group.
- If the disks are not synchronized, you do not see average rotational delays, the observed delays should move towards the worst case, hence the S factor in the equations above.

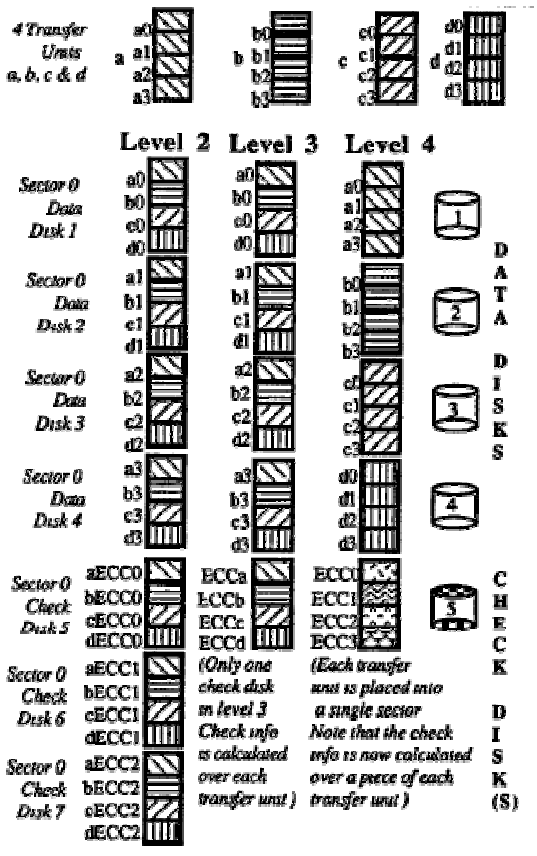



Figure 1.3 Comparison of location of data and check information in sector for RAID level 2, 3, and 4 for G=4. Not shown is the small amount of check information per sector added by the disk controller to detect and correct soft errors within a sector. Remember that we use physical

sector numbers and hardware control to explain these ideas, but RAID can be implemented by software using logical sectors and disks.

This fourth level RAID improves performance of small transfers through parallelism--the ability to do more than one I/O per group at a time. We no longer spread the individual transfer information across several disks, but keep each individual unit in a single disk.

The virtue of bit-interleaving is the easy calculation of the Hamming code needed to detect or correct errors in level 2. But recall that in the third level RAID we can rely on the disk controller to detect errors within a single disk sector. Hence, if we store an individual transfer unit in a single sector, we can detect errors on an individual read without accessing any other disk. Figure 3 shows the different ways the information is stored in a sector for RAID levels 2, 3, and 4. By storing a whole transfer unit in a sector, reads can be independent and operate at the maximum rate of a disk yet still detect errors. Thus the primary change between level 3 and 4 is that we interleave data between disks on a sector level rather than at the bit level.

At first thought you might expect that an individual write to a single sector still involves all the disks in a group since (1) the check disk must be rewritten with the new parity data, and (2) the rest of the data disks must be read to be able to calculate the new parity data. Recall that each parity bit is just a single exclusive OR of all the corresponding data bits in a group. In level 4 RAID, unlike level 3, the parity calculation is much simpler since if we know the old data value and the old parity value as well as the new data value, we can calculate the new parity information as follows:

*new parity = (old data xor new data ) xor old parity*

In level 4 a small write then uses 2 disks to perform 4 accesses - 2 reads and 2 writes - while a small read involves only one read on one disk. Table V summarizes the fourth level RAID characteristics. Note that all small accesses improve - dramatically for the reads - but the small read-modify-write is still so slow relative to a level 1 RAID that its applicability to transaction processing is doubtful. Recently Salem and Garcia-Molina proposed a Level 4 system [Salem 86].

Before proceeding to the next level we need to explain the performance of small writes in Table 1.5 (and hence small read-modify-writes since they entail the same operations in this RAID). The formula for the small writes divides D by 2 instead of 4 because 2 accesses can proceed in parallel: the old data and old parity can be read at the same time and the new data and new parity can be written at the same time. The performance of small writes is also divided by G because the single check disk in a group must be read and written with every small write in that group, thereby limiting the number of writes that can be performed at a time to the number of groups.

The check disk is the bottleneck, and the final level RAID removes this bottleneck.

| *MTTF* | | Exceeds Useful Lifetime | | | | | |
|---|---|---|---|---|---|---|---|
| | | *G=10* (820,000 hrs or >90 years) | | | *G=25* (346,000 hrs or 40 years) | | |
| *Total Number of Disks* | | 1.10D | | | 1.04D | | |
| *Overhead Cost* | | 10% | | | 4% | | |
| *Useable Storage Capacity* | | 91% | | | 96% | | |
| *Events/Sec (vs Single Disk)* | *Full RAID* | *Efficiency Per Disk* *L4* | *L4/L3* | *L4/L1* | *Efficiency Per Disk* *L4* | *L4/L3* | *L4/L1* |
| *Large Reads* | *D/S* | 91/S | 100% | 91% | 96/S | 100% | 96% |
| *Large Writes* | *D/S* | 91/S | 100% | 182% | 96/S | 100% | 192% |
| *Large R-M-W* | *D/S* | 91/S | 100% | 136% | 96/S | 100% | 146% |
| *Small Reads* | *D* | 91 | 1200% | 91% | 96 | 3000% | 96% |
| *Small Writes* | *D/2G* | 05 | 120% | 9% | 02 | 120% | 4% |
| *Small R-M-W* | *D/G* | 09 | 120% | 14% | 04 | 120% | 6% |

**Table 1.5** Characteristics of a Level 4 RAID. The L4/L3 column gives the % performance of L4 in terms of L3 and the L4/L1 column gives it in terms of L1 (>100% means U is faster). Small reads improve because they no longer tie up a whole group at a time. Small writes and R-M-Ws improve some because we make the same assumptions as we made in Table 1.2: the slowdown for two related I/Os can be ignored because only two disks are involved.

## 1.11 Fifth Level RAID: No Single Check Disk

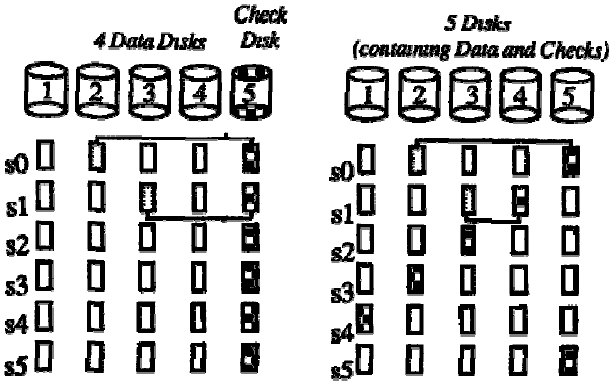
While level 4 RAID achieved parallelism for reads, writes are still limited to one per group since every write to a group must read and write the check disk. The final level RAID distributes the data and check information across all the disks--including the check disks. Figure 4 compares the location of check information in the sectors of disks for levels 4 and 5 RAIDs.

The performance impact of this small change is large since RAID level 5 can support multiple individual writes per group. For example, suppose in Figure 1.4 above we want to write sector 0 of disk 2 and sector 1 of disk 3. As shown on the left Figure 1.4, in RAID level 4 these writes must be sequential since both sector 0 and sector 1 of disk 5 must be written. However, as shown on the right, in RAID level 5 the writes can proceed in parallel since a write to sector 0 of disk 2 still involves a write to disk 5 but a write to sector 1 of disk 3 involves a write to disk 4.

These changes bring RAID level 5 near the best of both worlds: small read-modify-writes now perform close to the speed per disk of a level 1 RAID while keeping the large transfer performance per disk and high useful storage capacity percentage of the RAID levels 3 and 4. Spreading

the data across all disks even improves the performance of small reads, since there is one more disk per group that contains data. Table 1.6 summarizes the characteristics of this RAID.

Keeping in mind the caveats given earlier, a Level 5 RAID appears very attractive if you want to do just supercomputer applications, or just transaction processing when storage capacity is limited, or if you want to do both supercomputer applications and transaction processing.




*(a) Check information for Level 4 RAID for G=4 and C=1 The sectors are shown below the disks (The checked areas indicate the check information ) Writes to s0 of disk 2 and s1 of disk 3 imply writes to s0 and s1 of disk 5 The check disk (5) becomes the write bottleneck.*

*(b) Check information for Level 5 RAID for G=4 and C=1 The sectors are shown below the disks, with the check information and data spread evenly through all the disks Writes to s0 of disk 2 and s1 of disk 3 still imply 2 writes, but they can be split across 2 disks to s0 of disk 5 and to s1 of disk 4*

**Figure 1.4** Location of check information per sector for Level 4 RAID vs. Level 5 RAID

| MTTF | | Exceeds Useful Lifetime | | | Exceeds Useful Lifetime | | |
|---|---|---|---|---|---|---|---|
| | | G=10 (820,000 hrs or >90 years) | | | G=25 (346,000 hrs or 40 years) | | |
| Total Number of Disks | | 1.10D | | | 1.04D | | |
| Overhead Cost | | 10% | | | 4% | | |
| Useable Storage Capacity | | 91% | | | 96% | | |
| Events/Sec (vs Single Disk) | Full RAID | Efficiency Per Disk L5 | L5/L4 | L5/L1 | Efficiency Per Disk L5 | L5/L4 | L5/L1 |
| Large Reads | D/S | 91/S | 100% | 91% | 96/S | 100% | 96% |
| Large Writes | D/S | .91/S | 100% | 182% | 96/S | 100% | 192% |
| Large R-M-W | D/S | 91/S | 100% | 136% | 96/S | 100% | 144% |
| Small Reads | (1+C/G)D | 1.00 | 110% | 100% | 1.00 | 104% | 100% |
| Small Writes | (1+C/G)D/4 | .25 | 550% | 50% | .25 | 1300% | 50% |
| Small R-M-W | (1+C/G)D/2 | .50 | 550% | 75% | .50 | 1300% | 75% |

Table 1.6 Characteristics of a Level 5 RAID. The L5/L4 column gives the % performance of L5 in terms of L4 and the L5/L1 column gives it in terms of L1 (~100% means L5 is faster). Because reads can be spread over all disks, including what were check disks in level 4, all small I/Os improve by a factor of 1 + C/G. Small writes and R-M-Ws improve because they are no longer constrained by group size, getting the full disk bandwidth for the 4 I/O's associated with these accesses. We again make the same assumptions as we made in Tables II and V: the slowdown for two related I/Os can be ignored because only two disks are involved.

## 1.12 Discussion

Before concluding the paper, we wish to note a few more interesting points about RAIDs. The first is that while the schemes for disk stripping and parity support were presented as if they were done by hardware, there is no necessity to do so. We just give the method, and the decision between hardware and software solutions is strictly one of cost and benefit. For example, in cases where disk buffering is effective, there is no extra disks reads for level 5 small writes since the old data and old parity would be in main memory, so software would give the best performance as well as the least cost.

In this paper we have assumed the transfer unit is a multiple of the sector. As the size of the smallest transfer unit grows larger than one sector per drive--such as a full track with an I/O protocol that supports data returned out-of-order--then the performance of RAIDs improves significantly because of the full track buffer in every disk. For example, if every disk begins transferring to its buffer as soon as it reaches the next sector, then S may reduce to less than 1 since there would be no rotational delay. With transfer units the size of a track, it is not even clear if synchronizing the disks in a group improves RAID performance.

This paper makes two separable points: the advantages of building I/O systems from personal computer disks and the advantages of five different disk array organizations, independent of disks used in those array. The later point starts with the traditional mirrored disks to achieve acceptable reliability, with each succeeding level improving:

- the data rate, characterized by a small number of requests per second for massive amounts of sequential information (supercomputer applications);
- the I/O rate, characterized by a large number of read-modify-writes to a small amount of random information (transaction-processing);
- or the useable storage capacity;

or possibly all three.

Figure 1.5 shows the performance improvements per disk for each level RAID. The highest performance per disk comes from either Level 1 or Level 5. In transaction-processing situations using no more than 50% of storage

capacity, then the choice is mirrored disks (Level 1). However, if the situation calls for using more than 50% of storage capacity, or for supercomputer applications, or for combined supercomputer applications and transaction processing, then Level 5 looks best. Both the strength and weakness of Level 1 is that it duplicates data rather than calculating check information, for the duplicated data improves read performance but lowers capacity and write performance, while check data is useful only on a failure.

Inspired by the space-time product of paging studies [Denning 78], we propose a single figure of ment called the space-speed product, the useable storage fraction times the efficiency per event. Using this metric, Level 5 has an advantage over Level 1 of 17 for reads and 33 for writes for G=10.

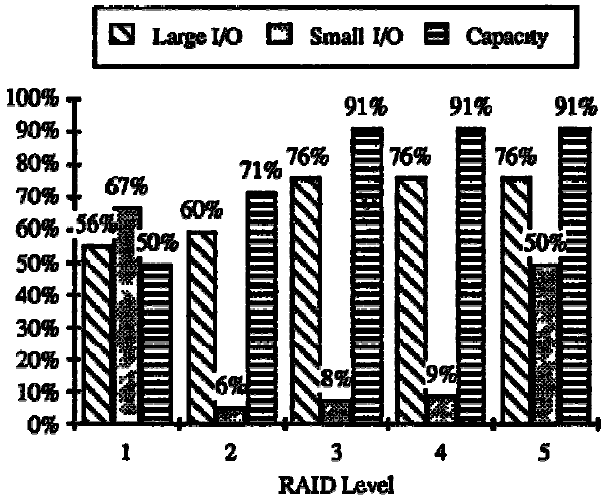



**Figure 1.5** Plot of Large (Grouped) and Small (Individual) Read-Modify-Writes per second per disk and usable storage capacity for all five levels of RAID (D=100, G=10). We assume a single S factor uniform for all levels with S=13 where it is needed.

Let us return to the first point, the advantages of building I/O system from personal computer disks. Compared to traditional Single Large Expensive Disks (SLED), Redundant Arrays of Inexpensive Disks (RAID) offer significant advantages for the same cost. Table 1.7 compares a level 5 RAID using 100 inexpensive data disks with a group size of 10 to the IBM 3380. As you can see, a level 5 RAID offers a factor of roughly 10 improvement in performance, reliability, and power consumption (and hence air conditioning costs) and a factor of 3 reduction in size over this SLED. Table 1.7 also compares a level 5 RAID using 10 inexpensive data disks with a group size of 10 to a Fujitsu M2361A "Super Eagle". In this comparison RAID offers roughly a factor of 5 improvement in performance, power consumption, and size with more than two orders of magnitude improvement in (calculated) reliability.

| *Characteristics* | *RAID 5L (100 10) (CP3100)* | *SLED (IBM 3380)* | *RAID v SLED (>1 better for RAID)* | *RAID 5L (10,10) (CP3100)* | *SLED (Fujitsu M2361)* | *RAID v SLED (>1 better for RAID)* |
|---|---|---|---|---|---|---|
| Formatted Data Capacity (MB) | 10,000 | 7,500 | 1 33 | 1,000 | 600 | 1 67 |
| Price/MB (controller incl.) | $11-$8 | $18-$10 | 2 2- 9 | $11-$8 | $20-$17 | 2 5-1 5 |
| Rated MTTF (hours) | 820,000 | 30,000 | 27 3 | 8,200,000 | 20,000 | 410 |
| MTTF in practice (hours) | ? | 100,000 | ? | ? | ? | ? |
| No Actuators | 110 | 4 | 22 5 | 11 | 1 | 11 |
| Max I/O's/Actuator | 30 | 50 | 6 | 30 | 40 | 8 |
| Max Grouped RMW/box | 1250 | 100 | 12 5 | 125 | 20 | 6 2 |
| Max Individual RMW/box | 825 | 100 | 8 2 | 83 | 20 | 4 2 |
| Typ I/O's/Actuator | 20 | 30 | 7 | 20 | 24 | 8 |
| Typ Grouped RMW/box | 833 | 60 | 13 9 | 83 | 12 | 6 9 |
| Typ Individual RMW/box | 550 | 60 | 9 2 | 55 | 12 | 4 6 |
| Volume/Box (cubic feet) | 10 | 24 | 2 4 | 1 | 3 4 | 3 4 |
| Power/box (W) | 1100 | 6,600 | 6 0 | 110 | 640 | 5 8 |
| Min Expansion Size (MB) | 100-1000 | 7,500 | 7 5-75 | 100-1000 | 600 | 0 6-6 |

Table 1.7 Comparison of IBM 3380 disk model AK4 to Level 5 RAID using 100 Conners & Associates CP 3100s disks and a group size of 10 and a comparison of the Fujitsu M2361A "Super Eagle" to a level 5 RAID using 10 inexpensive data disks with a group size of 10. Numbers greater than 1 in the comparison columns favor the RAID.

RAID offers the further advantage of modular growth over SLEDs. Rather than being limited to 7,500 MB per increase for $100,000 as in the case of this model of IBM disk, RAIDs can grow at either the group size (1000 MB for $11,000) or, if partial groups are allowed, at the disk size (100 MB for $1,100). The flip side of the coin is that RAID also makes sense in systems considerably smaller than a SLED. Small incremental costs also makes hot standby spares practical to further reduce MTTR and thereby increase the MTTF of a large system. For example, a 1000 disk level 5 RAID with a group size of 10 and a few standby spares could have a calculated MTTF of 45 years.

A final comment concerns the prospect of designing a complete transaction processing system from either a Level 1 or level 5 RAID. The drastically lower power per megabyte of inexpensive disks allows systems designers to consider battery backup for the whole disk array--the power needed for 110 PC disks is less than two Fujitsu Super Eagles. Another approach would be to use a few such disks to save the contents of battery backed-up main memory in the event of an extended power failure. The smaller capacity of these disks also ties up less of the database during reconstruction, leading to higher availability. (Note that Level 5 ties up all the disks in a group in event of failure while Level 1 only needs the single mirrored disk during reconstruction, giving Level 1 the edge in availability).

## 1.13 Conclusion

RAIDs offer a cost effective option to meet the challenge of exponential growth in the processor and memory speeds.

We believe the size reduction of personal computer disks is a key to the success of disk arrays, just as Gordon Bell argues that the size reduction of microprocessors is a key to the success m multiprocessors [Bell 85]. In both cases the smaller size simplifies the interconnection of the many components as well as packaging and cabling. While large arrays of mainframe processors (or SLEDs) are possible, it is certainly easier to construct an array from the same number of microprocessors (or PC drives). Just as Bell coined the term "multi" to distinguish a multiprocessor made from microprocessors, we use the term "RAID" to identify a disk array made from personal computer disks.

With advantages in cost-performance, reliability, power consumption, and modular growth, we expect RAIDs to replace SLEDs in future I/O systems. There are, however, several open issues that may bare on the practicality of RAIDs:

- *What is the impact of a RAID on latency?*
- *What is the impact on MTTF calculations of non-exponential failure assumptions for individual disks?*
- *What will be the real lifetime of a RAID vs. calculated MTTF using the independent failure model?*
- *How would synchronized disks affect level 4 and 5 RAID performance?*
- *How does "slowdown" S actually behave? [Livny 87]*
- *How do defective sectors affect RAID?*
- *How do you schedule I/O to level 5 RAIDs to maximize write parallelism?*
- *Is there locality of reference of disk accesses in transaction processing?*
- *Can information be automatically redistributed over 100 to 1000 disks to reduce contention?*
- *Will disk controller design limit RAID performance?*
- *How should 100 to 1000 disks be constructed and physically connected to the processor?*
- *What is the impact of cabling on cost, performance, and reliability?*
- *Where should a RAID be connected to a CPU so as not to limit performance? Memory bus? I/O bus? Cache?*
- *Can a file system allow differ striping policies for different files?*
- *What is the role of solid state disks and WORMs in a RAID?*
- *What is the impact on RAID of "parallel access" disks (access to every surface under the read/write head in parallel)?*

## Acknowledgements

We wish to acknowledge the following people who participated in the discussions from which these ideas emerged: Michael Stonebraker, John Ousterhout, Doug Johnson, Ken Lutz, Anapum Bhide, Gaetano Boriello, Mark Hill, David Wood, and students in SPATS seminar offered at U. C. Berkeley in Fall 1987. We also wish to thank the following people who gave comments useful in the preparation of this paper: Anapum Bhide, Pete Chen, Ron David, Dave Ditzel, Fred Douglis, Dieter Gawlick, Jim Gray, Mark Hill, Doug Johnson, Joan Pendleton, Martin Schulze, and Herve Touati. This work was supported by the National Science Foundation under grant #MIP-8715235.

## Appendix: Reliability Calculation

Using probability theory we can calculate the $MTTF_{Group}$. We first assume independent and exponential failure rates. Our model uses a biased coin with the probability of heads being the probability that a second failure will occur within the MTTR of a first failure. Since disk failures are exponential.

$$\Pr obability\,(at\ least\ one\ of\ the\ remaining\ disks\ \ failing\ in\ MTTR) = [1 - (e^{-MTTR\ /\ MTTF_{Disk}})(G + C - 1)]$$

In all practical cases,

$$MTTR << \frac{MTTF_{Disk}}{G + C}$$

and since $(1 - e^x)$ is approximately X for $0 < X << 1$:

$$\Pr obability\,(at\ least\ one\ of\ the\ remaining\ disks\ \ failing\ in\ MTTR) = MTTR * (G + C - 1) / MTTF_{Disk}$$

Then that on a disk failure we flip this coin

heads => a system crash, because a second failure occurs before the first was repaired;

tails => recover from error and continue.

Then

$$\begin{aligned} MTTF_{Group} &= Expected[Time\ between\ Failures] \\ &\quad * Expected[no\ of\ \ flips\ until\ \ first\ heads] \\ &= \frac{Expected[Time\ between\ Failures]}{\Pr obability(heads)} \\ &= \frac{MTTF_{Disk}}{(G + C) * (MTTR * (G + C - 1) / MTTF_{Disk})} \end{aligned}$$

$$MTTF_{Group} = \frac{(MTTF_{Disk})^2}{(G + C) * (G + C - 1) * MTTR}$$

Group failure is not precisely exponential in our model, but we have validated this simplifying assumption for

practical cases of MTTR << MTTF/(G+C). This makes the MTTF of the whole system just $MTTF_{Group}$ divided by the number of groups, $^{n}G$.

## 1.14 References

[Bell 84] C. G. Bell, "The Mini and Micro Industries," *IEEE Computer*, Vol. 17, No. 10 (October 1984), pp. 14-30.

[Joy 85] B. Joy, presentation at ISSCC '85 panel session, Feb. 1985.

[Siewiorek 82] D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples*, p. 46.

[Moore 75] G E Moore, "Progress in Digital Integrated Electronics," *Proc. IEEE Digital Integrated Electronic Device Meeting*, (1975), p. 11

[Myers 86] G. J. Myers, A. Y. C. Yu, and D. L. House, "Microprocessor Technology Trends," *Proc. IEEE*, Vol. 74, no 12, (December 1986), pp. 1605-1622

[Garcia 84] H. Garcia Molina, R. Cullingford, P. Honeyman, R.. Lipton, "The Case for Massive Memory," Technical Report 326, Dept of EE and CS. Princeton Univ. May 1984

[Myers 86] W. Myers, "The Competitiveness of the United States Disk Industry," *IEEE Computer*, Vol. 19, No. 11 (January 1986), pp. 85-90.

[Frank 87] P. D. Frank, "Advances in Head Technology," presentation at *Challenges in Disk Technology Short Course*, Institute for Information Storage Technology, Santa Clara University, Santa Clara, California, December 15-17.1987

[Stevens 81] L. D. Stevens, "The Evolution of Magnetic Storage," *IBM Journal of Research and Development*, Vol. 25, No. 5, September 1981, pp. 663-675.

[Harker 81] J. M. Harker et al., "A Quarter Century of Disk File Innovation," *ibid*, pp. 677-689.

[Amdahl 67] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proceedings AFIPS 1967 Spring Joint Computer Conference* Vol. 30 (Atlantic City, New Jersey April 1967), pp. 483-485.

[Boral 83] H. Boral and D. J. DeWitt, "Database Machines: An Ideas Whose Time Has Passed? A Critique of the Future of Database Machines," *Proc. International Conf. on Database Machines*, Edited by H.O. Leilich and M. Misskoff, Springer-Verlag, Berlin, 1983

[IBM 87] "IBM 3380 Direct Access Storage Introduction," IBM GC 26-4491-0, September 1987.

[Gawlick 87] D. Gawlick, private communication, Nov. 1987.

[Fujitsu 87] "M2361A Mini-Disk Drive Engineering Specifications," (revised) Feb. 1987, B03P-4825-000lA.

[Adaptec 87] AIC-6250, *IC Product Guide*, Adaptec, stock # DB0003-00 rev. B, 1987, p. 46

[Livny87] Livny, M., S. Khoshafian, H. Boral. "Multi-disk management algorithms." *Proc. of ACM SIGMETRICS*, May 1987

[Kim 86] M. Y. Kim. "Synchronized disk interleaving," *IEEE Trans. on Computers*, vol. C-35, no. 11, Nov 1986

[Salem 86] K. Salem and Garcia-Molina, H., "Disk Striping," *IEEE 1986 Int. Conf. on Data Engineering*, 1986

[Bitton 88] D. Bitton and J. Gray, "Disk Shadowing," *in press*, 1988

[Kurzweil88] F. Kurzweil, "Small Disk Arrays - The Emerging Approach to High Performance," presentation at Spring COMPCON 88, March 1, 1988, San Francisco, CA

[Hamming 50] R. W. Hamming, "Error Detecting and Correcting Codes," *The Bell System Technical Journal*, Vol. XXVI, No. 2 (April 1950), pp. 147-160.

[Hillis 87] D. Hillis, private communication, October, 1987.

[Park86] A. Park and K. Balasubramanian, "Providing Fault Tolerance in Parallel Secondary Storage Systems," Department of Computer Science, Princeton University, CS-TR-057-86, November 7, 1986.

[Maginnis 87] N. B. Maginnis, "Store More, Spend Less: Mid-range Options Abound9" *Computerworld*, November 16, 1987, p. 71.

[Denning 78] P. J. Denning and D. F. Slutz, "Generalized Working Sets for Segment Reference Strings," *CACM*, vol. 21, no. 9. (Sept. 1978) pp. 750-759

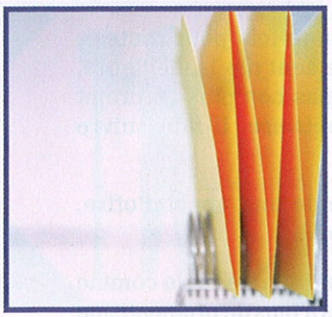
[Bell 85] Bell, C. G, "Multis: a new class of multiprocessor computers," *Science*, vol. 228 (April 26.1985) 462-467

Ci joint dans la version imprimée de ce cours, un article intitué « L'avenir des systèmes de fichiers » du numéro 124 de Février 2010 de LINUX Magazine France.

Adresse web : « `http://www.gnulinuxmag.com` »



# L'avenir des systèmes de fichiers



*Les systèmes de fichiers sont en pleine mutation pour répondre aux nouveaux usages. Plusieurs approches sont envisagées, répondant à des besoins spécifiques. Cela aura un impact sur les applications à développer. Regardons les différentes approches et les produits proposés.*

Auteur

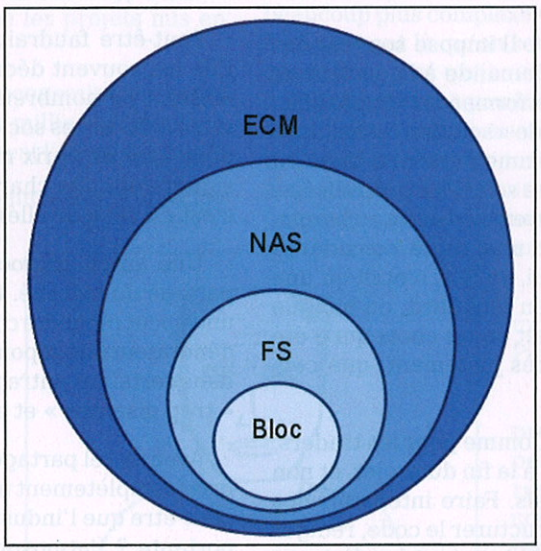
- Philippe PRADOS

Pour livrer des fichiers à des applications, plusieurs niveaux d'abstraction sont possibles. Des systèmes d'ECM proposent d'organiser les fichiers, de gérer leurs cycles de vie ou l'évolution des versions. Par exemple, Alfresco **[1]** publie des fichiers sous différents protocoles.

Puis viennent les composants s'occupant de publier des fichiers sur le réseau, comme NFS ou SAMBA. Ce sont des NAS.

Viennent ensuite les systèmes de fichiers en charge d'organiser le disque dur pour mémoriser les données et les organiser dans des fichiers.

Enfin, interviennent les drivers de niveau bloc, en charge d'écrire physiquement sur les disques, morceau par morceau.




Dans le cadre de cet article, nous nous intéressons aux gestionnaires de fichiers ainsi qu'aux systèmes de fichiers et leurs relations avec les niveaux inférieurs.

Les gestionnaires de fichiers sont généralement conformes aux spécifications POSIX, définies depuis 1988 par l'IEEE **[2]** et formellement désignées par IEEE 1003. Ces standards ont émergé d'un projet d'uniformisation des API des logiciels destinés à fonctionner sur des variantes du système d'exploitation UNIX. Les spécifications étant payantes et non publiques, le standard Single UNIX Specification **[3]** propose une version allégée publique. Les chapitres 2.5 et 2.6 du chapitre « System Interface » traitent spécifiquement des API d'accès aux fichiers.

## 1 LIMITATIONS FONCTIONNELLES

Les systèmes de fichiers contemporains ont plusieurs difficultés :

- la taille des fichiers augmente considérablement ;
- les disques doublent leurs capacités tous les deux ans ;
- il persiste à y avoir une erreur par mega-byte ;
- l'augmentation des volumes entraîne des problèmes de performance et de contrôle de qualité sur de gros volumes. (un fsck ext3 sur 1 To est supérieur à 4 heures) ;
- il est de plus en plus difficile de gérer efficacement les centres de données ;
- il n'y a pas de correction de la corruption des données ;
- des limitations (nombre de fichiers, taille des fichiers, etc.) ;
- une gestion peu souple (redimensionnements, partitionnements, volumes, ...).

D'où la volonté de revoir les gestionnaires de fichiers pour offrir de meilleures performances, adaptées aux nouveaux besoins.

# 2 PRINCIPES ARCHITECTURAUX

Les systèmes de fichiers organisent les masses importantes de données disponibles sur les disques physiques en informations structurées, permettant la lecture et la modification de fichiers dans de bonnes conditions de performance et d'intégrité.

Deux familles de gestionnaires de fichiers traitent d'architectures différentes. La famille « orienté OS » permet une gestion des fichiers par le système d'exploitation. La famille « orienté Cluster » permet la gestion d'un système de fichiers par différents nœuds d'une grappe de serveur.

## 2.1 Systèmes de gestion de fichiers orientés OS

Les systèmes de gestion de fichiers orientés OS sont en charge de l'organisation des fichiers sur un disque physique. Initialement, cela faisait partie du cœur des systèmes d'exploitation. Depuis de nombreuses années, le système d'exploitation propose une abstraction (VFS) permettant l'intégration de différents gestionnaires de fichiers par ajout de composant logiciel.

Les données et les métadonnées sont organisées pour minimiser autant que possible le déplacement de la tête de lecture et limiter le nombre d'accès en lecture et en écriture. Inévitablement, les structures se dégradent, entraînant un besoin de défragmentation des données. De plus, lors d'un dysfonctionnement (coupure électrique, échec de lecture ou d'écriture), il doit être possible de vérifier rapidement le système de fichiers pour garantir son intégrité.

Les nouveaux usages des systèmes de fichiers entraînent de nouveaux besoins que les évolutions à venir se proposent de combler :

- intégrité, vérification et réparation plus rapides des disques ;
- réduction de la fragmentation ;
- permettre la prise de clichés du système de fichiers ;
- extension dynamique de la taille du volume (sans migration de données).

### 2.1.1 Intégrité, vérification et réparation plus rapides lors d'un crash

Pour permettre une vérification du système de fichiers plus rapide (fsck), les nouvelles structures de données sont moins sensibles à un crash. Un fichier de log permet de connaître les dernières modifications effectuées dans le système de gestion de fichiers, juste avant le crash. Certains systèmes de fichiers sont capables d'effectuer cette vérification pendant le fonctionnement du système, permettant une reprise rapide d'activité, la réparation s'effectuant à chaud.

De plus, chaque bloc de données ou de métadonnées possède un résultat de calcul de *checksum* permettant d'identifier plus facilement les erreurs.

#### 2.1.1.1 Réduction de la fragmentation

Pour réduire la fragmentation des fichiers, plusieurs stratégies se mettent en place. La première consiste à pré-réserver de l'espace sur le disque lors de la création d'un fichier (amélioration de la fonction `fallocate()`). Ainsi, un programme qui est capable de prédire une taille prévisible du fichier qu'il souhaite créer peut demander un espace disque continu plus important. Les données complémentaires ajoutées par la suite seront proches des données initiales. Cela n'est bénéfique que si les applications utilisent cette nouvelle fonctionnalité.

La deuxième approche consiste à retarder la réservation des ressources. La réservation de l'espace sur disque et l'écriture des données sont effectuées plus tard, permettant au système d'avoir une meilleure connaissance du volume de données à traiter, et d'allouer ainsi des ressources contiguës.

D'autre part, les petits fichiers peuvent être directement inclus dans les métadonnées du gestionnaire de fichiers, évitant ainsi la lecture de nouveaux secteurs. Les blocs sont traités par les deux bouts : en tête, les métadonnées ; en queue, les données peu volumineuses.

#### 2.1.1.2 Prise de clichés

Les données étant volatiles sur disque, il est difficile de reconstituer une situation passée. De plus, lors d'un archivage des données, il est préférable d'interrompre tous traitements afin de garantir la consistance des données sauvegardées. Pour résoudre cela, les nouveaux systèmes de fichiers proposent la prise de clichés du système de fichiers. Ainsi, toutes les modifications suivantes n'auront pas d'impact sur le cliché. Il est alors possible d'effectuer un backup, sans interrompre les traitements. Cela peut également être utilisé comme moteur de transaction lors d'une modification. En cas d'échec, le cliché précédent est récupéré.

Pour faire cela, ils proposent la notion de *copy-on-write*. C'est-à-dire que le cliché est un paramètre qui permet, par la suite, la création d'une copie d'un bloc seulement si celui-ci évolue après la prise d'empreinte.

Les clichés sont parfois également accessibles en écriture, permettant une évolution arborescente des données d'un disque. Il est également possible d'avoir un cliché d'un autre cliché. Cela risque de complexifier la gestion des disques.

Les structures de données nécessaires à ces fonctionnalités permettent également d'avoir plusieurs racines dans le même gestionnaire de fichiers. Il s'agit en fait d'un cliché vide depuis la racine. Ainsi, il est plus facile d'isoler des systèmes, sans devoir utiliser plusieurs disques ou plusieurs partitions. Les ressources utilisées sont alors optimums.

Des machines virtuelles peuvent, par exemple, partager le même disque et la même structure, tout en étant isolées l'une de l'autre, ou s'appuyer sur le même OS et avoir des paramètres ou des composants différents dans des clichés spécifiques. Cela contribue à une meilleure exploitation des ressources.

#### 2.1.1.3 Augmentation dynamique des volumes

Les volumes de données traitées par les entreprises sont en inflation constante. Les structures actuelles des gestionnaires de fichiers ne permettent pas l'augmentation des volumes sans une migration des données. Les nouveaux gestionnaires cherchent à permettre cette évolution à chaud, rendant l'ajout de disques possible lors du fonctionnement.

### 2.1.2 Systèmes de gestion de fichiers orientés Cluster

Quatre stratégies permettent le partage de fichiers dans un cluster :

- La première consiste à s'appuyer sur le volume physique pour gérer les verrous. Chaque serveur a un accès direct au disque physique via un GNBD (*Global Network Block Device*).
- La deuxième approche consiste à proposer des serveurs logiques en écoute du réseau pour synchroniser les accès aux disques. Chaque serveur possède un disque avec une partie des répliquas de l'ensemble.
- La troisième approche est un mélange des deux premières. Des serveurs partagent des verrous et tous partagent les mêmes volumes physiques.
- Enfin, la dernière est un serveur de fichiers type NFS, permettant d'exposer un disque distant.

Notez que l'utilisation de ces architectures peut avoir des influences **[4]** sur les bonnes pratiques de développement. Les programmes qui fonctionnent convenablement sur un disque local peuvent présenter des erreurs lors de l'utilisation d'un disque partagé.

### 2.1.3 Normes et standards

Pour le système Linux, les standards de fait consistent à identifier les gestionnaires de fichiers inclus dans la branche principale du noyau.

- ext4 est un standard depuis la version 2.6.28 du noyau et est conforme à POSIX.
- btrfs n'est pas encore un standard, car considéré comme expérimental et est conforme à POSIX.
- ZFS est un produit incompatible avec la licence du noyau Linux. Oracle ayant racheté Sun (à confirmer), il est probable que la licence évolue pour permettre une intégration dans Linux. ZFS est conforme à POSIX.
- OCFS est un produit Oracle, spécialisé sur leurs bases de données. Il n'a pas vocation à être un standard.
- Hadoop FS est une implémentation open source non normalisée et non POSIX.
- GoogleFS est un produit privé, non disponible à ce jour et non POSIX.

## 2.2 Les acteurs des futurs FS

Les différentes évolutions des gestionnaires de fichiers étaient réparties dans différentes entreprises. Depuis peu, avec le rachat de Sun, Oracle possède un quasi-monopole sur les systèmes les plus aboutis. Certaines solutions sont propriétaires, d'autres sont libres. Elles recouvrent parfois les mêmes besoins.

### 2.2.1 Oracle

- OCFS est sous licence GPL.
- ZFS est sous licence CDDL. L'équipe de Sun a été rachetée par Oracle.
- BTRFS d'Oracle est également sous licence GPL et devrait être intégré dans le noyau Linux d'ici deux ans.

Depuis le rachat de Sun par Oracle, la stratégie d'Oracle sur les FS n'est pas lisible. OCFS n'a vocation qu'a répondre aux spécificités des bases de données de l'éditeur.

### 2.2.2 Red Hat

- GFS est sous licence GPL.

Red Hat propose un gestionnaire de fichiers hautement disponible pour enrichir son offre sur le système d'exploitation. GFS est une brique d'un ensemble plus vaste de technologies.

### 2.1.3 Open source

- ext3 et ext4 sont sous licence GPL.
- Hadoop FS de la fondation Apache est sous licence Apache.

Les solutions open source répondent aux exigences nouvelles des applications.

## 2.3 Analyse de l'offre

L'offre se divise en deux catégories, suivant que le gestionnaire de fichier est orienté cluster ou OS.

### 2.3.1 Gestionnaires de fichiers orientés cluster

#### 2.3.1.1 Hadoop FS

Hadoop FS **[5]** est un gestionnaire de fichiers répartis, sous licence Apache, conçu pour fonctionner sur des hardwares à bas coût. Les principes fondateurs sont les suivants :

- Une défaillance hardware est la norme et non l'exception ;
- Accès aux fichiers en streaming (orienté batch et non accès direct) ;
- Ensemble de données très vaste (fichiers de plusieurs terabytes. Supporte dix millions de fichiers par serveur) ;

- Modèle de cohérence simple (modèle *write-once-read-many* par fichier) ;
- « Déplacer le traitement est moins coûteux que déplacer les données ». L'infrastructure permet le déplacement des traitements proches des données en mode GRID ;
- Portabilité.

Les stratégies de réplication des données sont adaptées suivant la localisation des nœuds dans les racks. Les données sont dupliquées d'abord dans le même rack, puis au moins une fois dans un autre rack. Les données sont créées dans un rack identique au rack consommateur de la ressource (affinité d'usage).

Hadoop FS s'appuie sur un autre gestionnaire de fichiers orienté OS pour sauver les données dans chaque serveur, à raison d'un fichier par bloc. Chaque disque possède un fragment des données de l'ensemble.

Ce système n'est pas Posix ni intégré au noyau Linux. Il propose des API spécifiques.

#### 2.3.1.2 OCFS

OCFS **[6]** d'Oracle est un gestionnaire utilisant le disque pour communiquer entre les nœuds d'une grappe. Comme indiqué par le constructeur, ce gestionnaire de fichiers a été conçu pour gérer les fichiers d'une base Oracle, pas pour traiter tous types de fichiers **[7]**. Il s'appuie sur un GNBD (*Global Network Block Device*).

Un mécanisme de verrou global et de verrou local par membre de la grappe est mis en place, par gestion de time-out sur l'évolution de certains secteurs. (La manipulation d'un secteur est considérée comme atomique). Lors du lancement d'un nouveau nœud, ce dernier utilise le verrou global pour obtenir un identifiant de nœud et bénéficier de secteurs privés pour permettre la communication avec les autres nœuds. Un protocole de vote est également disponible, permettant la synchronisation des nœuds.

Cette implémentation présente la particularité de ne pas dépendre du réseau pour gérer les verrous. Par contre, le nombre maximum de nœuds est un paramètre à indiquer lors du formatage du disque. La découverte et la communication des nœuds utilisent l'écriture de secteurs sur disque. Ces derniers sont fortement sollicités et ne doivent pas présenter de défaillances.

OCFS est sous licence GPL.

#### 2.3.1.3 GoogleFS

GoogleFS **[8]** propose une approche similaire à Hadoop HS. Un serveur intermédiaire est en charge de la distribution des données. Pour éviter les contentions lors des mises à jour, le système est spécialisé pour permettre facilement l'ajout d'informations à un fichier par plusieurs processus. L'ordre des ajouts n'est pas garanti, mais chaque ajout est atomique. Pour gérer les défaillances, les données sont dupliquées au minimum sur trois disques.

GoogleFS s'appuie sur un autre gestionnaire de fichiers orienté OS pour sauver un fragment des données dans chaque serveur.

Au niveau architecture, un serveur maintient en mémoire la localisation de tous les blocs (taille de 1 Go) sur tous les disques. Les évolutions sont enregistrées dans un log redondé. Des serveurs de données se chargent alors de sauver les blocs sur leurs disques respectifs et de maintenir les caches mémoires. En cas de défaillance d'un disque, toutes les données perdues sont dupliquées dynamiquement à partir des autres disques vers un nouveau serveur de données.

Des API spécifiques sont nécessaires pour traiter ce gestionnaire de fichiers, permettant l'ajout de données à un fichier et d'obtenir en réponse sa localisation physique par rapport au début du fichier. Ce système n'est pas POSIX ni intégré au noyau Linux.

### 2.3.2 Gestionnaires de fichiers orientés OS

#### 2.3.2.1 ext4

Le système de fichiers ext, utilisé actuellement sous Linux, a été conçu il y a plus de quinze ans, en 1992, puis amélioré en 1999 avec ext2 et en 1999 avec ext3 et l'ajout d'un fichier journal.

ext4 **[9]** a été intégré dans la version 2.6.19 du noyau Linux, à titre expérimental. Il est considéré comme stable depuis la version 2.6.28 du noyau.

La structure de données est compatible avec la précédente version (ext3), il n'est donc pas nécessaire d'effectuer une migration des données pour l'utiliser. Par contre, la migration est préférable si on souhaite bénéficier des nouvelles optimisations.

Certaines limitations d'ext3 sont repoussées comme le nombre de sous-répertoires qui est maintenant illimité.

Un mécanisme d'*extend* permet d'indiquer qu'un fichier débute par un bloc et continue dans les n blocs suivants. Cela permet de réduire la fragmentation. Ce mécanisme est mis en place lors de la création d'un répertoire, accélérant ainsi les accès. La réservation retardée permet également de réduire la défragmentation, en attendant d'avoir un certain volume avant de décider de la localisation des données sur le disque. Attention, cela peut entraîner une perte de données **[10]** si un crash arrive avant le flush des données sur disque. Les programmes qui ne demandent pas de vider le cache avant la fermeture du fichier sont vulnérables (`fsync()` ; `fclose()`).

Un mécanisme de défragmentation on-line sera disponible, permettant aux gros fichiers d'être lus de manière séquentielle.

Pour garantir l'intégrité, un mode « barrière » est mis en place, assurant l'ordre d'écriture sur le disque. En effet, certains disques réorganisent les lectures et les écritures pour éviter de switcher de mode pour chaque secteur. Cela peut présenter des risques lors de l'écriture des fichiers de logs. La barrière garantit l'écriture des données avant tous autres traitements. Cela a évidemment un impact sur les performances et peut être désactivé (barriers=0).

Il n'y aura pas d'ext5, car une nouvelle implémentation est en préparation : btrfs.

#### 2.3.2.2 BTRFS

BTRFS **[11]** est une refonte complète du système de fichiers, suite **[12]** aux travaux d'Ohad Rodeh sur les arbres binaires incrémentaux **[13]**.

Les nouvelles fonctionnalités sont nombreuses :

- Étend la taille maximum des fichiers à 2^64 ;
- Améliore la gestion de petits fichiers. Les meta-informations sont agrégées en tête de bloc, les petits fichiers en queue, jusqu'à épuisement ;
- Améliore l'indexation des répertoires ;
- Allocation dynamique des inodes, permettant l'augmentation dynamique des volumes ;
- Création d'instantanés, modifiables en écriture ;
- Création de sous-volumes (instantané vide de la racine) ;
- Checksum des données et des méta-datas, facilitant la détection d'erreurs ;
- Compression à la volée, au niveau du volume entier ;
- Intégration de multiples devices, avec plusieurs algorithmes type RAID ;
- Vérification en ligne de l'intégrité du système de fichiers ;
- Vérification très rapide du système off-line (utilisation des logs) ;
- Back-up incrémental et miroir du FS rapide ;
- Défragmentation en ligne.

Par nature, le copy-on-write défait toute stratégie d'effacement sécurisé puisque les blocs ne sont pas écrasés mais réécrits à côté. Cela peut poser des problèmes de confidentialité. Il n'y a d'ailleurs pas d'option pour chiffrer les blocs (encryptfs peut être une option).

Il est prévisible que BTRFS soit le prochain système de fichiers Linux par défaut d'ici deux ans.

Les benchs publiés présentent des résultats **[14]** contradictoires **[15]**

#### 2.3.2.3 zfs

ZFS **[16]** est un gestionnaire de fichiers qui mime un gestionnaire de mémoire. Il propose des capacités gigantesques (128 bits), à l'aide de disques mis en commun de tous types (physique, NAS, SAN, baies de disques, etc.). Le mécanisme de transaction permet d'éviter l'utilisation de journal d'entrée/sortie, sans resynchronisation en cas de panne de courant. Les blocs ont des tailles variables, choisies en fonction de la donnée écrite.

Utilisant également le copy-on-write, ZFS défait toute stratégie d'effacement sécurisé. Le chiffrement à la volée est en cours de réalisation, ainsi que la défragmentation on-line.

La licence CDDL **[17]** est incompatible avec la licence Linux, compromettant tous portages. Le rachat de Sun par Oracle fera peut-être évoluer la situation. Des portages **[18]** Linux sont expérimentés.

### 2.3.3 Tendance générale du domaine

Les nouveaux systèmes de fichiers cherchent à proposer le maximum de fonctionnalités pour répondre aux nouveaux usages. Les gestionnaires orientés Cluster proposent de gérer de très gros volumes en streaming (type vidéo) et en optimisant les accès concurrents. Les données sont dupliquées plusieurs fois sur des disques considérés comme défaillants par nature.

Les gestionnaires orientés OS proposent une gestion d'instantanés plus ou moins évolués, et cherchent à améliorer l'intégrité des données et la reprise sur incident. Les volumes traités sont très importants et doivent pouvoir s'amplifier sans interruption de service. ■

**Auteur : Philippe PRADOS**

### Liens

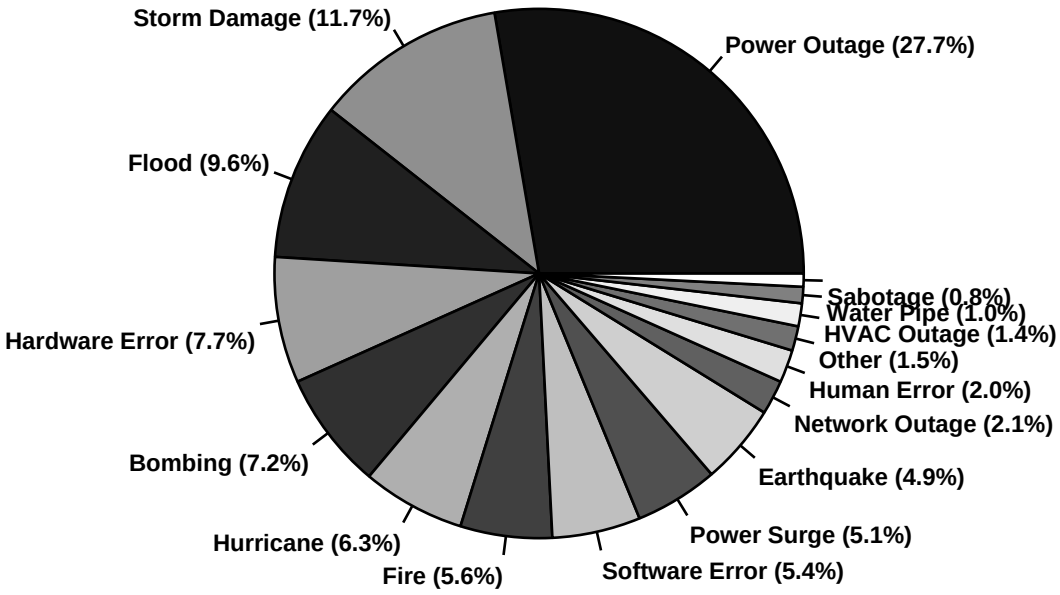

- **[1]** http://www.alfresco.com
- **[2]** http://fr.wikipedia.org/wiki/IEEE
- **[3]** http://tinyurl.com/ylnr3kd
- **[4]** http://tinyurl.com/d3uds
- **[5]** http://tinyurl.com/ycqyrjm
- **[6]** http://oss.oracle.com/projects/ocfs/
- **[7]** http://tinyurl.com/yleuyjm
- **[8]** http://labs.google.com/papers/gfs.html
- **[9]** http://tinyurl.com/2rz37u
- **[10]** http://tinyurl.com/d9pcfg
- **[11]** http://tinyurl.com/5n5xjg
- **[12]** http://tinyurl.com/ygpxd3z
- **[13]** http://tinyurl.com/yd5w3mx
- **[14]** http://www.linux-mag.com/id/7308/3/
- **[15]** http://tinyurl.com/cwtw2s
- **[16]** http://opensolaris.org/os/community/zfs/
- **[17]** http://fr.wikipedia.org/wiki/CDDL
- **[18]** http://tinyurl.com/ykhzspv

Chapitre 11

# *Mécanismes de sauvegarde*








*There are two types of computer users in the world. . .*
*those that have lost data, and those that are going to.*

Quand un fichier est détruit, il n'y aucun moyen de le récupérer sous UNIX.
⇒ Seule prévention : faire des sauvegardes

Les sauvegardes sont donc **indispensables**.

UNIX offre une vue unifiée des fichiers ou des périphériques. On peut donc sauvegarder :

- sur n'importe quel périphérique en mode caractère
- dans un fichier
- sur un autre disque
- sur une autre machine





Beaucoup de discipline et de rigueur.
Pseudo plan :

- Que faut-il sauvegarder ?
- Avec quelle fréquence ?
- Combien de temps conservera-t-on les sauvegardes, à quels endroits, en combien d'exemplaires ?
- A quel endroit sera stocké l'historique des sauvegardes ?
- Quel est le support le plus approprié ?
- Quels sont les besoins, en capacité, de support de sauvegarde ?
- Combien de temps, au plus, doit durer la sauvegarde ?
- Combien de temps prévoit-on pour restaurer un ficher, un système de fichiers, est-ce raisonnable ?
- La sauvegarde doit-elle être automatique ou manuelle ?
- Quelle est la méthode de sauvegarde la plus appropriée ?

Il y a plusieurs technologies de lecteur de bandes magnétiques :

- bande 9 pouces
- QIC
- Exabyte
- DDS
- DLT
- Super DLT (alias SDLT)
- AIT
- Super AIT
- Ultrium LTO

Et d'autres viendront...





## ◇ Technologie bande 9 pouces






## ◇ Technologie QIC

(en anglais *Quarter Inch Cartridge* car largeur de la bande 0.25 pouce)




Date de 1972
Cartouche QIC60 = 60 MB
Cartouche QIC150 = 150 MB
Déclinaison par les firmes TRAVAN et TANDBERG

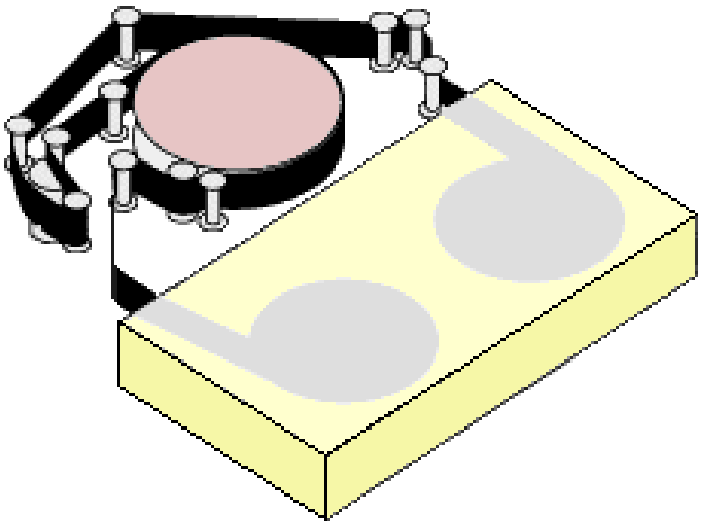




## ◇ Technologie EXABYTE

Bande de 8mm
Technologie adaptée du monde de la vidéo
Une seule firme fabricant cette technologie : `http://www.exabyte.com`

Chemin de la bande compliquée dans les lecteurs :



| Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---:|---:|---:|
| 3.5 GB | 7 GB | 32 MB/min |
| 5 GB | 10 GB | 60 MB/min |
| 7 GB | 14 GB | 60 MB/min |
| 7 GB | 14 GB | 120 MB/min |



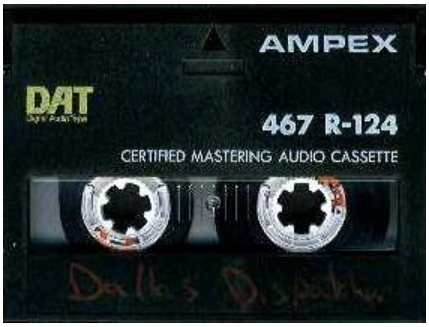





## ◇ Technologie DDS

(en anglais *Digital Data Storage*)
`http://www.datmgm.com`

Bande de 4mm
Technologie adaptée du monde de l'audio DAT (Digital Audio Tape)



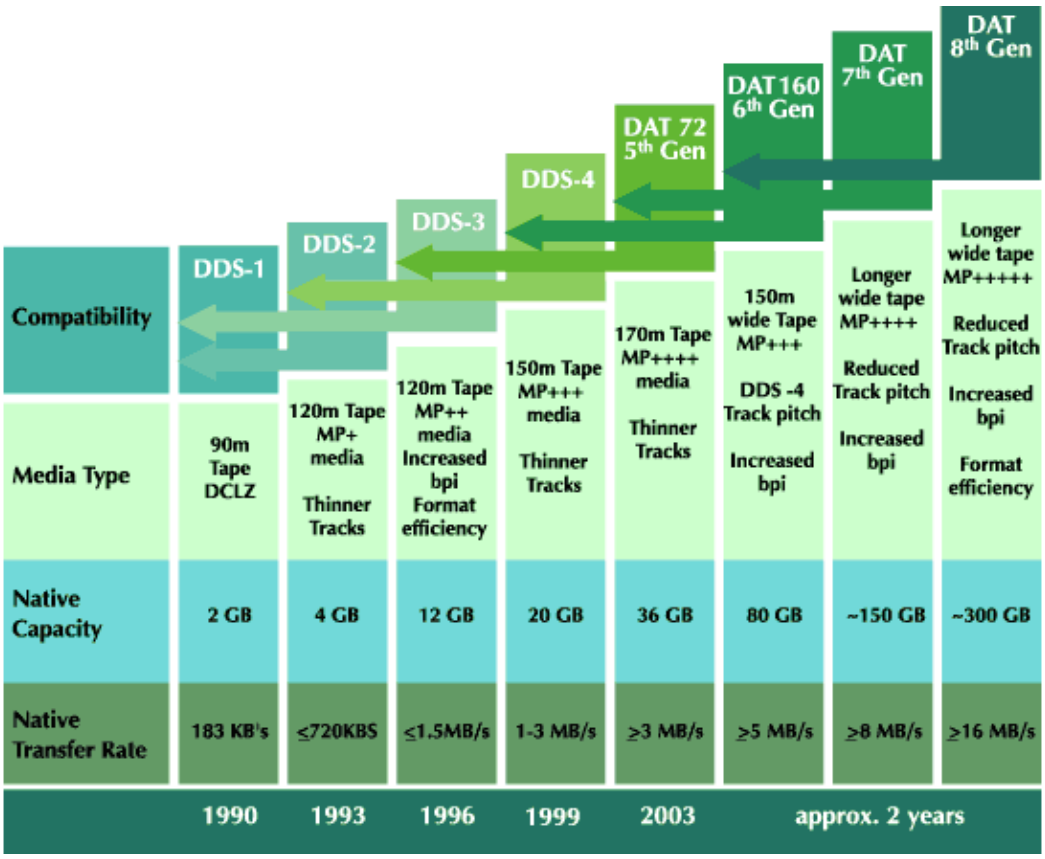

1998 → DDS (en anglais *Digital Data Storage*)

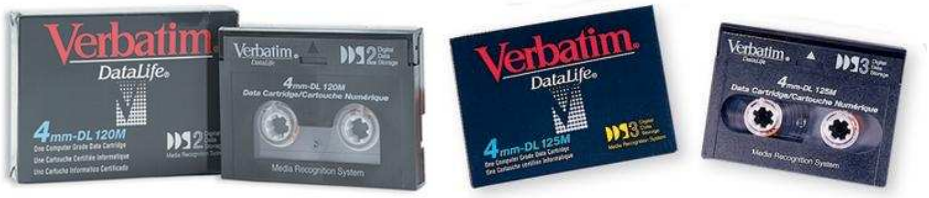
| Norme | Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---|---|---|---|
| DDS | 2 GB | | 55 KB/s |
| DDS-1 | 2 GB | 4 GB | 0.5 / 1 MB/s |
| DDS-2 | 4 GB | 8 GB | 0.5 / 1 MB/s |
| DDS-3 | 12 GB | 24 GB | 1 / 2 MB/s |
| DDS-4 | 20 GB | 40 GB | 3 / 6 MB/s |
| DDS-5 | 36 GB | 72 GB | 3 / 6 MB/s |

**Un lecteur de bande DDS meurt en 3 ans (vérifié par 5 collègues).**







| | |
|---|---|
|  DDS | DDS-1 |
|  DDS-2 | DDS-3 |
|  DDS-4 | DDS-5 |

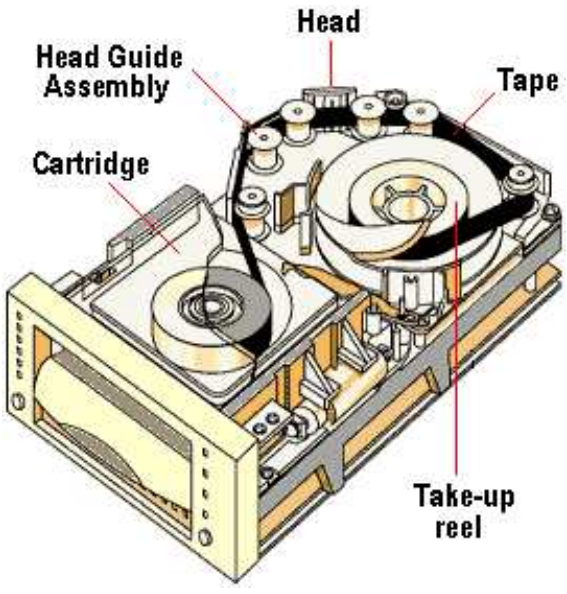


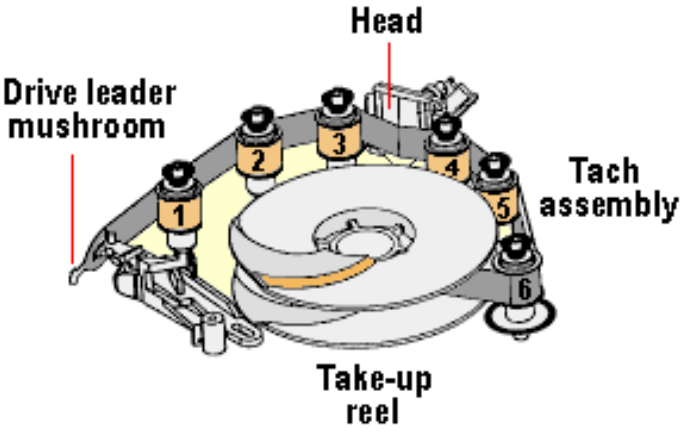


## ◇ Technologie DLT

(en anglais *Digital Linear Tape*)

`http://www.dlttape.com`



| Norme | Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---|---|---|---|
| DLT2000 | 15 GB | 30 GB | 2.5 MB/s |
| DLT4000 | 20 GB | 40 GB | 3 MB/s |
| DLT7000 | 35 GB | 70 GB | ? MB/s |
| DLT8000 | 40 GB | 80 GB | 6 / 12 MB/s |
| DLT VS80 | 40 GB | 80 GB | 3 / 6 MB/s |
| DLT VS160 | 80 GB | 160 GB | 8 / 16 MB/s |








?




DLT




DLT4000

## ◇ Technologie SDLT

(en anglais *Super Digital Linear Tape*)

`http://www.sdlt.com`

| Norme | Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---|---|---|---|
| SDLT 220 | 110 GB | 220 GB | 11 / 22 MB/s |
| SDLT 320 | 160 GB | 320 GB | 16 / 32 MB/s |








## ◇ Technologie AIT

(en anglais *Advanced Intelligent Tape*)

| Norme | Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---|---|---|---|
| AIT-1 | 35 GB | 90 GB | 4 / 10.4 MB/s |
| AIT-2 | 50 GB | 130 GB | 6 / 15.6 MB/s |
| AIT-3 | 100 GB | 260 GB | 12 / 31 MB/s |
| S-AIT | 500 GB | 1300 GB | 30 / 60 MB/s |



## ◇ Technologie LTO

http://www.lto-technology.com

| Norme | Capacité native | Capacité compressée | Vitesse (Mo/s) |
|---|---|---|---|
| Ultrium LTO 1/2 | 100 GB | 200 GB | 7.5 / 15 MB/s |
| Ultrium LTO | 100 GB | 200 GB | 15 / 30 MB/s |
| Ultrium LTO 2 | 200 GB | 400 GB | 30 / 60 MB/s |




Ultrium 200



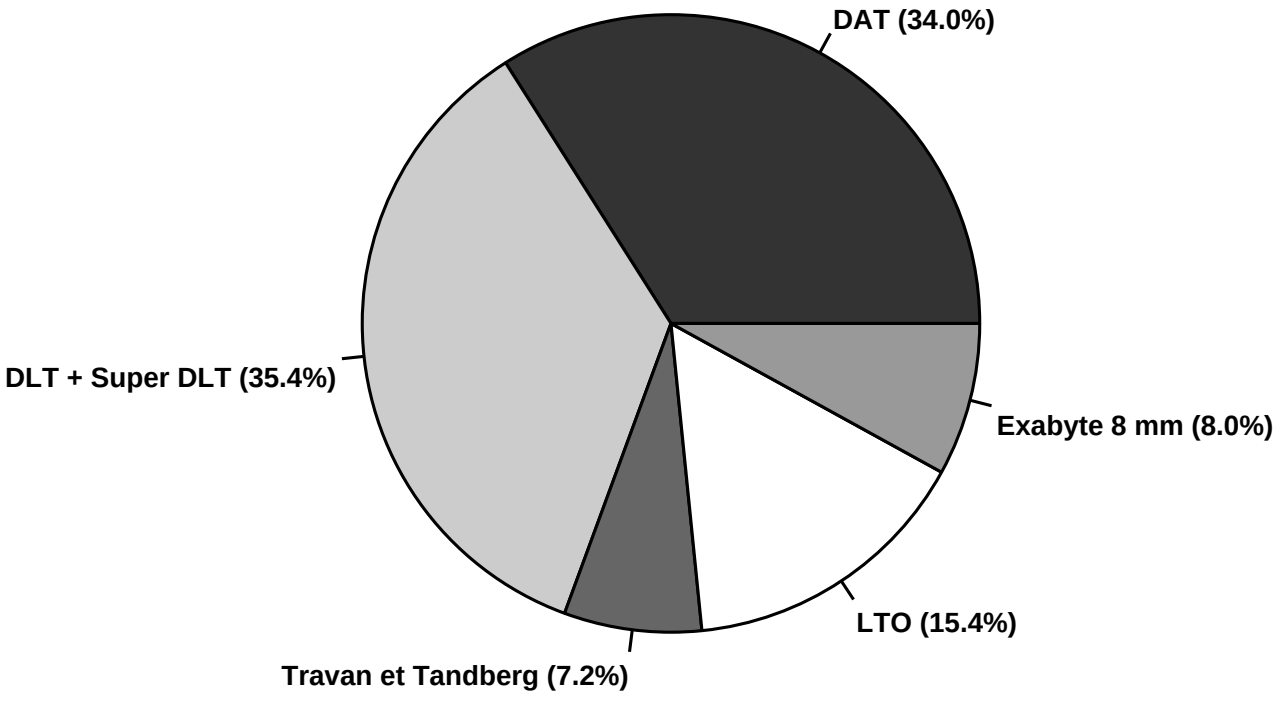

Ultrium 200




Ultrium 400







Le lecteur peut être isolé (interne ou externe)







Le lecteur peut être monté dans une library ≡ ensemble lecteur + robot de manipulation de cartouches stockées dans un caroussel




Le lecteur peut être plus gros ! voir les pages suivantes.


IBM 3494
Automatic
Tape Library



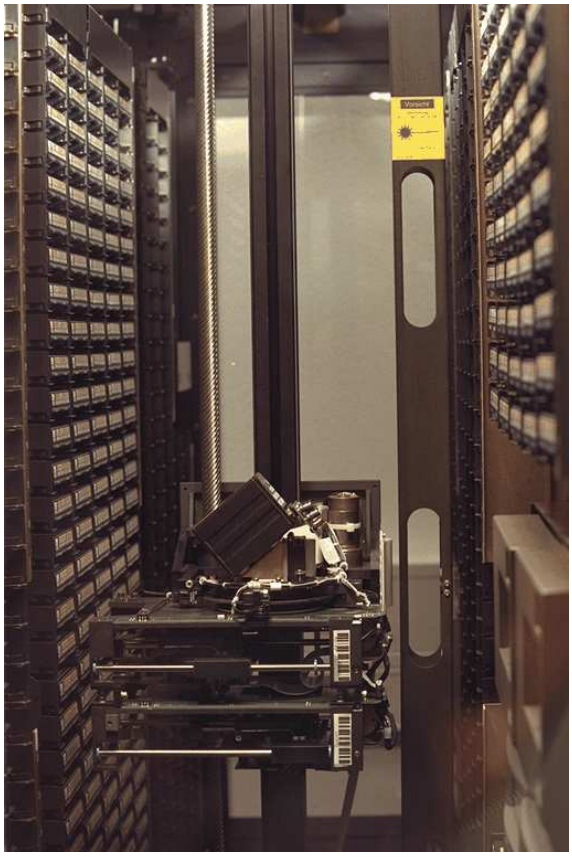



ATTENTION















Il y a plusieurs déclinaisons possibles :

- compression hardware activée ou pas
- rembobinage en fin d'écriture automatique ou non

Les périphériques par défaut :

- Sur LINUX : « `/dev/st0` » (en anglais *scsi tape*)
- Sur SOLARIS : « `/dev/rmt/0` » (en anglais *raw magnetic tape*)
- Sur DIGITAL UNIX : « `/dev/tape/0` »
- Sur FreeBSD : « `/dev/st0` » (en anglais *scsi tape*)

A noter que les lecteurs de bande magnétique sont en SCSI.

On a en général plusieurs fichiers device par lecteur car :

- plusieurs densités possibles
  Exemple de SOLARIS avec l (*low*), m (*medium*), h (*high*), u (*ultra*), c (*compressed*)
- rembobinage ou pas
  Exemple de SOLARIS avec n (*no rewind*)

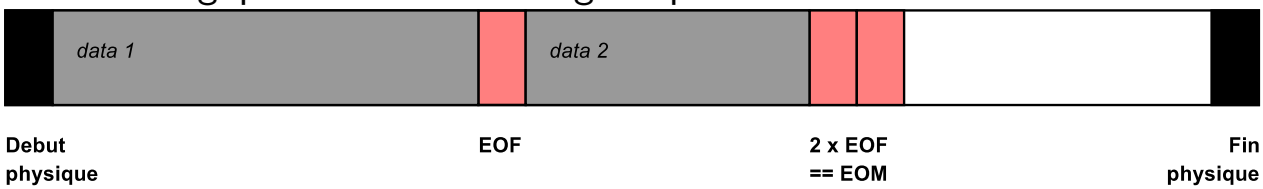
Exemple (SOLARIS) :

- « `/dev/rmt/0` » et « `/dev/rmt/0n` »
- « `/dev/rmt/0l` » et « `/dev/rmt/0ln` » : low density
- « `/dev/rmt/0m` » et « `/dev/rmt/0mn` » : medium density
- « `/dev/rmt/0h` » et « `/dev/rmt/0hn` » : high density
- « `/dev/rmt/0u` » et « `/dev/rmt/0un` » : ultra density
- « `/dev/rmt/0c` » et « `/dev/rmt/0cn` » : compression hardware





Structure logique d'une bande magnétique :




Commandes de manipulation de la bande :

- rembobiner : « `mt -f /dev/rmt/0n rewind` »
- avant d'un registrement : « `mt -f /dev/rmt/0n fsf 1` »
- reculer d'un registrement : « `mt -f /dev/rmt/0n bsf 1` »
- info sur la bande : « `mt -f /dev/rmt/0n status` »
- rembobiner et ejecter la cartouche :
  « `mt -f /dev/rmt/0n offline` »

Exemple :

```
# mt status
Sun DLT7000 tape drive:
   sense key(0x0)= No Additional Sense   residual= 0   retries= 0
   file no= 0   block no= 0
```

La particularité de **dump** est d'assurer la **sauvegarde d'une partition** en consultant la structure interne même du filesystem via le fichier spécial en mode caractère de la partition.

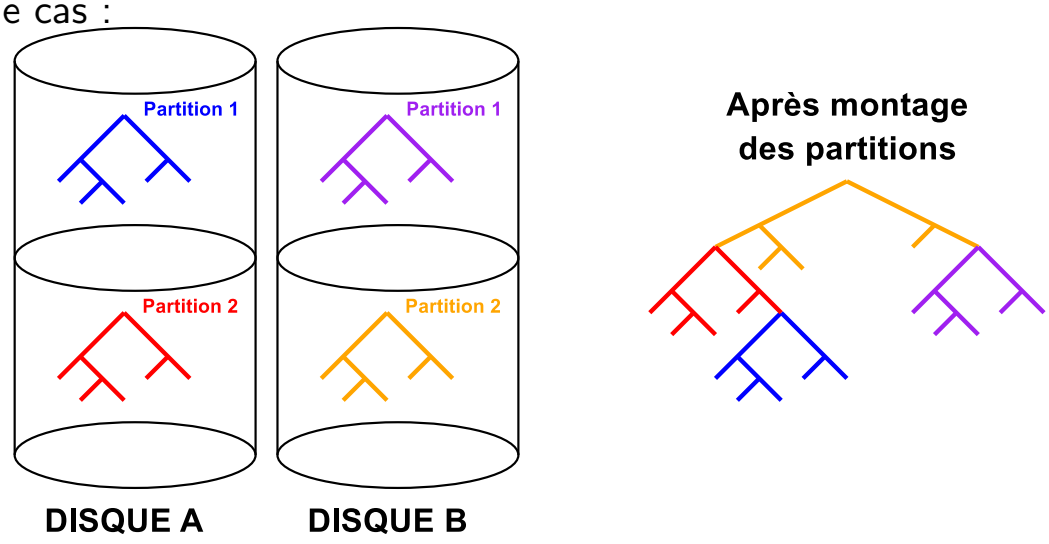
Syntaxe conseillée :
`dump 0f bande-magnétique partition`
(sur Solaris, utiliser `/sbin/ufsdump`)

Possibilité de sauver dans un fichier.
Syntaxe analogue :
`dump 0f fichier-de-sauvegarde partition`





La commande `dump` est faite pour sauvegarder une partition.
Dans le cas :




alors lancer 4 commandes de sauvegarde pour sauvegarder l'arborescence complète :

```
# dump 0f sauvegarde1.dump partition-B2-orange
# dump 0f sauvegarde2.dump partition-A2-rouge
# dump 0f sauvegarde3.dump partition-A1-bleue
# dump 0f sauvegarde4.dump partition-B1-violette
```

Points forts de « dump » :

- Possibilité de sauvegardes incrémentales. Un niveau est associé à la sauvegarde. Le niveau 0 correspond à une sauvegarde complète. Le niveau 1 correspond à la sauvegarde des seuls fichiers modifiés depuis le niveau 0. etc.
- Sauvegarde des fichiers spéciaux en tant que fichiers spéciaux.
- Sauvegarde des fichiers à trous à l'identique (sauvegarde des seuls blocs utilisés)





Un dump se compose de 4 étapes :

```
# dump 0f /tmp/archive /etc
  DUMP: Date of this level 0 dump: Sat Jan 27 16:48:52 2001
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/hda2 (/ (dir etc)) to /tmp/archive
  DUMP: Label: none
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 3514 tape blocks on 0.09 tape(s).
  DUMP: Volume 1 started at: Sat Jan 27 16:48:52 2001
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing /tmp/archive
  DUMP: Volume 1 completed at: Sat Jan 27 16:48:54 2001
  DUMP: Volume 1 took 0:00:02
  DUMP: Volume 1 transfer rate: 2535 KB/s
  DUMP: DUMP: 5071 tape blocks on 1 volumes(s)
  DUMP: finished in less than a second
  DUMP: DUMP: Date of this level 0 dump: Sat Jan 27 16:48:52 2001
  DUMP: DUMP: Date this dump completed:  Sat Jan 27 16:48:54 2001
  DUMP: DUMP: Average transfer rate: 2535 KB/s
  DUMP: DUMP IS DONE
```

La commande pour relire une sauvegarde réalisée par dump est **restore** (sur Solaris, utiliser `/sbin/ufsrestore`).

Possibilité de restoration totale automatique :

```
cd partition-vide
restore rf archive
```

On peut utiliser dump + restore pour dupliquer une arborescence. Par exemple :

```
dump 0f - /usr/local | ( cd /mnt ; restore rf - )
```





Possibilité de restoration en mode interactif avec sélection de ce que l'on veut restorer : « `restore -f archive -i` » puis utiliser les commandes « `add fichier/dir` », « `cd dir` », « `delete fichier/dir` », « `ls` » et la commande « `extract` » :

```
# restore -f /tmp/archive -i
...
restore > ls
.:
etc/

restore > cd etc
restore > ls
./etc:
...
im_palette-small.pal  passwd.OLD            yp.conf

restore > add yp.conf
restore > ls
./etc:
...
im_palette-small.pal   passwd.OLD            *yp.conf
```

```
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] y
restore > quit
# ls -Rl
.:
total 8
drwxr-xr-x    2 root     root         4096 Jan 27 14:47 etc

./etc:
total 4
-rw-r--r--    1 root     root          361 Jan 26 14:51 yp.conf
```





La commande pour connaitre la table des matières d'une sauvegarde réalisée par dump est aussi « **restore** » (sur Solaris, utiliser « /sbin/ufsrestore »).

```
# ufsdump 0f /tmp/archive /opt3
  DUMP: Date of this level 0 dump: Sat Jan 24 22:27:28 2004
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t0d0s6 (linux.example.com:/opt3) to /tmp/arch
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Writing 32 Kilobyte records
  DUMP: Estimated 350 blocks (175KB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 318 blocks (159KB) on 1 volume at 2650 KB/sec
  DUMP: DUMP IS DONE

# ufsrestore tf /tmp/archive
          2      .
          3      ./lost+found
          4      ./exemple.txt
```

**tar** utilitaire assure une **sauvegarde d'arborescence** en accèdant normalement au filesystem UNIX.

Cette commande a un immense avantage : elle est disponible sur la totalité des plateformes UNIX.

Contrairement à « `dump` », elle permet la sauvegarde d'une partie seulement d'une partition. En fait, on peut sauvegarder des fichiers ou directories sans aucun rapport entre eux.

Syntaxe conseillée :

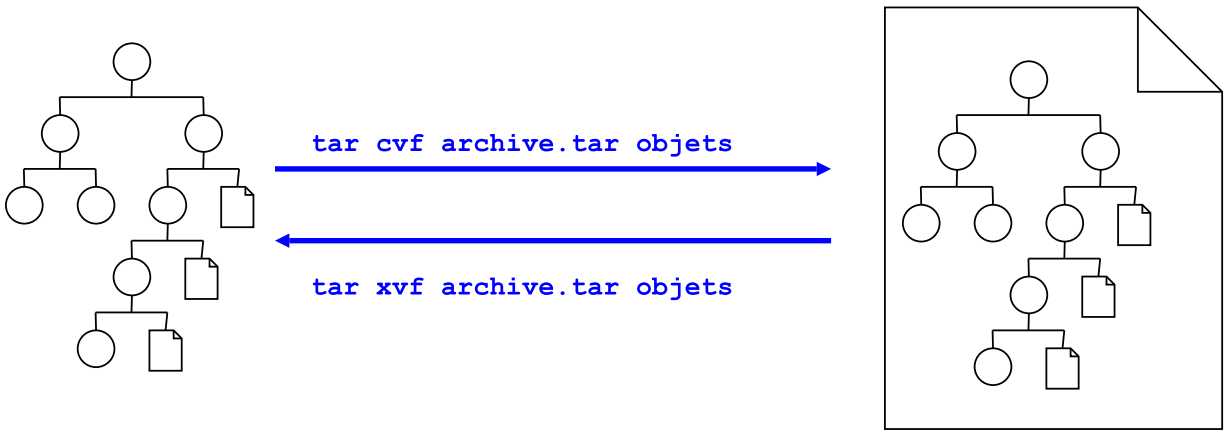
```
tar -cvf bande-magnétique fichiers-ou-répertoires
```

Possibilité de sauver dans un fichier. Syntaxe analogue :

```
tar -cvf fichier-de-sauvegarde fichiers-ou-répertoires
```







Deux notations possibles des options :

1. sans tirets
   Par exemple :

```
# tar cvf /dev/rmt/0 /
```

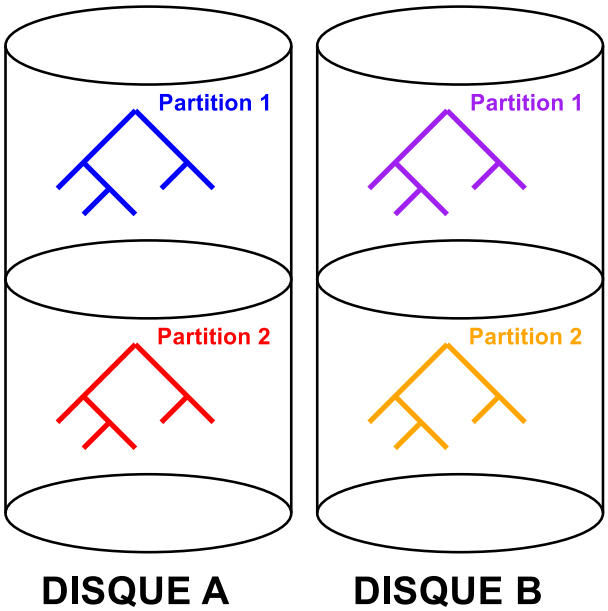
2. avec tirets
   Par exemple :

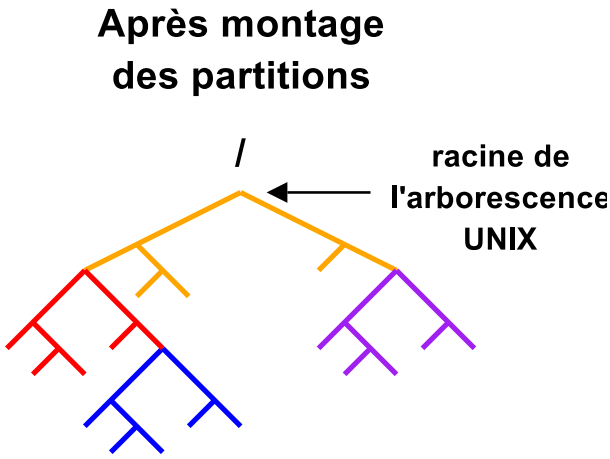
```
# tar -cvf /dev/rmt/0 /
# tar -c -v -f /dev/rmt/0 /
```





La commande « `tar` » est faite pour sauvegarder une arborescence.
Dans le cas :




alors lancer **une** commande de sauvegarde pour sauvegarder l'arborescence complète :

```
# tar -cvf /dev/rmt/0 /
```

De préférence, utiliser le GNU tar :
`ftp://ftp.lip6.fr/pub/gnu/tar/tar-1.14.tar.gz`
associé au GNU zip :
`ftp://ftp.lip6.fr/pub/gnu/gzip/gzip-1.2.4a.tar`

Avantages :

- le GNU tar ne sauve pas les fichiers avec un '/' initial ;
- il offre la possibilité de se cantonner au sein d'un filesystem.

Souvent appelé « `gtar` ».

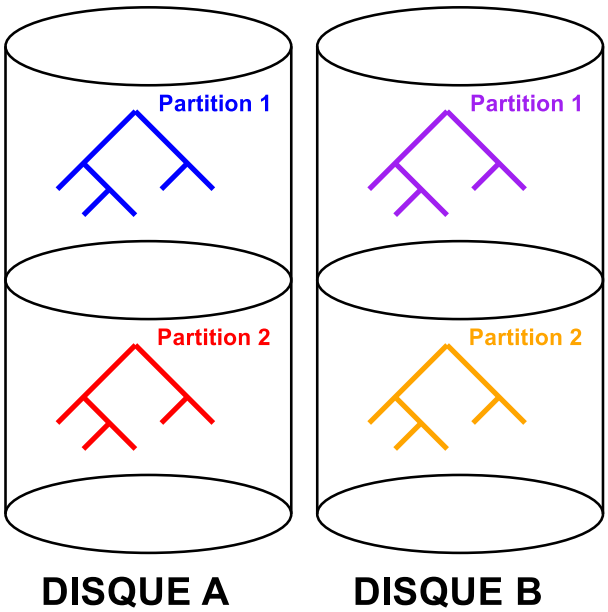
Options longues disponibles :

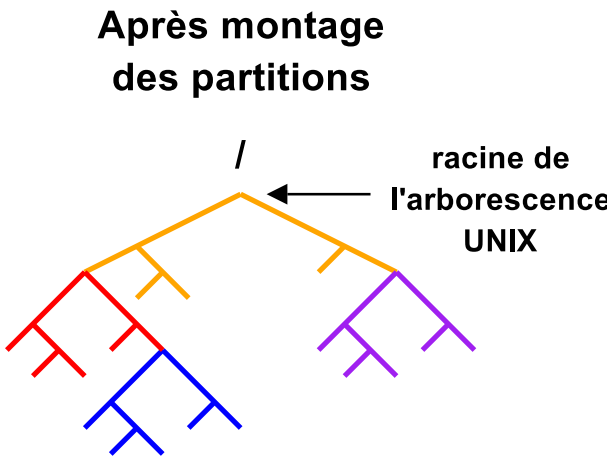
- option « `--create` » équivalente à « `-c` »
- option « `--extract` » équivalente à « `-x` »
- option « `--list` » équivalente à « `-t` »
- option « `--verbose` » équivalente à « `-v` »
- option « `--file=backup.tar` » équivalente à « `-f backup.tar` »





Le GNU tar peut contrôler si l'on doit rester au sein d'une partition.
Pour rester au sein de la partition orange B2 (montée dans `/`) :




alors lancer la commande :

```
# gtar --create --verbose --file /dev/rmt/0 --one-file-system /
```

Problèmes classiques avec tar :

- Il ne sait pas sauver convenablement les major et minor numbers des fichiers de « /dev ». Pour sauver « /dev », on préférera plutôt recréer les devices à partir de « /dev/MAKEDEV ».
- Par défaut, il ne sait pas sauver un fichier à trous (les blocs intermédiaires non alloués sont sauvés sous forme de caractère de code ASCII zéro).
  ⇒ effacer par exemple les fichiers core avant de sauvegarder par cron, sinon remplissage inutile de la bande





Exemple de sauvegarde par tar (option « c ») :

```
# tar -cvf archive.tar /etc
tar: Removing leading `/' from member names
etc/
etc/csh.cshrc
etc/csh.login
etc/exports
etc/filesystems
etc/group
etc/host.conf
etc/hosts.allow
etc/hosts.deny
...
```

## Exemple de consultation de la table de contenu de l'archive (option « t ») :

```
# tar -tvf archive.tar
drwxr-xr-x root/root         0 2001-01-27 14:47:46 etc/
-rw-r--r-- root/root       220 2000-01-13 00:18:52 etc/csh.cshrc
-rw-r--r-- root/root       674 2000-01-13 17:24:18 etc/csh.login
-rw-r--r-- root/root         0 2000-01-13 00:18:52 etc/exports
-rw-r--r-- root/root        43 2000-02-17 22:42:07 etc/filesystems
-rw-r--r-- root/root       419 2001-01-26 14:51:13 etc/group
-rw-r--r-- root/root        26 2000-01-13 00:18:52 etc/host.conf
-rw-r--r-- root/root       161 2000-01-13 00:18:52 etc/hosts.allow
-rw-r--r-- root/root       347 2000-01-13 00:18:52 etc/hosts.deny
...
drwxr-xr-x root/root         0 2001-01-26 14:48:49 etc/profile.d/
...
```





## Extraction d'un fichier (option « x ») :

```
# ls -l etc/passwd
ls: etc/passwd: No such file or directory
# tar -xvf archive.tar etc/passwd
etc/passwd
# ls -l etc/passwd
-rw-r--r--   1 root   root      597 Jan 27 14:47 etc/passwd
```

Sur Internet, il y a souvent des packages avec « .tar.gz » ou « .tgz » comme suffixes.
Comment en extraire le contenu ?

```
% gunzip archive.tar.gz
% tar -xvf archive.tar
..
```

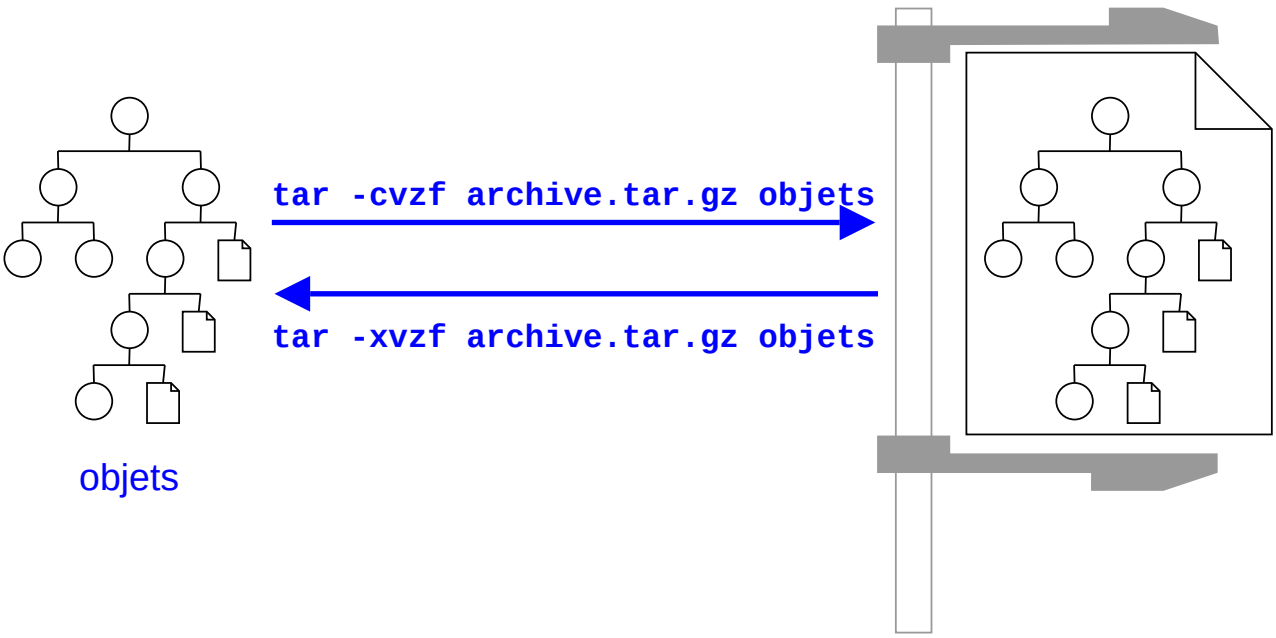
Cela nécessite le GNU zip.

Si l'on a le GNU tar, on peut combiner les 2 opérations en une seule :

```
% gtar -xvzf archive.tar.gz
...
```







Exemple d'emploi de tar pour dupliquer une arborescence :

```
% ( cd /usr/local ; tar -cf - . ) | ( cd /mnt ; tar -xf - )
```





**A completer...**

http://www.amanda.org

# A completer...

Options : voir article 814583 de la Knowledge Base

A signaler : « http ://www.ntbackup.info »





A signaler dans le Resource Kit Windows : « mtfcheck.exe »

MTF = Microsoft Tape Format

MTFCHECK :

- est capable de vérifier une bande écrite par NTBACKUP.
- est capable de restorer des fichiers à partir une bande écrite par NTBACKUP.
- est capable d'effacer une bande (écriture de codes ASCII zéro).

Exemple : « mtfcheck.exe -ttape0 -rC:\temp\report.txt »

Autre univers :

- Legato networker
- Time Navigator
- Arkeia
- etc.





**A completer...**

`http://www.arkeia.com`

**A completer...**

http://www.bacula.org





**A completer...**

http://www.ibm.com/software/tivoli/

1. Sauvegarder
2. Sauvegarder lorsque le filesystem est le moins actif :
   - en mode mono-utilisateur
   - pendant les périodes creuses d'utilisation (la nuit par exemple)
3. Ne jamais restaurer de fichiers dans le homedir d'un utilisateur de peur d'y effacer des fichiers plus récents que ceux de la sauvegarde.
4. Vérifiez périodiquement les sauvegardes.
5. Conserver un jeu de sauvegardes ok à part des autres en cas de destruction des autres par un sinistre.

# Les catastrophes existent !





# Les catastrophes existent !

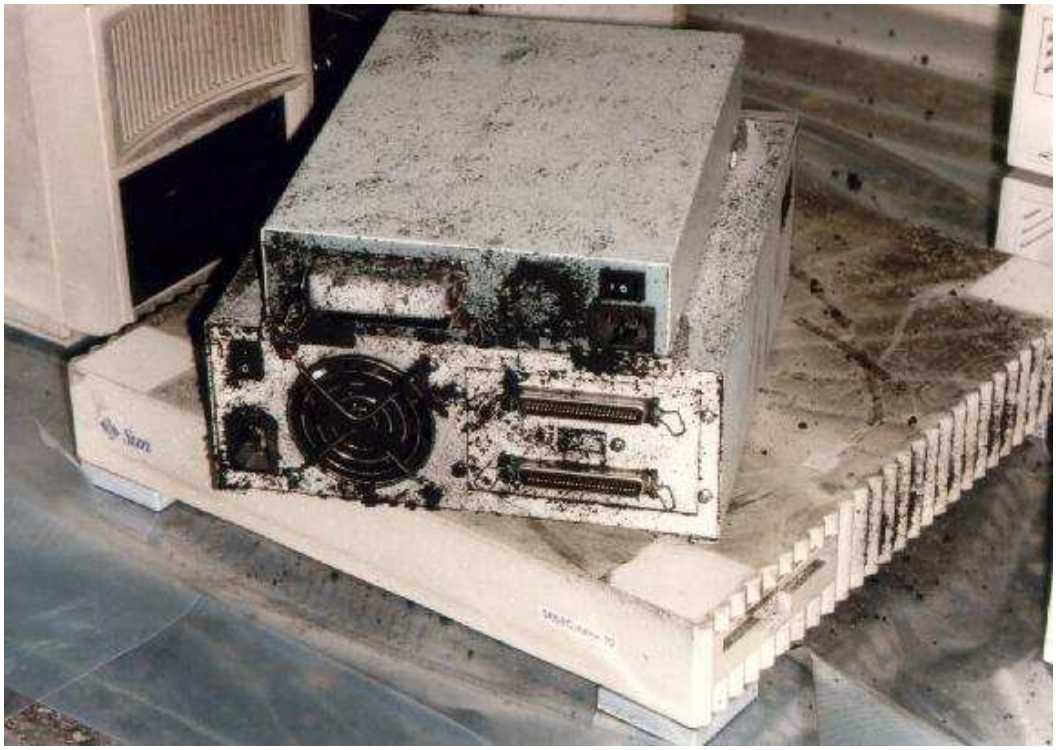
Quelques expériences personnelles. . .



Matériels encrassés suite à un incendie (1/4) – 1997







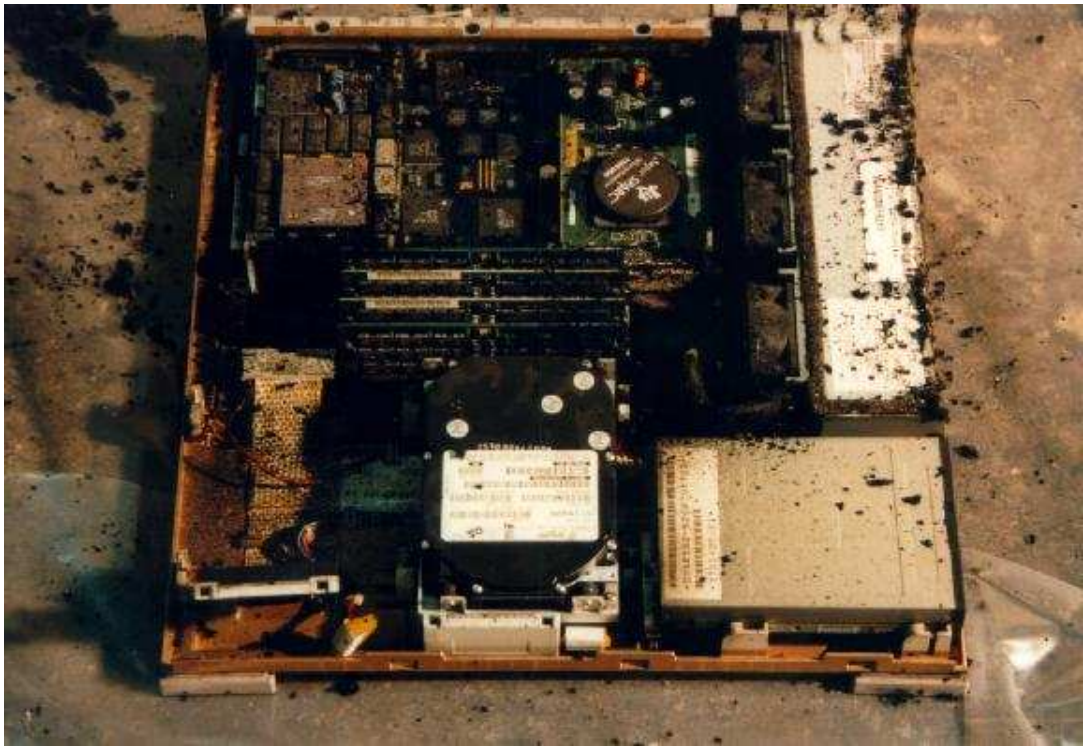
Matériels encrassés suite à un incendie (2/4) – 1997



Matériels encrassés suite à un incendie (3/4) – 1997







Matériels encrassés suite à un incendie (4/4) – 1997



Faux plafond effondré suite à une inondation – 1997







Locaux inondés suite à la tempête de décembre 1999.




Matériel cassé suite à une crise de folie d'un employé (1/2) – été 2004







Matériel cassé suite à une crise de folie d'un employé (2/2) – été 2004



Fibres optiques arrachées par une pelleteuse à Jussieu (1/2) – Octobre 2005







Fibres optiques arrachées par une pelleteuse à Jussieu (2/2) – Octobre 2005

RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'INTÉRIEUR, DE LA SÉCURITÉ INTÉRIEURE ET DES LIBERTÉS LOCALES
DIRECTION GÉNÉRALE DE LA POLICE NATIONALE

PREFPOL/DPUP
SVP / SPQ 13ème arrdt
Tél : 01-40-79-05-05

Code INSEE du service : 071 551/113

VICTIME/DECLARANT

# COMPTE RENDU D'INFRACTION INITIAL

VICTIME : UNIVERSITE PARIS 7 - BIBLIOTHEQUE DE MATHEMATIQUE, société FRANCAISE
établi(e) 16, RUE CLISSON 75013 PARIS
Téléphone : 01.44.27.37.23

PREJUDICE : Butin : A EVALUER Dégâts : AUCUN

INFRACTION : **VOL SANS EFFRACTION**
Entre le 28/07/2006 à 19 H 00 et le 31/07/2006 à 08 H 00 (INDT), PERIODE CONGES SCOLAIRES.
16, RUE CLISSON à PARIS (13 arrdt).
Nature du lieu : LIEU PRIVE
Eléments d'enquête :
- Personnes remarquées : néant
- Mode opératoire : VOL SANS EFFRACTION DE DEUX ORDINATEURS.

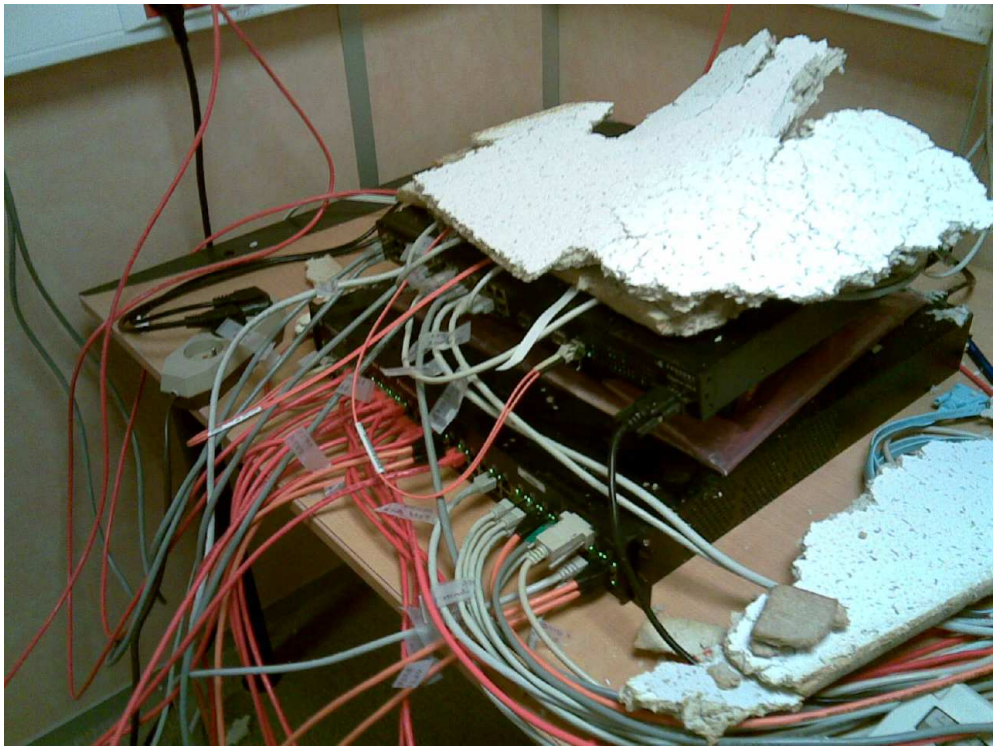
P.V. : 2006/ /001
Affaire contre X
Pièces jointes :

## PROCES VERBAL

Vol – Juillet 2006







Faux plafond effondré sur un routeur et un switch gigabit suite à une fuite d'eau sur une climatisation – Août 2007

# On entre à la fac comme dans un moulin

LA TOUTE NOUVELLE faculté des Grands-Moulins porte bien son nom. Car on y entre… comme dans un moulin. Pas de vigile visible, pas d'accès sécurisé, pas de caméras. Une trentaine d'ordinateurs — dont les derniers la semaine dernière dans le bâtiment Buffon — y ont déjà été dérobés. A deux reprises, les vols ont été assortis de violence, le malfrat menaçant une employée à l'aide d'une pince coupe-boulon…

« La priorité était de permettre aux étudiants de débuter leur année universitaire dans leurs nouveaux locaux, explique François Montarras, le vice-président de l'université. A la fin de l'année, tous les accès seront sécurisés et il y aura une vingtaine de caméras par bâtiment. Il n'y aura plus aucun vol. »

## Des sorties de cours sur la rue

Etudiante en informatique, Jasmine, 22 ans, a vu « des écrans d'ordinateur rester par terre dans un couloir pendant des mois ». Ses amis assurent cependant que « les conditions de sécurité sont bien meilleures qu'à Jussieu. On pouvait rester là-bas toute la nuit sans que personne ne s'en aperçoive… ».

Un autre dysfonctionnement en matière de sécurité a été découvert dans la superbe bibliothèque de la faculté, accessible à tous grâce à une simple inscription. A l'origine, les sorties de secours (non surveillées) donnaient sur la rue. N'importe qui pouvait donc partir avec des livres sous le bras. Le rectorat et l'université y ont remédié.

RAPHAËL DOMENACH

Article de journal concernant l'architecture du nouveau campus de l'Université Paris VII – Janvier 2008







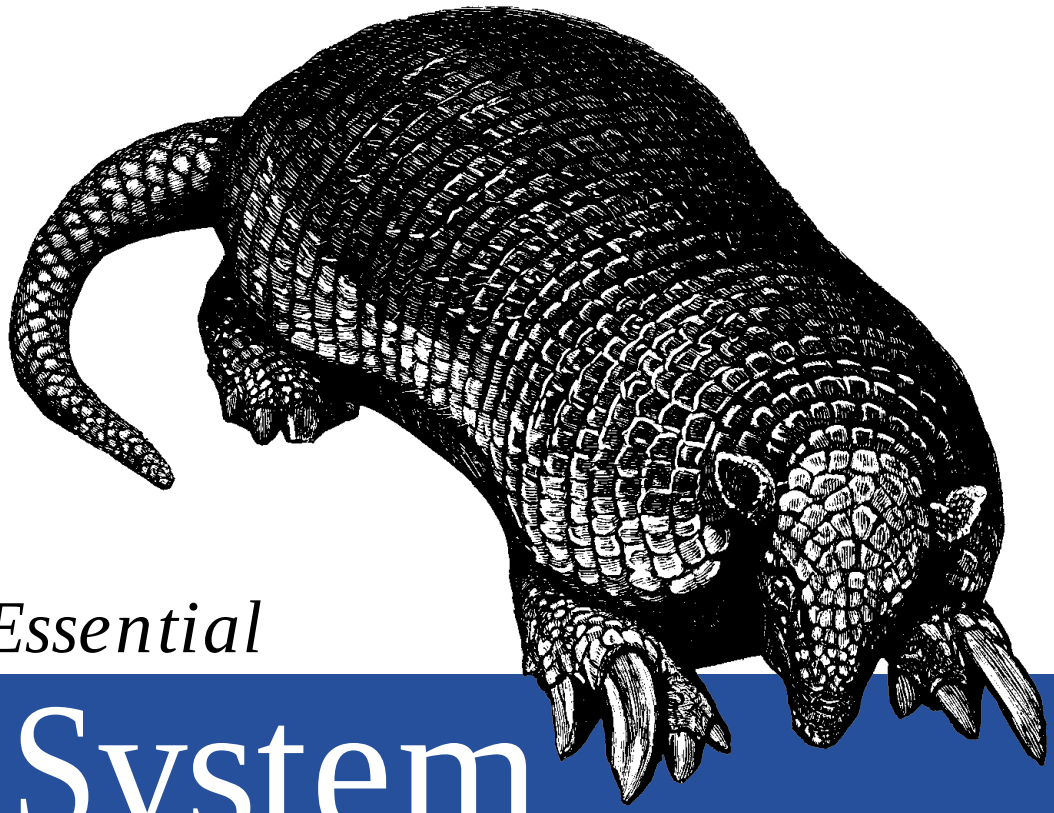
Incendie au ministère des finances à Paris – Mars 2008

Ci joint dans la version imprimée de ce cours, le chapitre gratuit d'un livre sur l'administration UNIX. Le chapitre traite de sauvegarde.

Adresse web du document fourni :
« `http://www.???.com/???/oreilly-backup.pdf` »

*Tools and Techniques for*
*Linux and Unix Administration*

3rd Edition
Revised & Updated



*Essential*

# System Administration

O'REILLY®

*Æleen Frisch*

THIRD EDITION

# Essential System Administration

*Æleen Frisch*

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

CHAPTER 11

# Backup and Restore

Every user of any computer figures out sooner or later that files are occasionally lost. These losses have many causes: users may delete their own files accidentally, a bug can cause a program to corrupt its data file, a hardware failure may ruin an entire disk, and so on. The damage resulting from these losses can range from minor to expansive and can be very time-consuming to fix. To ensure against loss, one primary responsibility of a system administrator is planning and implementing a backup system that periodically copies all files on the system to some other location. It is also the administrator's responsibility to see that backups are performed in a timely manner and that backup tapes (and other media) are stored safely and securely. This chapter will begin by discussing backup strategies and options and then turn to the tools that Unix systems provide for making them.

An excellent reference work about backups on Unix systems is *Unix Backup and Recovery* by W. Curtis Preston (O'Reilly & Associates). It covers the topics we are discussing here in complete detail and also covers material beyond the scope of this book (e.g., backing up and restoring databases).

## Planning for Disasters and Everyday Needs

Developing an effective backup strategy is an ongoing process. You usually inherit something when you take over an existing system and start out doing the same thing you've always done when you get a new system. This may work for a while, but I've seen companies try to retain their centralized, hordes-of-operators–based backup policies after they switched from a computer room full of mainframes to a building full of workstations. Such an attempt is ultimately as comical as it is heroic, but it all too often ends up only in despair, with no viable policy ever replacing the outdated one. The time to develop a good backup strategy is right now, starting from however you are approaching things at the moment.

Basically, backups are insurance. They represent time expended in an effort to prevent future losses. The time required for any backup strategy must be weighed

against the decrease in productivity, schedule slippage, and so on if the files are needed but are not available. The overall requirement of any backup plan is that it be able to restore the entire system—or group of systems—within an acceptable amount of time in the event of a large-scale failure. At the same time, a backup plan should not sacrifice too much in the way of convenience, either in what it takes to get the backup done or how easy it is to restore one or two files when a user deletes them accidentally. The approaches one might take when considering only disaster recovery or only day-to-day convenience in isolation are often very different, and the final backup plan will need to take both of them into account (and will accordingly reflect the tension between them).

There are many factors to consider in developing a backup plan. The following questions are among the most important:

***What files need to be backed up?*** The simplest answer is, of course, *everything*, and while everything but scratch files and directories needs to be saved somewhere, it doesn't all have to be saved as part of the system backups. For example, when the operating system has been delivered on CD-ROM, there is really no need to back up the system files, although you may choose do so anyway for reasons of convenience.

***Where are these files?*** This question involves both where the important files are within the filesystem and which systems hold the most important data.

***Who will back up the files?*** The answer may depend on where the files are. For example, many sites assign the backup responsibility for server systems to the system administrator(s) but make users responsible for files that they keep on their workstation's local disks. This may or may not be a good idea, depending on whether or not all of the important files really get backed up.

***Where, when, and under what conditions should backups be performed?*** Where refers to the computer system on which the backup will be performed; this need not necessarily be the same as the system where the files are physically located. Similarly, in an ideal world, all backups would be performed after hours on unmounted filesystems. That's not always practical in the real world, however.

***How often do these files change?*** This information will help you decide both when and how often to perform backups and the type of schedule to implement. For example, if your system supports a large, ongoing development project, the files on it are likely to change very frequently and will need to be backed up at least daily and probably after hours. On the other hand, if the only volatile file on your system is a large database, its filesystem might need to backed up several times every day while the other filesystems on the same system would be backed up only once a week.*

* In actual fact, a database is often backed up using a facility provided by the software vendor, but you get the idea here.

***How quickly does an important missing or damaged file need to be restored?*** Since backups protect against both widespread and isolated file loss, the timeframe in which key files need to be back online needs to be taken into account. The number of key files, how widely spread they are throughout a filesystem (or network), and how large they are will also influence matters. Your system may only have one irreplaceable file, but you'll need to plan very differently depending on whether it is 1 KB or 1 GB in size. Note that losing even a single 1 KB file can wreak havoc if it's the license file without which the central application program won't run.

***How long do we need to retain this data?*** Backups protect current data from accidents. As such, they are normally needed—or useful—only for a relatively short period (months or a year or two) In contrast, most sites also need to create permanent archives of important "point-in-time" data, for example, the software and data used to prepare a tax return. These need to be saved for an indefinite period: many years or even decades. While the requirements are similar, the goals are different enough that you are unlikely to be able to rely on your regular backups for archival purposes. Thinking about this kind of data and how to create and store it must be part of every effective backup plan.

***Where should the backup media be stored?*** Recent backups are generally kept close to the computer for quick restoration. Long-term backups and archives should be stored in a secure offsite location.

***Where will the data be restored?*** Will the backup files be used only on the system from which they were made, or is there an expectation that they could be restored to a different system in an emergency? If multisystem compatibility is ever important, it needs to be taken into account in designing the backup and recovery plan. For example, you might need to ensure that any compression scheme in use on one system can be decoded by the other target systems (or avoid using any vendor-specific formats). Other examples of this sort of issue include access control list data that might be backed up along with files and backups of a filesystem from a system that is larger than the maximum filesystem size on the target system.



**Backing up Active Filesystems**

Virtually all Unix documentation recommends that filesystems be unmounted before a backup is performed (except for the root filesystem). This recommendation is rarely followed, and in practice, backups can be performed on mounted filesystems. However, you need to make users aware that open files are not always backed up correctly. It is also true that there are circumstances in which events in an active filesystem can cause some files or even the entire backup archive itself to be corrupt. We will consider those that are relevant to the various available backup programs as we discuss them.

## Backup Capacity Planning

Once you have gathered all the data about what needs to be backed up and the resources available for doing so, a procedure like the following can be used to develop the detailed backup plan itself:

1. Begin by specifying an ideal backup schedule without considering any of the constraints imposed by your actual situation. List what data you would like to be backed up, how often it needs to be backed up, and what subdivisions of the total amount make sense.
2. Now compare that ideal schedule to what is actually possible in your environment, taking the following points into consideration:
   - When the data is available to be backed up: backing up open files is always problematic—the best you can hope for is to get an uncorrupted snapshot of the state of the file at the instant that the backup is made—so, ideally, backups should be performed on idle systems. This usually translates to after normal working hours.
   - How many tape drives (or other backup devices) are available to perform backups at those times and their maximum capacities and transfer rates: in order to determine the latter, you can start with the manufacturer's specifications for the device, but you will also want to run some timing tests of your own under actual conditions to determine realistic transfer rates that take into account the system loads, network I/O rates, and other factors in your environment. You will also need to take into account whether all the data is accessible to every backup device or not.

   At this point (as with any aspect of capacity planning), there is no substitute for doing the math. Let's consider a simple example: a site has 180 GB of data that all needs to be backed up once a week, and there are 3 tape drives available for backups (assume that all of the data is accessible to every drive). Ideally, backups should be performed only on week nights between midnight and 6 A.M. In order to get everything done, each tape drive will have to back up 60 GB of data in the 30 hours that the data is available. That means that each tape drive must write 2 GB of data per hour (333 KB/sec) to tape.

   This is within the capabilities of current tape drives when writing local data.* However, much of the data in our example is distributed across a network, so there is a chance that data might not be available at a fast enough rate to sustain the tape drive's top speed. Some backup programs also pause when they encounter an open file, giving it a chance to close (30 seconds is a typical wait period); when there are a lot of open files in a backup set, this can substantially increase how long the backup takes to complete.

* In practice, of course, you would also need an auto loading tape device (or someone to change tapes in the middle of the night).

In addition, we have not made any allowances for performing incremental backups (discussed below) between full backups. Thus, this example situation seems to strain the available resources.

3. Make modifications to the plan to take into account the constraints of your environment. Our example site is cutting things a bit too close for comfort, but they have several options for addressing this:
   - Adding additional backup hardware, in this case, a fourth tape drive.
   - Decreasing the amount of data to be backed up or the backup frequency: for example, they could perform full backups only every two weeks for some or all of the data.
   - Increasing the amount of time available/used for backups (for example, performing some backups on weekends or doing incremental backups during the early evening hours).
   - Staging backups to disk. This scheme writes the backup archives to a dedicated storage area. The files can then be written to tape at any subsequent time. Disks are also faster than tape drives, so this method also takes less time than directly writing to tape. It does, of course, require that sufficient disk space be available to store the archives.
4. Test and refine the backup plan. Actually trying it out will frequently reveal factors that your on-paper planning has failed to consider.
5. Review the backup plan on a periodic basis to determine if it is still the best solution to your site's backup needs.

## Backup Strategies

The simplest and most thorough backup scheme is to copy all the files on a system to tape or other backup media. A *full backup* does just that, including every file within a designated set of files, often defined as those on a single computer system or a single disk partition.*

Full backups are time-consuming and can be unwieldy; restoring a single file from a large backup spanning multiple tapes is often inconvenient, and when files are not changing very often, the time taken to complete a full backup may not be justified by the number of new files that are actually being saved. On the other hand, if files are changing very rapidly, and 50 users will be unable to work if some of them are lost, or when the amount of time a backup takes to complete is not an issue, then a full backup might be reasonable even every day.

* For the purposes of this discussion, I'll focus on per-disk partition backups, but keep in mind that this is not the only reasonable way of organizing things. I'll also refer to "backup tapes" most of the time in this chapter. In most cases, however, what I'll be saying will apply equally well to other backup media.

*Incremental backups* are usually done more frequently. In an incremental backup, the system copies only those files that have been changed since some previous backup. Incrementals are used when full backups are large and only a small amount of the data changes within the course of, say, one day. In such cases, backing up only the changed files saves a noticeable amount of time over performing a full backup.

Some Unix backup programs use the concept of a *backup level* to distinguish different kinds of backups. Each backup type has a level number assigned to it; by definition, a full backup is level 0. Backing up the system at any level means saving all the files that have changed since the last backup at the previous level. Thus, a level 1 backup saves all the files that have changed since the last full (level 0) backup; a level 2 backup saves all the files that have been changed since the last level 1 backup, and so on.*

A typical backup strategy using multiple levels is to perform a full backup at the beginning of each week, and then perform a level 1 backup (all files that have changed since the full backup) each day. The following weekly backup schedule summarizes one implementation of this plan:

*Monday:* Level 0 (full)
*Tuesday–Friday:* Level 1 (incremental)

A seven-day version of this approach is easy to construct.

The primary advantage of this plan is that only two sets of backup media are needed to restore the complete filesystem (the full backup and the incremental). Its main disadvantage is that the daily backups will gradually grow and, if the system is very active, may approach the size of the full backup set by the end of the week.

A popular monthly plan for sites with very active systems might look something like this:

*First Monday:* Level 0 (full)
*All other Mondays:* Level 1 (weekly incremental to previous Level 0)
*Tuesday–Friday:* Level 2 (daily incremental to previous Level 1)

This plan will require three sets of backup media to do a complete restore (the most recent backup of each type).

In deciding on a backup plan, take into account how the system is used. The most heavily used portions of the filesystem may need to be backed up more often than the other parts (such as the root filesystem, which contains standard Unix programs and files and which therefore rarely changes). A few parts of the system (like */tmp*) need never be backed up. You may want to create some additional filesystems that

* Not all backup commands explicitly use level numbers, but the concept is valid for and can be implemented with any of the available tools, provided you are willing to do some of the record keeping yourself (by hand or by script).

will never be backed up; anyone using them would be responsible for backing up his own files.

You should also consider performing a full backup—whether the schedule calls for it or not—before you make significant changes to the system, such as building a new kernel, adding a new application package, or installing a new version of the operating system. This may be one of the few times that the root filesystem gets backed up, but if you ever have a problem with your system disk, you will find it well worth the effort when you can avoid a significant amount of reconfiguration.

### Unattended backups

The worst part of doing backups is sitting around waiting for them to finish. Unattended backups solve this problem for some sites. If the backup will fit on a single tape, one approach is to leave a tape in the drive when you leave for the day, have the backup command run automatically by `cron` during the night, and pick up the tape the next morning.

Sometimes, however, unattended backups can be a security risk; don't use them if untrusted users have physical access to the tape drive or other backup device and thus could steal the media itself. Backups needed to be protected as strongly as the most secure file on the system.

Similarly, don't do unattended backups when you can't trust users not to accidentally or deliberately write over the tape or other rewriteable media (ejecting the tape after the backup is completed sometimes prevents this, but not always). You also won't be able to use them if the backup device is in heavy use and can't be tied up by the backup for the entire night.

### Data verification

In many cases, backups can simply be written to media, and the media can go directly to its designated storage location. This practice is fine as long as you are 100% confident in the reliability of your backup devices and media. In other cases, data verification is a good idea.

Data verification consists of a second pass through the backed-up data, in which each file is compared to the version on disk, ensuring that the file was backed up correctly. It also verifies that the media itself is readable.

Some sites will choose to verify the data on all backups. All sites should perform verification operations on at least a periodic basis for all of their backup devices. In addition, as they age and wear out, many devices begin to produce media that can only be successfully read in the drive that produced it. If you need backups that will be readable by devices or systems other than the one that originally wrote on the physical media, you should also periodically verify the backups' readability by examining them on the target devices and systems.

### Storing backup media

Properly storing the backup tapes, diskettes, or other media once you've written them is an important part of any backup plan. Here are some things to keep in mind when deciding where to store your backup media:

***Know where things are.*** Having designated storage locations for backups makes finding the right one quickly much more likely. It is also important that anyone who might need to do a restore knows where the media are kept (you will want to take a vacation occasionally). Installation CDs, bootable recovery tapes, boot diskettes, and the like also ought to be kept in a specific location known to those people who may need them. I can assure you from personal experience that a system failure is much more unpleasant when you have to dig through boxes of tapes or piles of CDs looking for the right one before you can even attempt to fix whatever's wrong with the system.

Another aspect of knowing where things are concerns figuring out what tape holds the file that you need to restore. Planning for this involves making records of backup contents, which is discussed later in this chapter.

***Make routine restorations easy.*** Backups should be stored close enough to the computer so that you can quickly restore a lost file, and tapes should be labeled sufficiently well so that you can find the ones you need.

Ideally, you should have a full set of tapes for each distinct operation in your backup schedule. For example, if you do a backup every day, it's best to have five sets of tapes that you reuse each week; if you can afford it, you might even have 20 sets that you rotate through every four weeks. Using a single set of tapes over and over again is inviting disaster.

Labeling tapes clearly is also a great help in finding the right one quickly later. Color-coded labels are favored by many sites as an easy yet effective way to distinguish the different sets of tapes. At the other extreme, I visited a site where the backup system they developed prints a detailed label for the tape at the conclusion of each backup.

***Write-protect backup media.*** This prevents backup media from being accidentally overwritten. The mechanism for write-protection varies with different media types, but most mechanisms involve physically moving a plastic dial or tab to some designated position. The position that is the unwriteable one varies: floppy disks, optical disks and DAT (4mm) tapes are writeable when the tabbed opening is closed, while 8mm tapes and removable disks are writeable when it is open.

***Consider the environment.*** Most backup media like it cool, dry, and dark. High humidity is probably the most damaging environment, especially for cartridge-enclosed media, which are easily ruined by the moisture condensation that accompanies temperature drops in humid conditions. Direct sunlight should also be avoided, especially for floppy disks, since most plastic materials will

deform when subjected to the temperature within the trunk of a car or the enclosed passenger compartment on a hot summer day. Dust can also be a problem for most backup media. I've had lint make floppy disks unreadable after taking them home in my coat pocket (now I put them in a zip-top plastic bag first).

The fact that backup media prefer the same environment used for many computer rooms does not necessarily mean that any or all backup media should be stored in the same room as the computer. Doing so runs the risk that a major problem will destroy both the computer and the backups. Backup tapes are actually more sensitive to some types of problems than some computer components. For example, if a pipe bursts above the computer room, the computer may suffer only minor damage, but your backup tapes will usually all be ruined if they get wet.

If the tape storage area differs in temperature from the computer area by more than a few degrees, allow the tapes to acclimate to the computer temperature before writing to them.

Magnetic interference is also something to think about. One of this book's technical reviewers relayed a story about "an entire backup library that kept getting wiped out on a nearly daily basis. Turns out that the tapes were in a secure location but placed against a wall that was shared with a freight elevator. The magnetic fields and such caused by the moving lift caused all that nice magnetic tape storage to become erased. Funny but cautionary."

***Handle media properly.*** Some media have special requirements that you'll need to take into account. For example, floppy disks and zip disks ideally should be stored upright, resting on a thin edge rather than stacked on top of one another. Similarly, cartridge tapes like to be stored with the spools vertical (perpendicular to the ground, like a car's tires) with the edge that contacts the drive heads down (so gravity pulls tape away from the spools). When you're counting on media to preserve important data, humor them and orient them the way they prefer.

***Take security into account.*** In every location where you store backup tapes, the usual physical security considerations apply: the tapes should be protected from theft, vandalism, and environmental disasters as much as is possible.

### Off-site and long-term storage

Off-site backups are the last barrier between your system and total annihilation. They are full backup sets that are kept in a locked, fireproof, environmentally-controlled location completely off site. Such backups should be performed on unmounted filesystems if at all possible.

Preparing backups for off-site storage is also one of the few times when simply making a backup is not enough.* In these cases, you also need to verify that the backup

* Another such time is when you are rebuilding a filesystem.

tape or diskette is readable. This is done by using an appropriate restore command to list the contents of the tape or diskette. While this will not guarantee that every file is completely readable, it will improve the odds of it considerably. Some backup utilities provide a full verification facility in which the entire content of each file in a backup set is compared with the corresponding file on disk; this is the preferred method of checking critical backups. In any case, backups should be verified in the best way available whenever the integrity of the backup is essential.



**Permanent Backups**

For data meant for permanent archiving, you should create and verify two sets of backup media with the idea that the redundant copy can be used should the first one fail. The media should also be checked periodically (annually or possibly biannually). When a particular media item fails—and they all will eventually—a new copy should be made from the other one to replace it.

You should also make sure that you have at least one working drive of the type that you are using for permanent storage media. For example, if you have an archive of 8 mm tapes, you will need to always have working 8 mm tape drives to read them. This will continue to be true if your primary backup medium changes. Similarly, you must maintain whatever software programs and other running environment is required to use the data for it to be of any use.

Finally, tapes should be rewound or retensioned regularly (perhaps twice a year) to maintain readability. Given this requirement, tapes are being superceded by CDs as permanent storage media.

## When Being Compulsive is Good

It's very easy to put off doing backups, especially when you are responsible only for your own files. However, performing backups regularly is vital. Basically, it's a good idea to assume that the next time you sit down at the computer, all your disks will have had head crashes. Keeping such a catastrophe in mind will make it obvious what needs to be backed up and how often. Backups are convenient for restoring accidentally deleted files, but they are also essential in the event of serious hardware failures or other disasters. Catastrophes *will* happen. All hardware has a finite lifetime, and eventually something will fail.

Given this reality, it is obvious why an almost drone-like adherence to routine is an important attribute for an effective system administrator. Planning for worst-case scenarios is part of the job. Let them call you compulsive if they want to; one day, your compulsiveness—also known to many as carefulness—will save them, or at least their files.

## Backup Media

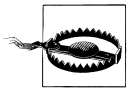
When I first started working as a system administrator, 9-track tape was the only medium you'd consider using for a backup.* That's certainly no longer true. Today, there are many different media suitable for storing backed-up data. This section provides a quick summary of the available choices. This list includes most of the drives and media types which are in common use. The backup strategy for any particular system will often involve more than one media type.

Up-to-the-minute information about available backup devices and media may be obtained from *http://www.storagemountain.com*. There is also an excellent discussion in *Unix Backup and Recovery*.

### Magnetic tape

Magnetic tape of one sort or another has been the traditional backup medium for decades. Over the years, it has taken on a variety of sizes and forms, beginning with 7-track and then 9-track tape: 1/2-inch wide tape wound around a circular reel. The introduction of plastic cartridges containing the tape and both reels was a major step forward in terms of reducing the space requirements of backup media. The first tape of this type was 1/4-inch cartridge tape (also known as QIC tape), which for a while was the medium of choice for most workstations; these tapes are still occasionally used.

Around 20 years ago, higher-capacity tapes in formats originally developed for other markets became available. 8 mm tape drives became popular in the late 1980s and are still in wide use. Originally designed for video uses, the tapes are about the size of an audio cassette. 4 mm digital audio tapes (commonly called DAT tapes although the data storage scheme is technically known as DDS) are also in wide use. DAT cartridges are about 25% smaller than 8 mm tapes.



8 mm and 4 mm tapes come in two grades, one designed for video and audio recording (respectively), and a better, more expensive grade designed for data. Be sure to purchase only data-quality tapes. Although lower-quality tapes will sometimes appear to work fine, in my experience they are much, much less reliable about retaining data (despite urban legends to the contrary).

Both of these tape types are in use today, although DAT is far more prevalent than 8 mm. Both types of tape come in a variety of lengths and corresponding data capacities. Currently, the largest ordinary 8 mm tapes are 160 meters long and hold up to

* The only other possibilities were punch cards and paper tape.

7 GB of data,* although there are also tapes that hold 1.2 GB (54 m) and 2.4 GB (112 m). DAT tapes correspond to various DDS levels:

*DDS-1*
: 2 and 3 GB tapes (60 and 90 m

*DDS-2*
: 4 GB tapes (120 m)

*DDS-3*
: 12 GB tapes (125 m)

*DDS-4*
: 20 GB tapes (150 m)

DDS-3 and DDS-4 use a different technology than the earlier versions.

Be aware that only the newest tape drives can support the largest tapes, but most drives provide read-only backward compatibility.

There are also several newer magnetic tape technologies. Exabyte's Mammoth-2† and Sony's Advanced Intelligent Tape (AIT) technologies take 8 mm tapes to much higher capacities: 20, 40, or 60 GB and 35 or 50 GB, respectively. They both use the Advanced Metal Evaporative (AME) cartridge developed by Sony (a new 8 mm format). Some Mammoth-2 drives can also read earlier 8 mm tapes, but they require an extensive clearing procedure to be performed after each instance. These are also among the fastest tape drives, with transfer rates of up to 12 MB/s for Mammoth-2 drives and 6 MB/s for AIT drives.

The Digital Linear Tape (DLT) technology was initially developed by Digital Equipment Corporation, but they later sold it to Quantum Corporation. This format uses cartridges similar to DEC's old TK family, which have proven themselves to be extremely reliable and long-lived. It is also a fast format, with transfer rates of up to 10 MB/s.

The high capacity of magnetic tapes make them ideal for unattended backups: you can put a tape in at night, start a shell script that puts several filesystems on one tape, and go home.

Tapes also have some disadvantages:

- They are extremely sensitive to heat and electromagnetic fields and fail quite easily when they are mishandled. Electromagnetic fields are produced by a variety of common devices found near computers, including UPS power supplies, external peripheral devices containing their own power supplies, monitors, and

* That is, 7 GB of bits. The amount of "data" written may be much more if the original files are compressed before or as they are written to tape. Tape drive and media manufacturers love to inflate their products' capacities by quoting maximum compressed data numbers.

† This was preceded by the Mammoth technology, which was notoriously unreliable. Mammoth-2 initially seems to be better.

speakers. Moreover, simply reading a magnetic tape also contributes to data degradation.

- They are sequential storage devices. In order to reach a given file on a tape, you have to wind the tape to the proper point. This is more of a problem for older tapes drives; current high-end drives can reach an arbitrary point on a tape in seconds.

### Magneto-optical disks

Magneto-optical disks have the same width and length as floppy disks but are about twice as thick and hold a lot more data. Magneto-optical disks also come in 3.5-inch and 5.25-inch versions,* and their current capacity ranges up to 9.1 GB. Optical disks are purported to be much more stable than any of the purely magnetic media; the stability comes from the fact that they are written magnetically but are read optically, so reading the disk has no degrading effect on the stored data. In addition, the media can also be erased and rewritten as needed. Finally, magneto-optical disks also have the advantage of being random access devices. Transfer rates for these devices peak at about 5 MB/s.

Current drives are still quite expensive—over $2000—as are the disks themselves, but they are nevertheless very popular. As I noted in the previous edition of this book (circa 1995), "a rewriteable medium that can permanently store over a gigabyte of data in the space of a couple floppy disks probably has a future." Now it's gigabytes of data and a definite future.



There are also other optical formats used or in development by a few manufacturers.

### CDs and DVDs

Writeable CDs and DVDs have become viable backup media due to the substantial price reductions for both drives and media. There are two types of writeable CDs, referred to as CD-R (write-once CDs) and CD-RW (rewriteable CDs). Both come in 640 MB capacity, and recently 700 MB CD-R media have become available.

Writeable DVD technology is just emerging into the general marketplace at this writing. In fact, there are several DVD recording formats:

*DVD-RAM*
: The first available format, it is now falling out of use since it cannot be read in ordinary DVD drives.

* You might wonder what is so magical about 3.5 and 5.25 inches. Devices of this size fit easily into the standard device bays found in PCs as well as into available storage boxes.

*DVD-R*
: Write-once DVDs (also an aging technology).

*DVD-RW*
: Rewriteable DVDs that can be read by ordinary DVD drives.

*DVD+RW*
: An emerging technology devised by a coalition of drive manufacturers. These drives can produce ordinary (sequential) DVDs as well as random access disks. The former are readable by ordinary DVD players (but not by recorders of the other types), although some older models may require firmware updates. DVD+RW media can hold up to 4.7 GB per side.

  As of this writing, Hewlett-Packard has recently released a low-cost DVD+RW writer suitable for use on PC-based systems, so this may become a popular backup device in that market in time.

### Removable disks: Zip and Jaz

Removable disks are fully enclosed disk units that are inserted into a drive as needed. They tend to be significantly more reliable than either tapes or floppy disks. On Unix systems, they generally behave like a hard disk, but it is also possible to treat them as a giant floppy disk. They are suitable as backup media in some environments and circumstances.

There have been a variety of removable disk technologies over the years. The Zip and Jaz drives by Iomega have come to dominate this market. Zip drives—which come in 100 MB and 250 MB sizes—can be used with most Unix systems. Jaz drives, which have capacities of 1 GB or 2 GB, can also be used. I had a great deal of trouble with early Jaz drives, which were designed for infrequent, intermittent backup use and consistently failed when used on even a semi-continuous basis. More recent drives are said to be better. Both drive types are available with various I/O interfaces: SCSI, USB, IDE.

### Floppy disks

Floppy disk drives are still found on most PC-based computer systems,* and they do have some limited backup uses. For example, PC-based Unix versions (as well as a few running on larger systems) often use floppy disks for emergency boot devices. In addition, floppy disks can be useful for inherently limited backup tasks, such as saving customized system configuration files from the root filesystem. Standard floppy disks hold 1.44 MB, and some Unix workstations include drives that double that capacity to 2.8 MB. Occasionally, you will come across a floppy drive that also supports Super disks: media that look like floppy disks but hold 120 MB.

* Although this will probably no longer be true in a couple of years.

### Hard disks

Given the low prices of hard disks these days, they may also be a viable backup target device in some circumstances. For example, some sites provide a large backup disk on the local network where users can make periodic copies of key files that they are working on. Large disks can also be used for scratch purposes, for temporary data repositories and data holding areas, and similar purposes. They can also be used as a staging area where backups are stored temporarily on the way to being written to tape or other media.

### Stackers, jukeboxes, and similar devices

There are a variety of devices designed to make media handling more automated, as well as to store and make available large numbers of media units. For example, there are auto-loading tape drives—also known as *stackers*—which can feed tapes automatically from a stack of 10 or so. Early stackers could access the component tapes only in order, but many current devices can retrieve any desired tape.

Another type of device puts multiple drive units into a box that looks to users like a single tape drive with the combined capacity of all of its components. Alternatively, such a device can be used to make multiple identical tapes simultaneously.

Still other units combine both multiple drives and tape auto-loading capabilities. These devices are known as *jukeboxes* or *libraries*.* The most sophisticated of them can retrieve a specific tape and place it into the desired drive. Some of these devices include integrated bar-code readers so that tapes can be identified by their physical label rather than storage location or electronic label. Similar devices also exist for optical disks and writeable CD-ROMs.

## Media Lifetime

From time to time, you also need to think about the reasonable expected lifetime of your backup media. Stored under the right conditions, tapes can last for years, but unfortunately you cannot count on this. Some manufacturers recommend replacing tapes every year. This is certainly a good idea if you can afford to do so. The way that tapes and diskettes are stored also affects their lifetime: sunlight, heat, and humidity can all significantly shorten it. I always replace tapes that have had read errors or other failures more than once, regardless of their age; for some people and situations, a single failure is enough. I always throw away diskettes and Zip disks at the first hint of trouble.

* Very large libraries (greater than 500 volumes) are known as *silos*. The two types of devices used to be distinguished by whether or not multiple hosts could be connected, but some libraries now have this capability. Separate silos are also able to pass tapes between them.

Even so-called permanent media like CDs actually have a finite lifetime. For example, CDs begin to fail after about 5 years (and sometimes even sooner). Accordingly, creating two copies of important data and checking them periodically is the only prudent course.

Given these considerations, your site may want to consider alternative media for offsite and archival backups. For example, manufacturers of optical disks claim a lifetime of 15 years for this media (this is based on accelerated aging tests; as of this writing, we won't know for about 8–9 years whether this is really true).

## Comparing Backup Media

Table 11-1 lists the most important characteristics of a variety of backup media. The largest media capacity for each item shown is the biggest that was available as of this writing. These size values refer to raw data capacity: the actual amount of data that can be written to the media.

The drive price is the lowest generally available price at this time and can be assumed to use the least expensive I/O interface; you can expect SCSI versions of many devices that are also available in IDE form to cost at least 15% more (and sometimes much more). Similarly, at about $100, a USB floppy drive costs 10 times that of an ordinary one.

The media prices are the lowest commonly available when the media is purchased in large quantities (e.g., 50–100 for CDs) and in no-frills packaging (e.g., on a spindle for CDs rather than in individual jewel cases). All prices are domestic prices in the United States, in U.S. dollars, as of mid-2002.

The minimum lifetime column gives an approximate rule-of-thumb time period when you can expect some media to begin failing. Of course, individual media will fail even sooner in some cases.

*Table 11-1. Popular backup devices and media*

| Type | Media capacity | Drive price[a] | Media price[a] | Minimum lifetime |
|---|---|---|---|---|
| Floppy disk | 1.44 MB[b] | $10 | $0.25 | 2 years |
| Super disk | 120 MB | $120 | $8 | 2–3 years |
| Zip Disk | 100 MB | $70 | $5 | 3–5 years |
| | 250 MB | $140 | $12 | 3–5 years |
| Jaz Disk | 1 GB | $300 | $80 | 4–5 years |
| | 2 GB | $340 | $100 | 4–5 years |
| CD-R | 700 MB (80 minutes) | $150 | $0.85 | 5 years |
| CD-RW | 640 MB (74 minutes) | $150 | $1 | 5 years |
| DVD-R | 4.7 GB (single-sided) | $700 | $8 | 5 years? |
| | 9.4 GB (double-sided) | $700 | $40 | 5 years? |

*Table 11-1. Popular backup devices and media (continued)*

| Type | Media capacity | Drive price[a] | Media price[a] | Minimum lifetime |
|---|---|---|---|---|
| DVD+RW | 4.7 GB | $600 | $8 | 5 years? |
| DAT tape 4 mm DDS | 4 GB (120 m DDS-2) | $550 | $6 | 3–4 years |
| | 12 GB (125 m DDS-3) | $700 | $12.50 | 3–4 years |
| | 20 GB (150 m DDS-4) | $1200 | $26 | 3–4 years |
| 8 mm tape | 7 GB (160 m) | $1200 | $6 | 2–4 years |
| Mammoth-2 (AME) | 20 GB | $2500 | $36 | 3–4 years? |
| | 60 GB | $3700 | $45 | 3–4 years? |
| AIT tape | 35 GB | $900 | $79 | 3–4 years? |
| | 50 GB | $2600 | $85 | 3–4 years? |
| | 100 GB | $3900 | $105 | 3–4 years? |
| DLT | 40 GB | $4000 | $70 | 10 years |
| SuperDLT | 110 GB | $6000 | $150 | 10 years |
| Magneto-optical (RW) | 5.2 GB | $2300 | $65 | 15 years? |
| | 9.1 GB | $2700 | $93 | 15 years? |
| Hard disk | 100 GB (IDE) | N/A | $2–3/GB | 5–7 years |
| | 180 GB (SCSI) | N/A | $10/GB | 5–7 years |

[a] Approximate minimum price in U.S. dollars.
[b] A few floppy drives provided by Unix vendors increase the maximum capacity to 2.8 MB.

## Tape Special Files

Traditionally, special files used to access tape drives had names of the form */dev/rmt*n or */dev/rmt/*n, where *n* indicates the drive number. Tape drives are virtually always accessed via the character (raw) special file. Currently, special file names usually include other characters as prefixes and/or suffixes, which indicate the way the device is to be accessed: the density setting to use, whether to use the drive's built-in hardware compression, whether to rewind the tape after the operation is completed, and so on.

AIX systems also use suffixes to select whether the tape should be retensioned before use. Retensioning refers to equalizing the tension on a tape, and it consists of moving the tape to the beginning, then the end, and then rewinding back to the beginning; it's even slower than it sounds. The idea is to eliminate any latent slackness in the tape, but it is seldom necessary in practice.

Table 11-2 lists the current tape special file naming conventions for the various operating systems we are considering.

*Table 11-2. Tape special file names*

| Unix version | Format and examples[a] | Prefixes/suffixes | man page |
|---|---|---|---|
| AIX | */dev/rmt*n[.m]<br>*/dev/rmt0.1*<br>*/dev/rmt0.5*<br>**Note**: Compression is enabled and disabled with the `chdev` command. | m:<br>none=rewind, no retension, low density<br>*1*=no rewind, no retension, low density<br>*2*=rewind, retension, low density<br>*3*=no rewind, retension, low density<br>*4*=rewind, no retension, high density<br>*5*=no rewind, no retension, high density<br>*6*=rewind, retension, high density<br>*7*=no rewind, retension, high density | `rmt(4)` |
| FreeBSD | */dev/*[*n*]*rast*n<br>*/dev/*[*e*|*n*]*rsa*n<br>*/dev/nrast0*<br>*/dev/nrsa0* | *n*=no rewind<br>*e*=eject tape on close<br>(Density and compression are chosen with the `mt` utility.) | `ast`<br>`sa(4)` |
| HP-UX | */dev/rmt/c*i*t*j*d0*TYPE[*b*][*n*]<br>*/dev/c0t3d0DDSbn*<br>*/dev/c0t3d0BESTbn* | i=controller<br>j=SCSI ID<br>*n*=no rewind<br>*b*=use BSD-style error control<br>TYPE=keyword indicating tape type and/or density (e.g., *BEST*, *DDS*) | `mt(7)` |
| Linux | */dev/*[*n*]*st*nx<br>*/dev/nst0*<br>*/dev/nst0m* | *n*=no rewind<br>x:<br>none=default density<br>*l*=low density<br>*m*=medium density<br>*a*=autoselect density | `st` |
| Solaris | */dev/rmt/*nx[*b*][*n*]<br>*/dev/rmt/0lbn*<br>*/dev/rmt/0hbn* | *b*=use BSD-style error control<br>*n*=no rewind<br>x:<br>none=default density<br>*l*=low density<br>*m*=medium density<br>*h*=high density<br>*c*=use hardware compression | `st` |
| Tru64[b] | */dev/*[*n*]*rmt/tape*n*_d*m<br>*/dev/nrmt/tape0_d2*<br>*/dev/nrmt/tape0_d3* | m:<br>*0*=low density, use compression<br>*1*=high density, use compression<br>*2*=low density, no compression<br>*3*=high density, no compression<br>(values 4–7 are also defined for some drives) | `tz` |

[a] In all cases, n refers to the tape drive number. The examples are all for a non-rewinding tape device with hardware compression disabled using the lowest and highest density (as available).

[b] Older Tru64 systems use the now-obsolete device names of the form */dev/tz** and */dev/ta**.

Some systems provide simpler names as links to commonly-used tape devices. You can figure out which device they refer to by looking at a long directory listing. Here is an example from an HP-UX system:

```
crw-rw-rw- 2 bin bin 205 0x003000 Oct 7 1999 0m
crw-rw-rw-  2 bin    bin    205 0x003080 Oct  7  1999 0mb
crw-rw-rw-  2 bin    bin    205 0x003040 Oct  7  1999 0mn
crw-rw-rw-  2 bin    bin    205 0x0030c0 Oct  7  1999 0mnb
crw-rw-rw-  2 bin    bin    205 0x003000 Oct  7  1999 c0t3d0BEST
crw-rw-rw-  2 bin    bin    205 0x003080 Oct  7  1999 c0t3d0BESTb
crw-rw-rw-  2 bin    bin    205 0x003040 Oct  7  1999 c0t3d0BESTn
crw-rw-rw-  2 bin    bin    205 0x0030c0 Oct  7  1999 c0t3d0BESTnb
crw-rw-rw-  1 bin    bin    205 0x003001 Oct  7  1999 c0t3d0DDS
crw-rw-rw-  1 bin    bin    205 0x003081 Oct  7  1999 c0t3d0DDSb
crw-rw-rw-  1 bin    bin    205 0x003041 Oct  7  1999 c0t3d0DDSn
crw-rw-rw-  1 bin    bin    205 0x0030c1 Oct  7  1999 c0t3d0DDSnb
```

In this case, *0m* and *c0t3d0BEST* refer to the same tape drive and access mode (as do their corresponding suffixed forms).

The default tape drive on a system is usually the first drive in its default (rewinding) mode:

| | |
|---|---|
| AIX | */dev/rmt0* |
| FreeBSD | */dev/rsa0* |
| HP-UX | */dev/rmt/0m* |
| Linux | */dev/st0* |
| Solaris | */dev/rmt/0* |
| Tru64 | */dev/rmt/tape0_d0* |

On Linux systems (and some others), the device */dev/tape* is a link to the default tape device on the system. You can make the link point to whatever drive you want to by recreating the link. On FreeBSD systems, some commands use the *TAPE* environment variable to locate the default tape drive.

#### AIX tape device attributes

On AIX systems, you can use the `lsattr` command to view the attributes of a tape drive:

```
$ lsattr -E -H -l rmt0
attribute      value  description                    user_settable

block_size     1024   BLOCK size (0=variable length)     True
compress       yes    Use data COMPRESSION               True
density_set_1  140    DENSITY setting #1                 True
density_set_2  20     DENSITY setting #2                 True
extfm          yes    Use EXTENDED file marks            True
mode           yes    Use DEVICE BUFFERS during writes   True
```

This 8 mm tape drive will use data compression and a block size of 1024 by default.

You must use the chdev command to change the many attributes of a tape drive (rather than having these selections encoded into the special file name as with other systems). For example, the following command changes the block size to 1024 and turns off compression and retensioning for drive 1:

```
# chdev -l rmt0 -a block_size=1024 -a compress=no -a ret=no
```

## Backing Up Files and Filesystems

Most systems offer a variety of utilities for performing backups, ranging from general-purpose archiving programs like tar and cpio to programs designed for implementing multilevel incremental backup schemes on a per-filesystem basis. When the largest tapes held only a couple hundred megabytes, choosing the right utility for system backups was easy. tar and cpio were used for small and ad hoc backups and other data transfer needs, and the more sophisticated utilities specifically designed for the task were used for system backups, because their specialized abilities—the ability to span tapes and to automatically perform incremental backups—were essential to getting the job done.

This distinction breaks down to a great extent when a single tape can hold gigabytes of data. For example, incrementals are less important when you can fit all the important data on a system onto one or two tapes—and you have the time to do so. Large tapes also make it practical to back up a system in logically grouped chunks of files, which may be spread arbitrarily throughout the physical filesystem. A successful system backup process can be built around whatever utilities make sense for your system.



One dubious piece of advice about backups that is frequently given is that you should limit filesystem size to the maximum backup media capacity available on the system. In this view, multi-tape backup sets are simply too much trouble, and the backup process is simplified if all of the data from a filesystem will fit onto a single tape.

While being able to back up a filesystem with a single tape is certainly convenient, I think it is a mistake to let current media capacity dictate filesystem planning to such a degree. Breaking disks into more, smaller filesystems limits flexibility in allocating their resources, a concern that is almost always far more important than reducing the complexity of backing them up. Designing the filesystem needs to take *all* of the factors affecting the system and its efficient use into account. If tape-sized backup sets are what is desired, it's easy enough to write scripts to do so when overall circumstances dictate that some individual filesystems need to be bigger.

## When tar or cpio Is Enough

In some cases, especially single-user systems, an elaborate backup process is not needed. Rather, since the administrator and the user are one and the same person, it will be obvious which files are important, how often they change, and so on. In cases like this, the simpler tape commands, `tar` and `cpio`, may be sufficient to periodically save important files to tape (or other media).

While the canonical model for this situation is Unix running on a workstation, these utilities may also be sufficient for systems with relatively small amounts of critical data. `tar` and `cpio` also have the advantage that they will back up both local and remote filesystems mounted via NFS.

### The tar command

We'll begin with a simple example. The following `tar` command saves all files under */home* to the default tape drive:

```
$ tar -c /home
```

`-c` says to create a backup archive.

`tar`'s `-C` option (big C) is useful for gathering files from various parts of the filesystem into a single archive. This option causes the current directory to be set to the location specified as its argument before `tar` processes any subsequent pathname arguments. Multiple `-C` options may be used on the same command. For example, the following `tar` commands save all the files under the directories */home*, */home2*, and */chem/public*:

```
$ tar -cf /dev/rmt1 /home /home2 /chem/public
$ tar -cf /dev/rmt1 -C /home . -C /home2 . -C /chem public
```

The two commands differ in this respect: the first command saves all of the files using absolute pathnames: */home/chavez/.login*, for example. The second command saves files using relative pathnames: *./chavez/.login*. The file from the first archive would always be restored to the same filesystem location, while the file from the second archive would be restored relative to the current directory (in other words, relative to the directory from which the restore command was given).

It is a good idea to use absolute pathnames in the arguments to `-C`. Relative pathnames specified to `-C` are interpreted with respect to the current directory at the time that option is processed rather than with respect to the initial current directory from which the `tar` command was issued. In other words, successive `-C` options accumulate, and `tar` commands using several of them as well as relative pathnames can become virtually uninterpretable.



Traditionally, all tar options were placed in a single group immediately following the command verb, and a preceding hyphen was not needed. The POSIX standard specifies a more traditional Unix syntax, preferring the second form to the first one for this command:

```
$ tar xpfb /dev/rmt1 1024 ...
$ tar -x -p -f /dev/rmt1 -b 1024 ...
```

The versions of tar on current operating systems usually accept both formats, but an initial hyphen may become be a requirement at some point in the future.

tar archives are often compressed, so it is very common to see compressed tar archives with names like *file.tar.Z*, *file.tar.gz* or *file.tgz* (the latter two files are compressed with the GNU gzip utility).

**Solaris enhancements to the tar command.** The Solaris version of tar offers enhancements that make the command more suitable for system-level backups. They allow all or part of the list of files and directories to be backed up to be placed in one or more text files (with one item per line). These files are included in the file list given to tar, preceded by -I, as in this example:

```
$ tar cvfX /dev/rst0 Dont_Save /home -I Other_User_Files -I Misc
```

This command backs up the files and directories in the two include files, as well as those in */home*. The command also illustrates the use of the -X option, which specifies the name of an exclusion file listing the names of files and directories that should be skipped if encountered by tar. Note that wildcards are not permitted in either include or exclusion files. In case of conflicts, exclusion takes precedence over inclusion.



The -I and -X options may also be used in restore operations performed with the tar command.

On Solaris and a variety of other System V systems, the file */etc/default/tar* may be used to customize the mappings of the default archive destinations specified with tar's single-digit code characters (for example, the command tar 1c creates an archive on drive 1). Here is a version from a Solaris system:

```
#                               Block   #
#Archive=Device                 Size    Blocks
#
archive0=/dev/rmt/0             20      0
archive1=/dev/rmt/0n            20      0
archive2=/dev/rmt/1             20      0
archive3=/dev/rmt/1n            20      0
archive4=/dev/rmt/0             126     0
archive5=/dev/rmt/0n            126     0
```

```
archive6=/dev/rmt/1             126     0
archive7=/dev/rmt/1n            126     0
```

The first entry specifies the device that will be used when `tar 0` is specified. In this case, it is the first tape drive in its default modes. The second entry defines archive 1 as the first tape drive in non-rewinding mode. The remaining two fields are optional; they specify the block size for the device and its total capacity (which may be set to zero to have the command simply detect the end-of-media marker).

**The GNU tar utility: Linux and FreeBSD.** Linux distributions and FreeBSD provide the GNU version of the `tar` command. It supports `tar`'s customary features and contains some enhancements to them, including the ability to optionally span media volumes (`-M`) and to use `gzip` compression (`-z`). For example, the following command will extract the contents of the specified compressed `tar` archive:

```
$ tar xfz funsoftware.tgz
```

### The cpio command

`cpio` can also be used to make backups. It has several advantages:

- It is designed to easily back up completely arbitrary sets of files; `tar` is easiest to use with directory subtrees.
- It packs data on tape much more efficiently than `tar`. If fitting all your data on one tape is an issue, `cpio` may be preferable.
- On restores, it skips over bad spots on the tape, while `tar` just dies.
- It can span tapes, while many versions of `tar` are limited to a single volume.

Using its `-o` option, `cpio` copies the files whose pathnames are passed to it via standard input (often by `ls` or `find`) to standard output; you redirect standard output to use `cpio` to write to floppy disk or tape. The following examples illustrate some typical backup uses of `cpio`:

```
$ find /home -print | cpio -o >/dev/rmt0
$ find /home -cpio /dev/rmt0
```

The first command copies all files in */home* and its subdirectories to the tape in drive 0. The second command performs the identical backup via a version of `find` that offers a `-cpio` option.

### Incremental backups with tar and cpio

Combining `find` with `tar` or `cpio` is one easy way to perform incremental backups, especially when only two or three distinct backup levels are needed. For example, the following commands both copy all files under */home* which have been modified today into an archive on */dev/rmt1*, excluding any object (*.o*) files:

```
$ find /home -mtime -1 ! -name \*.o -print | cpio -o >/dev/rmt1
$ tar c1 `find /home -mtime -1 ! -name `*.o' ! -type d -print`
```

The find command used with tar needs to exclude directories, because tar will automatically archive *every* file underneath any directory named in the file list, and all directories in which *any* file has changed will appear in the output from find.

You can also use find's -newer option to perform an incremental backup in this way:

```
$ touch /backup/home_full
$ find /home -print | cpio -o > /dev/rmt0
A day later…
$ touch /backup/home_incr_1
$ find /home -newer /backup/home_full -print | cpio -o > /dev/rmt0
```

The first command timestamps the file */backup/home_full* using the touch command *(/backup* is a directory created for such backup time records), and the second command performs a full backup of */home*. Some time later, the second two commands could be used to archive all files that whose data has changed since the first backup and to record when it began. Timestamping the record files before this backup begins ensures that any files that are modified while it is being written will be backed up during a subsequent incremental, regardless of whether such files have been included in the current backup or not.

### pax: Detente between tar and cpio

The pax command attempts to bridge the gap between tar and cpio by providing a single general-purpose archiving utility.* It can read and write archives in either format (by default, it writes tar archives), and offers enhancements over both of them, making it an excellent utility for system backups in many environments. pax is available for all of the Unix versions we are considering. Like cpio, pax archives may span multiple media volumes.

pax's general syntax is:

```
pax [mode_option] other_options files_to_backup
```

The *mode_option* indicates whether files are being written to or extracted from an archive, where -w says to write to an archive, -r says to read and extract from an archive, and -rw indicates a pass-through mode in which files are copied to an alternate directory on disk (as with cpio -p); pax's default mode when no *mode_option* is given is to list the contents of an archive.

The following commands illustrate pax file archiving modes of operation:

```
$ pax -w -f /dev/rmt0 /home /chem
$ find /home /chem -mtime -1 -print | pax -w -f /dev/rmt0
$ pax -w -X -f /dev/rmt0 /
```

* Indeed, on systems offering pax, cpio and tar are often just links to it. pax's syntax is an amalgamation of the two, which is not surprising for a peace imposed by POSIX (although the name purportedly stands for *portable archive exchange*).

The first two commands perform a full and incremental backup of the files in */home* and */chem* to the default tape drive in each case. The third command saves all of the files in the disk partition corresponding to the root directory; the `-X` option tells `pax` not to cross filesystem boundaries.

AIX prefers `pax` over vanilla `tar` and `cpio`. The command has been enhanced to support large files (over 2 GB).

### Getting Users to Do Backups

At some sites, certain backup responsibilities are left to individual users: when a site has far too many workstations to make backing up all of their local disks practical, when important data resides on non-Unix systems like PCs (especially if they are not connected to the local area network), and so on.

However, even when you're not actually performing the backups yourself, you will probably still be responsible for providing technical support and, more often than not, reminders to the users who will be performing the backups. Here are some approaches I've tried to facilitate this:

- Make a habit of encouraging users rather than threatening them (threats don't work anyway).
- Use peer pressure to your advantage. Setting up a central backup storage location that you look after can make it obvious who is and isn't doing the backups they are supposed to. Note that this idea is inappropriate if data sensitivity is an issue.
- Create tools that automate the backup process as much as possible for users. Everyone has time to drop in a tape and start a script before they leave for the day.
- Provide a central repository for key files that get backed up as part of the system/site procedure. Users can copy key files and know they will be backed up when they're really in a jam and really don't have time to do a backup themselves.

## Backing Up Individual Filesystems with dump

The BSD `dump` utility represents the next level of sophistication for backup systems under Unix. It selectively backs up all of the files within a filesystem (single disk partition), doing so by copying the data corresponding to each inode to the archive on the backup device. It also has the advantage of being able to back up any type of file, including device special files and sparse files. Although there are slight variations among different versions of this command, the discussion here applies to the following Unix implementations of this command:

| | |
|---|---|
| AIX | `backup` |
| FreeBSD | `dump` |

| | |
|---|---|
| HP-UX | `dump` and `vxdump` |
| Linux | `dump` (but the package is not usually installed by default) |
| Solaris | `ufsdump` |
| Tru64 | `dump` and `vdump` |

On systems supporting multiple filesystem types, `dump` may be limited to UFS (BSD-type) filesystems; on Linux systems, it is currently limited to ext2/ext3 filesystems, although the XFS filesystem provides the similar `xfsdump` utility. Under HP-UX, `vxdump` and `vxrestore` support VxFS filesystems. Tru64 provides `vdump` for AdvFS filesystems.

`dump` keeps track of when it last backed up each filesystem and the level at which it was saved. This information is stored in the file */etc/dumpdates* (except on HP-UX systems, which use */var/adm/dumpdates*). A typical entry in this file looks like this:

```
/dev/disk2e       2 Sun Feb  5 13:14:56 1995
```

This entry indicates that the filesystem */dev/disk2e* was last backed up on Sunday, February 5 during a level 2 backup. If `dump` does not find a filesystem in this list, it assumes that it has never been backed up.

If *dumpdates* doesn't exist, the following command will create it:

```
# touch /path/dumpdates
```

The *dumpdates* file must be owned by the user *root*. If it does not exist, `dump` will not create it and won't record when filesystem backups occur, so create the file before running `dump` for the first time.

The `dump` command takes two general forms:

```
$ dump options-with-arguments filesystem
$ dump option-letters corresponding-arguments filesystem
```

where *filesystem* is the block special file corresponding to the filesystem to be backed up or the corresponding mount point from the filesystem configuration file. In the first, newer form, the first item is the list of options to be used for this backup, with their arguments immediately following the option letters in the normal way (e.g., `-f /dev/tape`).

In the second, older form, *option-letters* is a list of argument letters corresponding to the desired options, and *corresponding-arguments* are the values associated with each argument, in the same order. This syntax is still the only one available under Solaris and HP-UX.

Although not all options require arguments, the list of arguments must correspond *exactly*, in order and in number, to the options requiring arguments. For example, consider the set of options `0sd`. The `s` and `d` options require arguments; `0` does not. Thus, a `dump` command specifying these options must have the form:

```
$ dump 0sd s-argument d-argument filesystem
```

Failing to observe this rule can have painful consequences if you are running the command as *root*, including destroying the filesystem if you swap the argument to the `f` option and `dump`'s final argument when you are running the command as *root*. You'll get no argument from me if you want to assert that this is a design defect that ought to have been fixed long before now. When you use `dump`, just make sure an argument is supplied for each option requiring one. To avoid operator errors, you may want to create shell scripts that automatically invoke `dump` with the proper options.

`dump`'s most important options are the following (we will use the newer form):

`-0`, ..., `-9`
: These options indicate the level of the dump this command will perform. Given any level *n*, `dump` will search *dumpdates* for an entry reporting the last time this filesystem was dumped at level *n–1* or lower. `dump` then backs up all files that have been changed since this date. If *n* is zero, `dump` will back up the entire filesystem. If there is no record of a backup for this filesystem for level *n–1* or lower, `dump` will also back up the entire filesystem. If no level option is specified, it defaults to `-9`. This option does not require any argument.

  Older versions of `dump` not supporting hyphenated options require that the level option be the first option letter.

`-u`
: If `dump` finishes successfully, this option updates its history file, *dumpdates*. It does not require an argument.

`-f` *`device`*
: This option states that you want to send the dump to something other than the default tape drive (i.e., to a file or to another device). The defaults used by various Unix versions were listed previously. If you use this option, it must have an argument, and this argument must precede the filesystem being dumped. A value of "`-`" (a single hyphen) for its argument indicates standard output.

`-W`
: Display only what will be backed up when the indicated command is invoked, but don't perform the actual backup operation.

`-s` *`feet`* `-d` *`dens`*
: These options were needed on older versions of `dump` to determine the capacity of the backup media. Recent versions of `dump` generally don't need them as they keep writing until they detect an end-of-media mark.

  If you do need to use them to lie to `dump` about the tape length because your version uses a default capacity limit suitable for ancient 9-track tapes, `-s` specifies the *size* of the backup tape, in feet; `-d` specifies the density of the backup tape, in bits per inch. Since `dump` will respect end-of-media marks that it encounters before it has reached this limit, the fix for such situations is to set the capacity to

something far above the actual limit. For example, the options `-d 50000 -s 90000` define a tape capacity somewhat over 4 GB.

`-b` *factor*
: Specifies the block size to use on the tape, in units of 1024-byte (or sometimes 512-byte) blocks.

Here is a typical use of the `dump` command:

```
$ dump -1 -u -f /dev/tape /chem
```

The second command performs a level 1 incremental backup on the */chem* filesystem using the tape drive linked to */dev/tape*; `dump` will update the file the *dumpdates* file upon completion.

`dump` notifies the user whenever it requires some interaction. Most often, `dump` will have filled the tape currently in use and ask for another. It will also ask whether to take corrective actions if problems arise. In addition, `dump` prints many messages describing what it is doing, how many tapes it thinks it will need, and the like.

### The HP-UX fbackup utility

HP-UX provides the `fbackup` and `frecover` utilities designed to perform system backups. One significant advantage that they have over the standard Unix utilities is that they can save and restore HP-UX access control lists along with other file metadata.

`fbackup` provides for nine levels of incremental backups, just like `dump`. `fbackup` stores backup records in the file */var/adm/fbackupfiles/dates*, which the system administrator must create before using `fbackup`.

The following examples illustrate how `fbackup` might be used for system backup operations:

```
# fbackup -0u -f /dev/rmt/1m -i /chem
# fbackup -1u -i /chem -i /bio -e /bio/med
# fbackup -1u -f /dev/rmt/0m -f /dev/rmt/1m -i /chem
# fbackup -0u -g /backup/chemists.graph -I /backup/chemists.TOC
```

The first command performs a full backup of */chem* to tape drive 1, updating the `fbackup` database. The second command does a level 1 backup of */chem* and */bio*, excluding the directory */bio/med* (as many `-i` and `-e` options as you need can be included). The third command performs a level 1 backup of */chem* using multiple tape drives in sequence.

The final command performs a full backup as specified by the graph file */backup/chemists.graph*, writing an index of the backup to the file */backup/chemists.TOC*. A graph file is a text file with the following format:

```
c   path
```

where *c* is a code indicating whether *path* is to be included (`i`) or excluded (`e`) from the backup.

## Related Tape Utilities

There are two other Unix tape utilities you should know about, which are also of use in performing backups from time to time.

### Data copying and conversion with dd

The `dd` utility transfers raw data between devices. It is useful for converting data between systems and for reading and writing tapes from and to non-Unix systems. It takes a number of *`option=value`* pairs as its arguments. Some of the most useful options are:

`if` Input file: source for data.

`of` Output file: destination for data.

`ibs`
: Input block size, in bytes (the default is 512).

`obs`
: Output block size, in bytes (the default is 512).

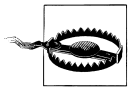
`fskip`
: Skip tape files before transferring data (not available in all implementations).

`count`
: The amount of data (number of blocks) to transfer.

`conv`
: Keyword(s) specifying desired conversion of input data before outputting: `swab` means swap bytes, and it is the most used conversion type. `lcase` and `ucase` mean convert to lower- and uppercase, respectively, and `ascii` and `ebcdic` mean convert to ASCII or EBCDIC.

For example, the following command processes the third file on the tape in drive 0, using an input block size of 1024 bytes and swapping bytes in all data; the command writes the converted output to the file */chem/data/c70o.dat*:

```
$ dd if=/dev/rmt0 of=/chem/data/c70o.dat \
     ibs=1024 fskip=2 conv=swab
```



As always, be careful to specify the appropriate devices for `if` and `of`; transposing them can have disastrous consequences.

### Tape manipulation with mt

Unix provides the `mt` command for direct manipulation of tapes. It can be used to position tapes (to skip past backup save sets, for example), to rewind tapes, and to perform other basic tape operations. Its syntax is:

```
$ mt [-f tape-device] command
```

where *tape-device* specifies which tape drive to use, and *command* is a keyword indicating the desired action. Useful keywords include `rewind` (to rewind the tape), `status` (display device status—you can see whether it is in use, for example), `fsf` *n* (skip the next *n* files), and `bsf` *n* (skip back *n* files).

For example, to rewind the tape in the second tape drive, you might use a command like:

```
$ mt -f /dev/rmt1 rewind
```

The Solaris version of `mt` includes an `asf` subcommand, which moves the tape to the *n*th file on the tape (where *n* is given as `asf`'s argument), regardless of the tape's current position.

Under FreeBSD, the `mt` command is used to set the tape drive density and compression:

```
$ mt -f /dev/nrsa0 comp on density 0x26
```

AIX also includes the `tctl` utility (to which `mt` is really a link). `tctl` has the same syntax as `mt` and offers a few additional seldom-wanted subcommands.

# Restoring Files from Backups

All of the backup facilities described in the previous sections have corresponding file restoration facilities. We'll look at each of them in turn in this section.

## Restores from tar and cpio Archives

Individual files or entire subtrees can be restored easily from `tar` and `cpio` archives. For example, the following pairs of commands restore the file */home/chavez/freeway/quake95.data* and user *harvey*'s home directory (respectively) from an archive made of */home* located on the tape in the default tape drive (here, we use */dev/rmt0* for as the example location):

```
$ tar -xp /home/chavez/freeway/quake95.data
$ cpio -im '*quake95.data' < /dev/rmt0
$ tar -xp /home/harvey
$ cpio -imd '/home/harvey*' < /dev/rmt0
```

The `-p` option to `tar` and `-m` option to `cpio` ensure that all file attributes are restored along with the file. `cpio`'s `-d` option creates subdirectories as necessary when restoring a directory subtree (`tar` does so by default).*

* The second `cpio` command also assumes that there is no file or directory in */home* that begins with "harvey" other than user *harvey*'s home directory.

Restores with pax are similar. For example, the first of the following commands lists the files on the tape in drive 0, and the remaining commands extract various files from it:

```
$ pax -f /dev/rmt0 -v                  –v  gives a more detailed/verbose listing.
$ pax -r '/h95/*.exe'                  Select files via a regular expression.
$ pax -r /home/chavez                  Restore chavez's home directory.
$ pax -r -f my_archive -c '*.o'        Restore everything except object files.
# pax -r -pe -f /dev/rmt0              Restore files incl. owner, mode & mod. time.
```

pax's coolest feature has to be its -s option, which allows you to massage filenames as files are written to, extracted from, or even just listed from an archive. It takes a substitution command as used in ed or sed as its argument (which will usually need to be enclosed in single quotation marks) indicating how filenames should be transformed. For example, the following command changes the second-level directory name of each file from *chavez* to *harvey* as files are read from the archive, changing their target location on disk:

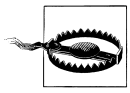
```
$ pax -r -s ',^/home/chavez/,/home/harvey/,' \
      -f /dev/rmt0 /home/chavez
```

The substitution clause searches for */home/chavez* at the beginning of the pathname of each file to be restored and changes it to */home/harvey*, using commas as the field separator within the substitution string.

Here are some additional **-s** clauses for specific kinds of transformations:

```
-s ',^/home/chavez/,,'     Remove partial directory component.
-s ',^.*//*,,'             Remove entire directory component.
-s ',^//*,,'               Make pathnames relative to current directory.
```

Multiple -s options are allowed, but only the first matching one is used for any given filename.



Be aware that pax is not without its eccentricities. One of the most annoying is the following: in some versions of pax, directories matched via wildcards in the pattern list during restore operations are not extracted in their entirety; only explicitly listed ones are. Note that this is the opposite of the way cpio works and also counter to the way tar operates. I'd be positive this was a bug except that it happens in more than one vendor's version, although not in every vendor's version. With pax, *caveat emptor* would appear to be the watchword.

## Restoring from dump Archives

The restore utility retrieves files from backup tapes made with the dump utility. It is supported by those systems supporting a version of dump. Solaris calls its version ufsrestore in keeping with the name of its version of dump. HP-UX and Tru64 provide vxrestore and vrestore commands for their default filesystem types. All of these commands have the same syntax and options. The commands can restore single files, directories, or entire filesystems.

To restore an entire filesystem, you must restore the most recent backup tapes from *each* backup level: the most recent full dump (0), the most recent level 1 dump, and so on. You must restore each level in numerical order, beginning with level 0. `restore` places the files it retrieves in the current working directory. Therefore, to restore a filesystem as a whole, you may wish to create and mount a clean, empty filesystem, make the current working directory the directory in which this filesystem is mounted, and then use `restore` to read the backup tapes into this directory. Note that such `restore` operations will have the side effect of recreating deleted files.

After a full restore, you need to do a full (level 0) backup. The reason for this is that `dump` backs up files by their inode number internally, so the tape from which you just restored from won't match the inodes in the new filesystem since they were assigned sequentially as files were restored.

In general, the `restore` command has the following forms (similar to `dump`'s):

```
$ restore options-with-arguments [files-and-directories]
$ restore option-letters corresponding-arguments [files-and-directories]
```

where *files-and-directories* is a list of files and directories for `restore` to retrieve from the backup tape. If no files are listed, the entire tape will be restored.

In the first, newer form, the first item is the list of options to be used for this backup with their arguments immediately following the option letters in the normal way (e.g., `-f /dev/tape`). In the second, older form, *option-letters* is a list of argument letters for the desired options, and *corresponding-arguments* are the values associated with each argument, in the same order. This syntax is still the only one available under AIX and Solaris.

Most options to `restore` do not have any arguments. However, as with `dump`, it is important that any arguments appear in the same order as the options requiring them.

`restore` places the files that it retrieves in the current working directory. When a directory is selected for restoration, `restore` restores the directory and all the files within it, unless you have specified the `-h` option (described later in this section).

`restore`'s most important options are the following:

`-r`

*Read* and restore the entire tape. This is a very powerful command; it should be used only to restore an entire filesystem located on one or more tapes. The filesystem into which the tape is read should be newly created and completely empty. This option can also be used to restore a complete incremental dump on top of a newly restored filesystem. That is, after using the `-r` option to restore the most recent full dump, you use it again to restore successive incremental dumps until the filesystem has been completely restored.

`-x`
: *Extract* all files and directories listed and restore them in the current directory. Each filename to be extracted must be a complete pathname *relative* to the root directory of the filesystem being restored. For example, to restore the file */chem/pub/old/gold.dat* from a `dump` of the */chem* filesystem, you must specify the filename as *pub/old/gold.dat*. You should be in */chem* when you execute the `restore` command if you want the file to be restored to its original location.

`-t`
: *Type* the names of the listed files and directories if they appear on the backup tape. This option lets you find out whether a given file is on a particular tape more quickly than reading the entire tape. When used without a file list, it verifies that a `dump` tape is readable.

`-f` *`file`*
: The corresponding argument is the name of the file or device holding the dump. If this option is omitted, `restore` assumes that the dump tape is mounted on your default tape drive. Use a hyphen for *file* to specify standard input.

`-s` *`n`*
: The value *n* indicates which file on tape is to be used for the restore. For example, `-s 3` says to use the third tape file.

`-i`
: Enter *interactive* mode. This is almost always the most convenient method for restoring a small group of files. It is described in detail in the next section.

A typical usage of the `restore` command is:

```
# cd /home
# restore -x -f /dev/rmt1 chavez/mystuff others/myprogram
```

This restores the directory */home/chavez/mystuff* and the file called */home/others/myprogram* from a backup tape (assuming that */home* is the filesystem in the archive). The directories *chavez* and *others* are assumed to be in the current directory (and created if necessary), and the specified subdirectory and file are restored under them. These both originally resided within the */home* directory. Note, however, that the mount point name is not used in the `restore` command. The command must be executed from */home* to restore the files to their original locations.

On Solaris and HP-UX systems, the corresponding options would be:

```
xf /dev/rmt1 chavez/mystuff others/myprogram
```

`dump` and `restore` both save files independently of where the filesystem happens to be mounted at the time; that is, the pathnames used by these commands are relative to their position in their *own* filesystem, not in the overall system filesystem. This makes sense, because the filesystem could potentially be mounted anywhere in the overall directory tree, and files should still be able to be restored to their correct location relative to the current mount point for their filesystem.

If you need to restore some files that have been destroyed by accident, your most difficult problems will be determining which set of backup tapes contains these files and waiting for the system to read through one or more full backup tapes. If you do incremental backups, knowing when a file was last modified will help you to find the correct backup tape. Creating online table-of-contents files is also very useful (this topic is discussed later in this chapter).

### The restore utility's interactive mode

The interactive mode is entered with `restore`'s `-i` option. Once there, the contents of a tape can be scanned and files chosen for extraction. This mode's use is illustrated in this sample session:

```
$ restore -i -f /dev/rmt1          Initiate restore's interactive mode.
restore > help
Available commands are:
   ls [arg] - list directory
   cd arg - change directory
   add [arg] - add `arg' to list of files to be extracted
   delete [arg] - delete `arg' from list of files to be extracted
   extract - extract requested files
...
If no `arg' is supplied, the current directory is used
restore > ls                       List directory on tape.
chavez/    harvey/     /ng
restore > cd chavez/vp             Change tape current directory.
restore > ls
v_a.c      v_a1.c      v_b3.c      v_d23.c      v_early
restore > add v_a1.c               Select (mark) files to be restored.
restore > add v_early
restore > ls
v_a.c     *v_a1.c      v_b3.c      v_d23.c     *v_early
restore > delete v_early           Remove a file from the extract list..
restore > extract                  Write selected files to current directory.
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1           Tape number if known.
set owner/mode for '.'? [yn] n     Don't change ./'s ownership or protection.
restore > quit                     End the restore interactive session.
```

The final prompt from `restore` asks whether to change the ownership and protection of the current directory to match that of the root directory on the tape. Answer yes only if you are restoring an entire filesystem.



**Combining Several Backups onto a Single Tape**

If you want to place several archives onto the same tape, all you need to do is rewind the tape (if necessary) before writing the first archive and then use a nonrewinding device for all subsequent backup operations.

To retrieve files from a multiarchive tape, you must position the tape at the proper location before issuing the restoration command. `restore` can do this automatically using its `-s` option, which takes the tape file number you want to use as its argument.

For all other backup types, position the tape with the `mt` command. For example, the following commands position the tape just after the second archive on the tape:

```
$ mt -f /dev/rmt0 rewind      If necessary
$ mt -f /dev/nrmt0 fsf 2
```

Again, you will need to use the nonrewinding form of the tape device; otherwise, the tape will be rewound to the beginning after positioning. Once at the desired point, you can write an additional backup archive to the tape or perform a restore operation using the next archive on the tape, as appropriate.

### The HP-UX frecover utility

The HP-UX `frecover` utility restores files archived by `fbackup`, using a very similar syntax. For example, the first of the following commands restores the */chem/fullerenes* subdirectory tree:

```
# frecover -x -i /chem/fullerenes
# frecover -r -f /dev/rmt/1m
```

The second command restores all files on the tape in drive 1. `frecover` also accepts the `-i`, `-e`, and `-g` options. Other useful options include the following:

- `-X` and `-F` restore all retrieved files relative to the current directory (converting absolute pathnames to relative ones) or into the current directory (stripping off all paths), respectively.
- `-o` says to overwrite files on disk that are newer than the file in the backup set.
- `-N` says to read the backup media without restoring any files. It is useful for verifying the integrity of a backup and for creating table-of-contents files.

## Moving Data Between Systems

In general, `tar`, `cpio`, and `dump` write archives that are readable on many different computer systems. However, sometimes you will run into problems reading a tape on a system other than the one on which it was written. There are four major causes for such problems:

*Block size differences*
: The simplest cause of tape reading problems is a difference in the block size with which the archive was written and the block size expected by the drive on which

you're trying to read it. Some tape drives assume specific fixed block sizes. You can specify the block size to backup and restore utilities (`-b` is often the relevant option), and on many systems you can set the characteristics of the drive itself. The most commonly used block sizes are 512 and 1024.

*Archive format incompatibilities*
: The backup utilities provided by early versions of Unix differed from those in use today, so very old computer systems may not be able to read tapes written on a current machine. The modern versions of most utilities include backward-compatibility options that allow you to write tapes in the old format if you need to read them on an ancient system.

*Byte order differences*
: Whether a computer system is *big endian* or *little endian* determines how it interprets the individual bytes within larger data units, such as words. Big-endian systems consider the byte with the lowest address as the most significant; little-endian systems consider it to be the least significant. Tape archives, like all other data on a computer system, reflect this fundamental attribute of the hardware. When you want to read a tape produced by a computer of one type on a different computer of the other type, you'll need to swap the bytes before utilities like `tar` can make sense of the archive.

  For example, you could use this AIX command to list the contents of a tape written on an IRIX system:

```
$ dd if=/dev/rmt1 conv=swab | tar tvf -
```

  The `dd` command reads the tape file and swaps the bytes, passing the converted archive to the `tar` command, which lists the archive it finds on standard input. You could construct the equivalent reversed pipe to produce a byte-swapped archive on tape.

*Compressed archives*
: If you write a tape on a drive that performs automatic data compression, you won't be able to read it on a drive that lacks this feature. In order to write tapes that will be readable on drives without compression, you'll need to specify the noncompressing special file to the backup utility (refer to the discussion of special files earlier in this chapter, as well as the relevant manual pages for your systems, for details).

## Making Table of Contents Files

It is often convenient to have online listings of the contents of system backup tapes. For one thing, they make it much easier to figure out which tape has the file you need to restore, especially when multiple levels of incremental backups are in use. It is quite easy to create such files at the time the backup is performed.

If you're using tar or cpio for backup, you can take advantage of the -v option to create a listing of the tape's contents as it is written, as in these examples:

```
$ today='date +%d%b%Y'
$ tar -cv /home > /backup/home_full_$today.TOC
                    or
$ tar -cv /home | tee /backup/home_full_$today.TOC
```

Both tar commands archive the contents of */home*, generating a long, directory-like listing as it does so and saving it to a file with a name like */backup/home_full_21mar1995.TOC*. The second command also displays the same output on the screen.

cpio sends the file list to standard error, so it must be captured slightly differently:

```
$ toc='date +/backup/home_full_%d%b%y.TOC'
$ find /home -print |  cpio -ov > /dev/rmt0 2> $toc
```

If you want to use the C shell, the commands are a little different:

```
% set toc='date +/backup/home_full_%d%b%y.TOC'
% (find /home -print | cpio -ov > /dev/rmt0) >& $toc
```

The file lists produced by cpio commands like these contain only the pathnames of the files in the archive. If you want a more detailed listing, you can generate it with a second cpio command or a more complex pipe leading up to the cpio backup command:

```
$ cpio -itv < /dev/rmt0 > $toc
$ find /home | cpio -o | tee /dev/rmt0 | cpio -t -i -v > $toc
```

The first command lists the files in the archive on tape. The second command avoids having to reread the backup tape by using the find command to generate a list of files, which cpio makes into an archive. This archive is then sent both the the tape drive and to another cpio command. The latter lists the archive contents and writes it to the specified table-of-contents file.

Making a table of contents file for a dump tape requires a subsequent restore command. For example, here is a script that performs a backup with dump and then creates a table-of-contents file with restore:

```
#!/bin/csh
# bkup+toc - perform dump and verify tape/make TOC file
# $1 = filesystem
# $2 = dump level (default=0)
#
if ($#argv < 1) then
  echo "do_backup: filesystem [dump-level]"
  exit 1
endif

set lev=0
if ("$2" != "") set lev=$2
dump -${lev} -u -f /dev/rmt1 $1
if ($status) then
```

```
    echo "do_backup: dump failed"
    exit 1
  endif
  restore -t -v -f /dev/rmt1 > /backup/`date +$1:t_%m-%d-%Y.$lev`
```

This script runs the `dump` command on the filesystem given as its first argument, using the backup level specified as its second argument (or level 0 by default). If the `dump` command exits normally, the `restore` command is used to verify the backup and write its contents to a file. The file's name contains the disk name and the month, day, and year when the backup was done, and its extension is the backup level: e.g., *chem_06-24-2001.2* would be the filename for a level 2 backup of */chem* made on June 24, 2001.

On an HP-UX system, you can use this `frecover` command to create a table-of-content file:

```
# frecover -r -Nv -f /dev/rmt/0m > $toc
```

# Network Backup Systems

So far, we've considered only backups and restores of disks on a local computer system. However, many organizations need to take a more unified and comprehensive approach to their total backup needs. We will consider various available solutions for this problem in this section.

## Remote Backups and Restores

The simplest way to move beyond the single-system backup view is to consider remote backup and restores. It is very common to want to perform a backup over the network. The reasons are varied: your system may not have a tape drive at all since not all systems come with one by default any more, there may be a better (faster, higher capacity) tape drive on another system, and so on.

Most versions of `dump` and `restore` can perform network-based operations (Tru64 requires you to use the separate `rdump` and `rrestore` commands). This is accomplished by specifying a device name of the form *host:local_device* as an argument to the `-f` option. The hostname may also optionally be preceded by a username and at-sign; for example, `-f chavez@hamlet:/dev/rmt1` performs the operation on device */dev/rmt1* on host *hamlet* as user *chavez*.

This capability uses the same network services as the `rsh` and `rcp` commands. Remote backup facilities depend on the daemon */usr/sbin/rmt* (which is often linked to */etc/rmt*).* To be allowed access on the remote system, there needs to be a *.rhosts* in its root directory, containing at least the name of the (local) host from which the data will come. This file must be owned by *root*, and its mode must not allow any

* On a few older systems, you'll need to create the link yourself.

access by group or other users (for example, 400). This mechanism has the mechanism's usual negative security implications (see "Network Security" in Chapter 7).



Some versions of the `tar` command can also use the `rmt` remote tape facility.

The HP-UX `fbackup` and `frestore` utilities accept remote tape drives as arguments to the normal `-f` option. For example:

```
# fbackup -0u -f backuphost:/dev/rmt/1m -i /chem
```

## The Amanda Facility

Amanda is the Advanced Maryland Automated Network Disk Archiver. It was developed at the University of Maryland (James da Silva was the initial author). The project's home page is *http://www.amanda.org*, where it can be obtained free of charge. This section provides an overview of Amanda. Consult Chapter 4 of *Unix Backup and Recovery* for a very detailed discussion of all of Amanda's features (this chapter is also available on the Amanda home page).

### About Amanda

Amanda allows backups from a network of clients to be sent to a single designated backup server. The package operates by functioning as a wrapper around native backup software like GNU `tar` and `dump`. It can also back up files from Windows clients via the Samba facility (`smbtar`). It has a number of nice features:

- It uses its own network protocols and thus does not suffer from the security problems inherent in the `rmt` approach.
- It supports many common tape and other backup devices (including stackers and jukeboxes).
- It can perform full and incremental backups and decide the backup level automatically based on specified configuration parameters.
- It can take advantage of hardware compression features, or it can compress archives prior to writing them to tape (or other media) when the former is not available. Software compression may be performed either by the main server or by the client system.
- It provides excellent protection against accidental media overwriting.
- It can use holding disks as intermediate storage for backup archives to maximize tape write performance and to ensure that data is backed up in spite of tape errors (allowing the backup set to be written to backup media at a later time).
- It can use Kerberos-based authentication in addition to providing its own authentication scheme. Kerberos encryption can also be used to protect the data as it is transmitted across the network.

At present, Amanda does have a couple of annoying limitations:

- It cannot split a backup archive across multiple tapes. When it encounters an end-of-tape mark while saving a backup archive, it begins writing the archive from the beginning on the next tape.
- It cannot produce individual backup archives larger than a single tape. This is a consequence of the first limitation.
- Only a single backup server is supported.

### How Amanda works

Amanda uses a combination of full and incremental backups to save all of the data for which it is responsible, using the smallest possible daily backup set that can do so. Its scheme first computes the total amount of data to be backed up. It uses this total, along with a couple of parameters defined by the system administrator, to figure out what to do in the current run. These are the key parameters:

*The number of runs in a backup cycle*
: At a rate of one Amanda run per day, this corresponds to the desired number of days between full backups.

*The percentage of data that changes between Amanda runs*
: In the single run per day case, this is the percentage of the data that changes each day.

Amanda's overall strategy is twofold: to complete a full backup of the data within each cycle and to be sure that all changed data has been backed up between full dumps. The traditional method of doing this is to perform the full backup followed by incrementals on the days between them. Amanda operates differently.

Each run (night), Amanda performs a full backup of part of the data, specifically, the fraction that is required to back up the entire data set in the course of a complete backup cycle. For example, if the cycle is 7 days long (with one run per day), 1/7 of the data must be backed up each day to complete a full backup in 7 days. In addition to this "partial" full backup, Amanda also performs incremental backups for all data that has changed since its own last full backup.

Figure 11-1 illustrates an Amanda backup cycle lasting 4 days, in which 15% of the data changes from day to day. The box at the top of the figure stands for the complete set of data for which Amanda is responsible; we have divided it into four segments to represent the part of the data that gets a full backup at the same time.

The contents of the nightly backups are shown at the bottom of the figure. The first three days represent a start-up period. On the first night, the first quarter of the data is fully backed up. On the second night, the second quarter is fully backed up, and the 15% of the data from the previous night that changed during day 2 is also saved. On day 3, the third quarter of the total data is fully backed up, as well as the changed 15% of day 2's backup. In addition, 15% of the portion backed up on the first night

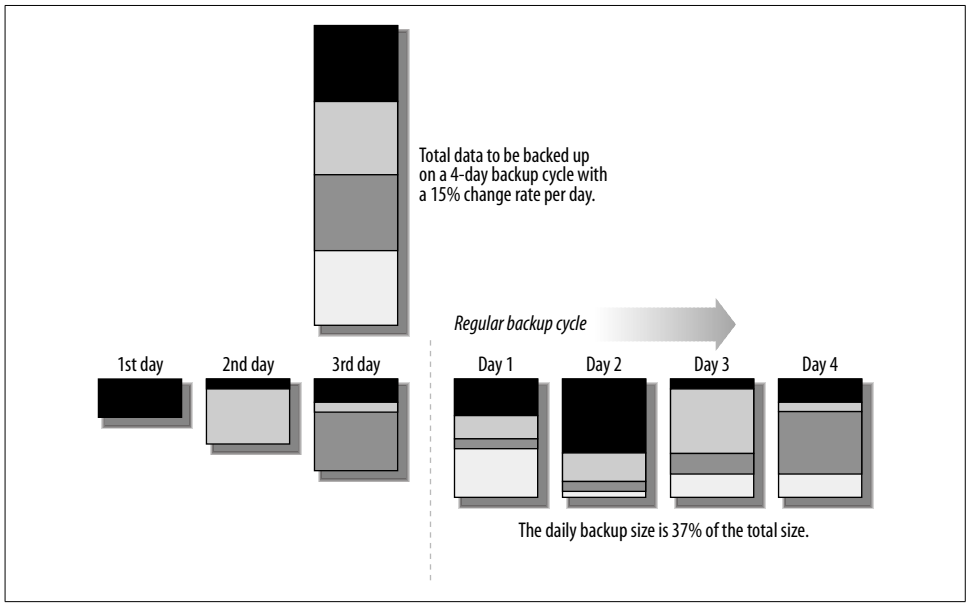



*Figure 11-1. The Amanda backup scheme*

is written for each of the intervening nights since its full backup: in other words, 30% of that quarter of the total data.

By day 4, the normal schedule is in force. Each night, one quarter of the total data is backed up in full, and incrementals are performed for each of the other quarters as appropriate to the time that has passed since their last full backup.



This example uses only first-level incremental backups. In actual practice, Amanda uses multiple levels of incremental backups to minimize backup storage requirements.

To restore files from an Amanda backup, you may need one complete cycle of media.

Let's now consider a numeric example. Suppose we have 100 GB of data that we need to back up. Table 11-3 illustrates four Amanda backup schedules based on differing cycle lengths and per-day change percentages.

*Table 11-3. Sample Amanda backup sizes (total data=100 GB)*

| | 3-day cycle 10% change | 5-day cycle 10% change | 7-day cycle 10% change | 7-day cycle 15% change |
|---|---|---|---|---|
| **Full portion** | 33.3 | 20.0 | 14.3 | 14.3 |
| **1st previous day** | 3.4 | 2.0 | 1.4 | 2.2 |
| **2nd previous day** | 6.8 | 4.0 | 2.8 | 4.4 |
| **3rd previous day** | | 6.0 | 4.2 | 6.6 |

*Table 11-3. Sample Amanda backup sizes (total data=100 GB) (continued)*

| | 3-day cycle 10% change | 5-day cycle 10% change | 7-day cycle 10% change | 7-day cycle 15% change |
|---|---|---|---|---|
| **4th previous day** | | 8.0 | 5.6 | 8.8 |
| **5th previous day** | | | 7.0 | 11.0 |
| **6th previous day** | | | 8.4 | 13.2 |
| **Daily size (GB)** | **43.5** | **40.0** | **43.7** | **60.5** |

The table columns illustrate the data that would comprise each daily backup, breaking it down by the full backup portion and the incremental data from each previous full backup within the cycle.

Note that Amanda computes what should be backed up every time it is run, so it is not as static as the preceding examples suggest, but the examples nevertheless provide a general picture of how the facility operates.

In the next section, we consider how the backup size depends on the backup cycle more formally, including some expressions that can be used to decide on an appropriate backup cycle for specific conditions.



**Estimating the Daily Change Rate**

You can use the `find` command to help estimate the daily change rate:

```
$ find dir -newer /var/adm/yesterday -ls | \
  awk '{sum+=$7}; END {print "diff =",sum}'
```

Repeat the command as needed to cover all the data to be backed up. Use `touch` to update the time for the file */var/adm/yesterday* after all the `find` commands are run.

Then, divide this value by the total used space (e.g., taken from `df` output). Repeat the process for several days or weeks to determine an average rate.

### Doing the math

Next, we consider some expressions that can be used to compute starting parameters for Amanda (which can be fine-tuned over time, based on actual use). If this sort of mathematical analysis is of no interest to you, just skip this section.

We will use the following variables:

$T$ = total amount of data
$p$ = percentage change between runs (in decimal form: e.g. 12%=0.12)
$n$ = number of runs in a complete cycle (often days)
$S$ = amount of data that must be backed up every run (day)
$F$ = fraction of the total data that must be backed up every run (day): $S/T$

To compute per-run amount of data that must be backed up, use this expression for $S$:

$$S = \frac{T}{n} + \frac{Tp(n-1)}{2}$$

For example, 70 GB of data that changes by 10% per day using a 1 week backup cycle requires that 31 GB be backed up every night ($70/7 + 70 \times 0.1 \times 6/2 = 10 + 42/2 = 10 + 21 = 31$). If 31 GB is larger than the maximum capacity that you have in the available time, you'll need to adjust the other parameters (see below).

Alternatively, if you have a fixed amount of backup capacity per run, you can figure out the required cycle length. Refer to the discussion of capacity planning earlier in this chapter for information on determining how much capacity you have.

To compute $n$ for a given nightly capacity, use this expression:

$$n = \frac{x \pm \sqrt{x^2 - 2p}}{p}$$

where

$$x = \left(\frac{p}{2} + \frac{S}{T}\right)$$

We have introduced the variable $x$ to make the expression for $n$ simpler. Suppose that you have a nightly backup capacity of 40 GB for the same scenario (70 GB total data, changing at 10% per day). Then $x = 0.1/2 + 40/70 = 0.05 + 0.57 = 0.62$. We can now compute $n$: $(0.62 \pm \sqrt{0.38 - 0.2}) / 0.1 = (0.62 \pm \sqrt{0.18}) / 0.1 = (0.62 \pm 0.42) / 0.1 = 6.2 \pm 4.2$.

This calculation yields solutions of 2 and 11 (rounding to integers). We can either do full backups of about half the data every night or use a much longer 11-day cycle and still be able to get the backups all done. Note that these values take maximum advantage of the available capacity.

Now suppose that you have a nightly backup capacity of only 20 GB for the same scenario (70 GB total data, changing at 10% per day). Then $x = 0.1/2 + 40/70 = 0.05 + 0.29 = 0.34$. We can now compute $n$: $(0.34 \pm \sqrt{0.12 - 0.2}) / 0.1$. The square-root term is now imaginary (since 0.12–0.20 is negative), indicating that this proposed configuration will not work in practice.* The available capacity is simply too small.

In general, you can compute the minimum per-run capacity for a given per-run percentage change ($p$) with this expression (which introduces $F$ as the fraction of the total data that must be backed up):

$$F_{minimum\ for\ fixed\ p} = 2\sqrt{\frac{p}{2} - \frac{p}{2}} \quad (\text{where } F = \frac{S}{T}); \quad \therefore S_{minimum} = FT$$

* Mathematically, there are no real solutions to the underlying quadratic equation.

$F$ indicates the fraction of that data that must be backed up each run in order for the system to succeed. So, in our case of a 10% change rate, $F = 2 \times \sqrt{0.1/2} - (0.1/2) = 2 \times \sqrt{0.05} - 0.05 = 2 \times 0.22 - 0.05 = 0.44 - 0.05 = 0.39 \approx 40\%$. Note that this expression is independent of $T$ (the total backup data); whenever the data changes by about 10% per run, you must be able to back up at least 40% of the total data every run for success. In our case, this corresponds to a minimum nightly capacity of $0.4 \times 70 = 29$ GB.

Alternatively, you can compute the run cycle $n$ that is required to minimize $F$ (and thus $S$) for a given value of $p$ with this expression:*

$$n_{minimum\ S} = \sqrt{\frac{2}{p}}$$

In our case, the cycle period which minimizes the amount of data to be backed up is $\sqrt{2/0.01} = \sqrt{20} = 4.47 \approx 5$ days. Again, this value is independent of the amount of data. In our case, when the data is changing by 10% per day, a cycle time of 5 days will minimize the amount of data that must be backed up every night. This is the most efficient cycle length with the minimum nightly backup capacity.

Thus, both the minimum time cycle and per-run fraction of data to back up are determined only by the rate at which the data is changing, and the actual per-run backup size for a given amount of total backup data can be easily computed from them. Thus, having an accurate estimate for $p$ is vital to rational planning.



This discussion ignores compression in analyzing backup procedures. If your tape drive can compress data, or if you decide to compress it with software before writing it to tape, you will need to take the expected compression factor into account in your computations.

### Configuring Amanda

Building and installing Amanda is generally straightforward, and the process is well-documented, so we will not consider it here.

The Amanda system includes the following components:

- Client programs, of which `amandad` is the most important. This daemon communicates with the Amanda server during backup runs, calling other client programs as appropriate: `selfcheck` (verify local Amanda configuration), `sendsize` (estimate backup size), `sendbackup` (perform backup operations), and `amcheck` (verify Amanda setup). These programs are part of the Amanda client system; on the Amanda server, these programs are found with the package's other helper programs, in */usr/local/lib/amanda* or */usr/lib/amanda*.

* Mathematically, the value of $n$ where $\partial F / \partial n = 0$. In this specific example, the mathematical region around the minimum is quite flat.

- Server programs to perform the various phases of the actual backup operations. The `amdump` program is the one that initiates an Amanda run, and it is usually run periodically from `cron`. It controls a number of other programs, including `planner` (determine what to backup), `driver` (interface to device), `dumper` (communicate with client `amandad` processes), `taper` (write data to media), and `amreport` (prepare report for an Amanda run).
- Administrative utilities to perform related tasks. They include `amcheck` (verify Amanda configuration is valid and the facility is ready to run), `amlabel` (prepare media for use with Amanda), `amcleanup` (clean up after an aborted run or system crash), `amflush` (force data from the holding area to backup media), and `amadmin` (perform various administrative functions).
- Configuration files that specify Amanda operations, such as what to back up and how often to do so, as well as the locations and characteristics of the tape device. These files are *amanda.conf* and *disklist*, and they reside in a subdirectory of the main Amanda directory (canonically, this location is */usr/local/etc/amanda*, but it can be */etc/amanda* when the package is preinstalled). A typical name is *Daily*. Each subdirectory corresponds to an Amanda "configuration," a distinct set of settings and options referred to by the directory name.
- The `amrestore` utility, which can be used to restore data from Amanda backups. In addition, the `amrecover` utility supports interactive file restoration. It relies on a couple of daemons to do its job: `amindexd` and `amidxtaped`.

**Setting up an Amanda client.** Once you have installed the Amanda software on a client system, there are a few additional steps to take. First, you must add entries to the */etc/inetd.conf* and */etc/services* files to enable support for the Amanda network services:

```
/etc/services:
amanda     10080/udp

/etc/inetd.conf:
amanda  dgram   udp   wait   amanda   /path/amandad   amandad
```

The Amanda daemon runs as user *amanda* in this example; you should use whatever username you specified when you installed the Amanda software.

In addition, you'll need to ensure that all the data that you want to be backed up is readable by the Amanda user and group. Similarly, the file */etc/dumpdates* must exist and be writeable by the Amanda group.

Finally, you must set up the authorization scheme that `amandad` will use. This is usually selected at compile time. You may use normal *.rhosts*-based authentication, Kerberos authentication (see below) or a separate *.amandahosts* (the default mechanism). The *.amandahosts* file is similar to a *.rhosts* file, but it applies only to the Amanda facility and so does not carry the same level of risk. Consult the Amanda documentation for full information about authentication options.

**Selecting an Amanda server.** Selecting an appropriate system as the Amanda server is crucial to good performance. You should keep the following items in mind:

- The system should have the best tape drives (or other backup devices) possible.
- The system should have sufficient network bandwidth for the estimated data flow.
- The system should have sufficient disk space for the holding area. A good size is at least twice the size of the largest per-run dump size.
- If the server will be performing software compression on the data, a fast CPU is necessary.
- Large amounts of memory will have little effect on backup performance, so there is no reason to overconfigure the system with memory.

**Setting up the Amanda server.** There are several steps necessary to configure the Amanda server once the software is installed. First of all, you must add entries to the same network configuration files as those for Amanda clients:

*/etc/services:*
```
amanda        10080/udp
amandaidx     10082/tcp
amidxtape     10083/tcp
```

*/etc/inetd.conf*:
```
amandaidx  stream  tcp  nowait  amanda /path/amindexd   amindexd
amidxtape  stream  tcp  nowait  amanda /path/amidxtaped amidxtaped
```

Next, you must configure Amanda by creating the required configuration files. Create a new subdirectory under *etc/amanda* in the top-level Amanda directory (i.e., */usr/local* or */*), if necessary. We will use *Daily* as our example. Then, create and modify *amanda.conf* and *disklist* configuration files in this subdirectory (the Amanda package contains example files that can be used as a starting point).

We will begin with *amanda.conf* and consider its contents in groups of related entries. We will examine an annotated sample *amanda.conf* file.

The initial entries in the file typically specify information about the local site and locations of important files:

```
org "ahania.com"                Organization name for reports.
mailto "amanda-rep"             Mail reports to this user.
dumpuser "amanda"               Amanda user account.
printer "tlabels"               Printer for tape labels.
logdir "/var/log/amanda"        Put log files here.
indexdir "/var/adm/amindex"     Store backup set index data here.
```

The next few entries specify the basic parameters for the backup procedure:

```
# fundamental parameters
dumpcycle 7 days      Length of the backup cycle (default=10 days).
runspercycle 5        Amanda runs per cycle (if < 1/day).
```

```
# network-related resource settings
netusage 400 kps      Maximum network bandwidth (default=300).
inparallel 20         Max. simultaneous backups (default=10).
ctimeout 120          Client timeout period (default=30 seconds).

# incremental level bump parameters
bumpsize 20 mb        Min. savings for level 2 incrs. (default=10).
bumpdays 1            Required # days at each level (default=2).
bumpmult 2            Multiply bumpsize by this for each higher incremental level (default=1.5).
```

The incremental bump level parameters specify when Amanda should increase the incremental backup level in order to make the backup set size smaller. Using these settings, Amanda will switch from level 1 incrementals to level 2 incrementals whenever it will save at least 20 MB of space. The multiplication factor has the effect of requiring additional savings to move to each higher incremental level. The threshold for each level is this factor times the saving required for the previous level, i.e., 40 for levels 2 to 3, 80 for levels 3 to 4, and so on. This strategy is designed to ensure that the added complexity of multiple levels of incremental backups also bring significant savings in the size of the backup set.

These next entries specify information about the tape drive and media to use:

```
# number of tapes in use          Set to at least # tapes required for one full cycle
tapecycle 25                        plus a few spares (default=15).
labelstr "Daily[0-9][0-9]*"       Format of the table labels (regular expression).

tapedev "/dev/rmt/0"
tapetype "DLT"

#changerdev "/dev/whatever"
#tpchanger "script-path"          Script to change to next tape (supplied).
#runtapes 4                        Maximum number of tapes per run.
```

The first two entries specify the number of tapes in use and the pattern used by their electronic labels. Note that tapes must be prepared with `amlabel` prior to use (discussed below).

The next two entries specify the location of the tape drive and its type. The final three entries are used with tape changers and are commented out in this example. Only one of *tapedev* and *tpchanger* must be used.

Tape types are defined elsewhere in the configuration file with stanzas like this:

```
define tapetype DLT {
    comment "DLT with 10 GB tapes"
    length 12500 mb      Tape capacity (takes compression into account).
    speed 1536 kps       Drive speed.
    lbl-templ "file"     PostScript template file for printed labels.
}
```

The example configuration file includes many defined tape types. The *length* and *speed* parameters are used only for estimation purposes (e.g., how many tapes will be required). When performing the actual data transfer to tape, Amanda will keep writing until it encounters an end-of-tape mark.

The following entry and *holdingdisk* stanza defines a disk holding area:

```
# When media is unavailable, save this % of holding space
# for degraded-mode incremental backups.
reserve 50                Default is 100%.

holdingdisk amhold0 {     Name is amhold0.
   comment "Primary holding disk"
   directory "/scratch/amanda"
# amount of space to use (+) or save (-); 0=use all (default)
   use -2 Gb              Always leave this much space.
}
```

More than one holding disk may be defined.

The final task to be done in the configuration file is to define various dump types: generalized backup actions having specific characteristics (but independent of the data to be backed up). Here is an example for the *normal* backup type (you can choose any names you like):

```
define dumptype normal {
   comment "Ordinary backup"
   holdingdisk yes      Use a holding disk.
   index yes            Maintain index info on contents.
   program "DUMP"       Backup command.
   priority medium      Specify backup relative priority.
#  use 24-hour clock without punctuation
   starttime 2000       Don't begin backup before this time (8 P.M. here).
}
```

This dump type uses a holding disk, creates an index for the backup set contents for interactive restoration and uses the `dump` program to perform the actual backup. It runs at medium priority compared to other backups (the possibilities are *low* (0), *medium* (1), *high* (2) and an arbitrary integer, with higher numbers meaning the backup will be performed sooner). Backups using this method will not begin before 8 pm regardless of when the `amdump` command is issued.

Amanda provides several pre-defined dump types in the example *amanda.conf* file which can be used or customized as desired.

Here are some other parameters that are useful in dump type definitions:

```
program "GNUTAR"          Use the GNU tar program for backups.
                          This is also the value to use for Samba backups.
exclude ".exclude"        GNU  tar exclusion file (located in top-level
                          of the filesystem to be backed up).
compress server "fast"    Use software compression on server using the
                          fastest compression method. Other keywords are
                          "client" and "best".
auth "krb4"               Use Kerberos 4 user authentication.
kencrypt yes              Encrypt transmitted data.
ignore yes                Do not run this backup type.
```

Amanda's *disklist* configuration file specifies the actual filesystems to be backed up. Here are some sample entries:

```
# host     partition      dumptype    spindle
hamlet      sd1a           normal        -1
hamlet      sd2a           normal        -1
dalton      /chem          srv_comp      -1
leda        //leda/e       samba         -1   # Win2K system
astarte     /data1         normal         1
astarte     /data2         normal         1
astarte     /home          normal         2   # dump all alone
```

The columns in this file hold the hostname, disk partition (specified by file in */dev*, full special file name, or mount point), the dump type, and a spindle parameter. The latter serves to control which backups can be done at the same time on a host. A value of -1 says to ignore this parameter. Other values define backup groups within a host; Amanda will only run backups from the same group in parallel. For example, on host *astarte*, the */home* filesystem must be backed up separately from the other two (the latter may be backed up simultaneously if Amanda so wishes).

There are a few final steps that are needed to complete the Amanda server setup:

- Prepare media with the `amlabel` command. For example, the following command will prepare a tape labeled "DAILY05" for use with the Amanda configuration named *Daily*:

  ```
  $ amlabel Daily DAILY05
  ```

  Similarly, the following command will prepare the tape in slot 5 of the associated tape device as "CHEM101" for use with the *Chem* configuration:

  ```
  $ amlabel Chem CHEM101 slot 5
  ```

- Use the `amcheck` command to check and verify the Amanda configuration.
- Create a `cron` job for the Amanda user to run the `amdump` command on a regular basis (e.g., nightly). This command takes the desired configuration as its argument.

Amanda expects the proper tape to be in the tape drive when the backup process begins. You can determine the next tape needed for the *Daily* configuration by running the following command:

```
# amadmin Daily tape
```

The Amanda system will need some ongoing administration, including tuning and cleanup. The latter is accomplished via the `amflush` and `amcleanup` commands. `amflush` is used to force the data in the holding disk to backup media, and it is typically required after a media failure occurs during an Amanda run. In such cases, the backup data is still written to the holding disk. The `amcleanup` command needs to be run after an Amanda run aborts or after a system crash.

Finally, you can temporarily disable an Amanda configuration by creating a file named *hold* in the corresponding subdirectory. While this file exists, the Amanda

system will pause. This can be used to keep the configuration information intact in the event of a hardware failure on the backup device or a device being temporarily needed for another task.

### Amanda reports and logs

The Amanda system produces a report for each backup run and sends it by electronic mail to the user specified in the *amanda.conf* configuration file. The reports are quite detailed and contain the following sections:

- The dump date and time and estimated media requirements:

```
These dumps were to tape DAILY05.
Tonight's dumps should go onto one tape: DAILY05.
```

- A summary of errors and other aberrations encountered during the run:

```
FAILURE AND STRANGE DUMP SUMMARY:
dalton.ahania.com /chem lev 0 FAILED [request ... timed out.]
```

  Host *dalton* was down so the backup failed.

- Statistics about the run, including data sizes and write rates (output has been shortened):

```
STATISTICS:
                          Total      Full      Daily
                        --------  --------   --------
Dump Time (hrs:min)        2:48      2:21       0:27
Output Size (meg)        9344.3    7221.1     2123.2
Original Size (meg)      9344.3    7221.1     2123.2
Avg Compressed Size (%)     --        --         --
Tape Used (%)              93.4      72.2       21.2
Filesystems Dumped           10         2          8
Avg Dump Rate (k/s)      1032.1    1322.7      398.1
Avg Tp Write Rate (k/s)  1234.6    1556.2     1123.8
```

- Additional information about some of the errors/aberrations, when available.
- Informative messages from the various subprograms called by `amdump`:

```
NOTES:
   planner: Adding new disk hamlet.ahania.com:/sda2
   taper: tape DAILY05 9568563 kb fm 1 [OK]
```

- A summary table listing the data that was backed up and related information:

```
DUMP SUMMARY:
                         DUMPER STATS              TAPER STATS
HOST   DISK  L ORIG-KB OUT-KB COMP% MMM:SS  KB/s MMM:SS  KB/s
-------------------------------------------------------------
hamlet sd1a  1   28255  28255   --    2:36 180.3  0:21 1321.1
hamlet sd2a  0  466523 466523   --   36:51 211.1  5:33 1400.8
dalton /chem 1  FAILED----------------------------------------
ada    /home 1   39781  39781   --    5:16 125.7  0:29 1356.7
...
```

You should examine the reports regularly, especially the sections related to errors and performance.

Amanda also produces log files for each run, *amdump.n*, and *log.date.n*, located in the designated log file directory. These are more verbose versions of the email report, and they can be helpful in tracking some sorts of problems.

#### Restoring files from an Amanda backup

Amanda provides the interactive `amrecover` utility for restoring files from Amanda backups. It requires that backup sets be indexed (using the *index yes* setting) and that the two indexing daemons mentioned previously be enabled. The utility must be run as *root* from the appropriate client system.

Here is a sample session:

```
# amrecover Daily
AMRECOVER Version 2.4.2. Contacting server on depot.ahania.com ...
...
Setting restore date to today (2001-08-12)
200 Working date set to 2001-08-14.
200 Config set to Daily.
200 Dump host set to astarte.ahania.com.
$CWD '/home/chavez/data' is on disk '/home' mounted at '/home'.
200 Disk set to /home.
amrecover> cd chavez/data
/home/chavez/data
amrecover> add jetfuel.jpg
Added /chavez/data/jetfuel.jpg
amrecover> extract
Extracting files using tape drive /dev/rmt0 on host depot...
The following tapes are needed: DAILY02
Restoring files into directory /home
Continue? [Y/n]: y
Load tape DAILY02 now
Continue? [Y/n]: y
warning: ./chavez: File exists
Warning: ./chavez/data: File exists
Set owner/mode for '.'? [yn]: n
amrecover> quit
```

In this case, the `amrecover` command is very similar to the standard `restore` command in its interactive mode.

The `amrestore` command can also be used to restore data from an Amanda backup. It is designed to restore entire images from Amanda tapes. See its manual page or the discussion in *Unix Backup and Restore* for details on its use.

## Commercial Backup Packages

There are several excellent commercial backup facilities available. An up-to-date list of current packages can be obtained from *http://www.storagemountain.com*. We won't consider any particular package here but, rather, briefly summarize the important features of a general-purpose backup package, which can potentially serve as criteria for comparing and evaluating any products your site is considering.

You should expect the following features from a high-end commercial backup software package suitable for medium-sized and larger networks:

- The ability to define backups sets as arbitrary lists of files that can be saved and reloaded into the utility as needed.
- A capability for defining and saving the characteristics and data comprising standard backup operations.
- A facility for exclusion lists, allowing you to create, save, and load lists of files and directories to exclude from a backup operation (including wildcard specifications).
- An automated backup scheduling facility accessed and controlled from within the backup utility itself.
- The ability to specify default settings for backup and restore operations.
- The ability to back up all important file types (e.g., device files, sparse files) and attributes (e.g., access control lists).
- The ability to back up open files or to skip them entirely without pausing (at your option).
- The ability to define and initiate remote backup and restore operations.
- Support for multiple backup servers.
- Support for high-end backup devices, such as stackers, jukeboxes, libraries and silos.
- Support for tape RAID devices, in which multiple physical tapes are combined into a single high-performance logical unit via parallel write operations.
- Support for non-tape backup devices, such as removable disks.
- The capability to perform multiple operations to distinct tape devices simultaneously.
- Support for multiplexed backup operations in which multiple data streams are backed up to a single tape device at the same time.
- Support for clients running all of the operating systems in use at your site.
- Compatibility with the standard backup utilities, which may be important to some sites (so that saved files can be restored to any system).
- Facilities for automatic archiving of inactive files to alternate online storage devices (for example, jukeboxes of optical disks) to conserve disk space and reduce backup requirements.
- Inclusion of some kind of database manager so that you (and the backup software) can perform queries to find the media needed to restore files.

See Chapter 5 of *Unix Backup and Recovery* for an extended discussion of commercial backup package features.

## Backing Up and Restoring the System Filesystems

This final section covers backing up and restoring the filesystem containing the operating system itself, including the case of a system disk failure. Recovering from such a disaster has come to be known as "bare metal recovery" in recent years. *Unix Backup and Restore* includes detailed chapters describing these techniques and procedures for several Unix varieties.

Filesystems containing operating system files such as */* and */usr* pose few problems when all you need to restore is the occasional accidentally deleted or otherwise lost file. When the file in question is an unmodified system file, you can usually restore it from the operating system installation media, provided you have it and that it is readable under normal system conditions. If either of these conditions is not true, you should do a full backup of all system filesystems from time to time.

Files that you modify in the system partitions should be backed up regularly. In Chapter 14, we looked at a script that saves all modified configuration and other files to a user filesystem, allowing them to be backed up regularly and automatically via the system backup procedures. Alternatively, the script could save them directly to the backup media (even to a diskette if the archive is small enough).

When system filesystems need to be completely restored (usually due to hardware problems), some special considerations come into play. There are often two distinct approaches that can be taken:

- Reinstalling from the original operating system installation tapes or CDs and then restoring files you have modified. This approach may also involve reconfiguring some subsystems.
- Booting from alternate media and then restoring the filesystems from full backups that you have made.

Which alternative is preferable depends a lot on the characteristics of your particular system: how many files have been customized and how widely they are spread across the various system filesystems, how much device and other reconfiguration needs to be redone, and similar considerations. If you have to restore multiple partitions, it is usually faster to reinstall the operating system from scratch unless unsaved data in another partition on the same disk will be lost using the standard installation procedures.

If you decide to take the second route, booting from alternate media and then restoring from a backup, you will need to make reliable full backups of the system whenever it changes significantly. Because you are depending on them for a system restoration in an emergency, these backups should be verified or even made in duplicate.

In either case, you will sometimes also need to consult records of the disk partitions and associated filesystem layouts, as well as the logical volume configuration, when a

logical volume manager is in use. This is vital when the system disk has been damaged and must be replaced to restore the system to its previous configuration. Be sure to keep records of this data (see below).

Here is a general procedure for restoring a key filesystem from a backup (some of the individual steps are discussed in detail Chapter 10):

- Boot off alternate media, either an installation tape or CD, or a special bootable diskette or tape (discussed in a bit). At this point, you will be running off an in-memory filesystem (RAM disk) or one based on the boot medium.
- Create device files for the disks, disk partitions, and/or tape drive that you will need to access, if necessary. They may already have been provided for you if you used a system utility to create the bootable tape or diskette.
- Prepare the hard disk as necessary. This may include formatting (rarely) or partitioning it. Be sure to do anything required to make the disk bootable.
- Create a new filesystem on the appropriate partition, if necessary.
- Mount the system filesystem *(/mnt* is the conventional location).
- Change the current directory to the mount point. Restore the files from the backup tape. Afterwards, change back to the root directory and dismount the restored filesystem.
- Repeat the process for any additional filesystem and then reboot the system.

There is one additional point to consider when using this approach—or planning to rely on it. The filesystem provided by emergency boot tapes or diskettes is very limited, and only a small subset of the normal system commands are available. You will need to verify that the restoration utility you need is available after booting from alternate media. For example, if the boot diskette provides only `cpio`, the backup of the root filesystem had better not be a `tar` archive or you will be in trouble. You should also ensure that any shared libraries needed by your desired utility are present. Be sure to verify this before the disaster occurs.

We will now look at this process on each of our Unix operating systems individually.

## AIX: mksysb and savevg

AIX provides the `mksysb` utility for creating bootable backup tapes of the actual live system, which are self-restoring in the event of a failure. It saves all of the filesystems in the root volume group, generally */*, */usr*, */var*, */home* (unless you've moved it), and */tmp*, plus any paging spaces in *rootvg*. `mksysb` is invoked as follows:

```
# mksysb -i /dev/rmt0
```

`mksysb` relies on a data file that records various system configuration information. It is updated by including `mksysb`'s `-i` option. Use the `-m` option instead if you wish to

restore the exact disk locations of the filesystems in the root volume group as well as their contents (`-m` says to save the logical volume maps as well as the other configuration information).

To restore the root volume group, boot from the `mksysb` tape and select the appropriate option from the resulting menu. The system will then be restored from the `mksysb` tape.

You can use a similar technique to clone a system from a `mksysb` tape made on a different system. If all the devices are identical, the only restriction is that you should not install a kernel from a multiprocessor system onto a single CPU system or vice versa.

When devices differ between the source and target system, a slightly modified technique is used. First, you boot off the install media, and then you select the option for restoring from a `mksysb` tape. In this mode, the operating system will automatically substitute drivers from the installation media when the ones on the `mksysb` tape are not correct for the target system. Note that this method will work only if the target system has the correct drives for accommodating both the `mksysb` and installation media simultaneously.

#### Restoring individual files from a mksysb tape

`mksysb` tapes can also serve as nonemergency backups of the root volume group. It is very easy to restore individual files from it. These tapes contain four distinct (tape) files, and the disk files from the root volume group are in the fourth file, which consists of a `restore` archive.

Thus, you could use the following command to restore the file */usr/bin/csh* and the subdirectory */etc/mf* from a `mksysb` backup tape:

```
# restore -s 4 -x -q -f /dev/rmt0 ./bin/csh ./etc/mf
```

The `-s` option indicates which tape file to use, and the `-q` option suppresses the initial prompt asking you to press the Enter key after you have mounted the first volume. Use `restore`'s `-T` option to list the contents of the archive.

#### Saving and restoring AIX user volume groups

The `savevg` command may be used to back up an entire user volume group, just as `mksysb` does for the root volume group. For example, the following command saves all of the files in the *chemvg* volume group to tape drive 1:

```
# savevg -i chemvg /dev/rmt1
```

The `-i` option creates the configuration file needed to save and restore the volume group; using `-m` instead also saves the logical volume maps, allowing their physical locations on disk to be reproduced.

savevg also has a -e option, which says to exclude the files and directories listed in the file */etc/exclude.vgname* from the save set.* Wildcards are not permitted in exclusion lists.

All of the logical volumes and filesystems and the files within them in a volume group can be restored from a savevg tape; the restvg utility performs this operation. For example, these commands restore the *chemvg* volume group we just saved:

```
# restvg -q -f /dev/rmt1
# restvg -q -s -f /dev/rmt1 hdisk4 hdisk5
```

The first command restores the volume group to its original disks, beginning immediately and without prompting for the first tape volume. The second command restores the structure and contents of the *chemvg* volume group to disks 4 and 5, shrinking all logical volumes to the minimum size necessary to hold the files within them (**-s**).

The tape made by savevg is a restore archive, so it is easy to extract individual files from it, as in this example:

```
# restore -f /dev/rmt1 -T -q
# restore -f /dev/rmt1 -x -q -d ./chem/src/h95
```

The first command lists the contents of the archive, and the second command restores the */chem/src/h95* subtree, creating any necessary subdirectories (-d).

## FreeBSD

FreeBSD provides a several options for restoring system files, but all of them require that you have a complete backup of the filesystem from which to restore.

In the event of a system disk or boot failure, you must boot from alternate media (CD-ROM or a boot floppy). Then select the Fixit option from the main menu that appears. At this point, you can choose to boot from the second installation CD (which will function as a live filesystem) or a fixit floppy, or you can start a limited shell. The first two options tend to be the most useful.

The fixit floppy is a limited FreeBSD operating system containing enough tools to restore from a backup. It includes support for the tar and restore commands and tape devices. You create a fixit floppy by mounting the first installation CD and using a command like this one:

```
# dd if=/cdrom/floppies/fixit of=/dev/rfd0c bs=36b
```

This floppy can be customized after creation for your specific needs.

In order to save the disk partition layouts on a FreeBSD system, use the fdisk -s and disklabel commands. Along with */etc/fstab*, this information will allow you to reconstruct the disk partitions and filesystem layout. The disklabel command can also be used to write a boot block to a replacement system disk.

* The mksysb command also recognizes -e, and its exclusion file is */etc/exclude.rootvg*.

## HP-UX: make_recovery

HP-UX provides the make_recovery facility for creating bootable recovery tapes as part of the Ignite-UX package (the utility is stored in */opt/ignite/bin*). A common method of using this utility is the following:

```
# make_recovery -p -A -d /dev/rmt/1mn
# emacs /var/opt/ignite/recovery/arch.include
# make_recovery -r -A -d /dev/rmt/1mn -C
```

First, we run the command in preview mode (-p). This command does not write any data to tape, but instead creates the file */var/opt/ignite/recovery/arch.include* which consists of a list of the items to be included. Here, we are choosing to save the entire root filesystem via -A; the default is to save only the subset of files that are part of the HP-UX operating system.

Once this command completes, we check the */var/opt/ignite/logs/makrec.log1* log file for any errors or warnings. If any are present, we must take any corrective action necessary and then rerun the first command.

Once any warnings are dealt with, the *arch.include* file can be edited to add or remove items, and then make_recovery can be run again in resume (-r) mode.* The -C option tells the command to update the stored data of the most recent make_recovery procedure.

This process must be repeated after each significant system change. The check_recovery command can be used to determine if make_recovery needs to be run.

Although these tapes are not intended to replace normal backups, it is possible to retrieve individual files from them. To do so, you must manually position the tape to the second file and then extract the desired items with tar:

```
# cd /
# mt -t /dev/rmt/1mn fsf 1
# tar xvf /dev/rmt/1m relative-pathname(s)
```

The file list should be specified as relative pathnames (e.g., *etc/hosts*, not */etc/hosts*).



The most recent versions of the HP Ignite-UX package also provide make_tape_recovery (creates tape recovery images on the client system itself and from the Ignite-UX server) and make_net_recovery (write a recovery image to the disk drive of the Ignite-UX server across the network). See the documentation for details

* In some cases, additional considerations apply when some system files reside outside the root volume group; see the manual page for details.

## Linux

On Linux systems, you can create a boot floppy of the current kernel with this command:

```
# dd if=/boot/file of=/dev/fd0
```

Simply copying the compressed kernel to diskette is all that is required, because the Linux kernel is structured so that it is the image of a bootable floppy disk (and it is loadable by either the DOS boot loader or `lilo`).

This procedure will enable you to boot your system should there be some problem booting from the hard disk. However, if your system disk is damaged and the root filesystem there is inaccessible, you will need a true recovery system to restore things. In such circumstances, you can boot using a *rescue disk*, which is created with the installation CD mounted with a command like this one:

```
# dd if=/cdrom/disks/rescue of=/dev/fd0 bs=18k
```

The rescue floppy contains tools needed to restore a saved backup, including tape devices and the `tar` command.

To record the disk partitioning information, use the `fdisk -l` command. Along with */etc/fstab*, this information will allow you to reconstruct the disk partitions and filesystem layout, and you can use `lilo` to create a boot block on a replacement system disk. Note that its `-r` option will prove very useful when the new root partition is mounted at some other point (e.g., */mnt*) within the rescue filesystem.



Recent versions of Red Hat Linux also provide a system rescue option when booting from the installation CD.

## Solaris

Solaris provides little in the way of tools for system backup and recovery. You should make full backups of the root filesystem. You can then boot off alternate media to create a minimally working system and restore from your backup.

The `prtvtoc` command along with */etc/checklist* will provide the information required to recreate the disk partitioning and filesystem layout scheme. You can use the `installboot` command to write a boot block to the system disk. Note that boot images are stored within the installed filesystem at */usr/platform/model/lib/fs/ufs/bootblk*, where *model* is a string corresponding to your specific Sun hardware model (e.g., *SUNW-Sun-Blade-100*).

## Tru64: btcreate

Tru64 provides the `btcreate` command for creating a bootable backup tape for the operating system. The tape consists of a bootable miniature operating system and a complete backup of the system files.

Running `btcreate` is very easy in that it will prompt you for all of the information that it requires. The default (suggested) answers are almost always correct. A restore from a `btcreate` tape will recreate the logical volume configuration from the original system in addition to restoring all of the system files.

On Tru64 systems, you can use the `disklabel -r` command to record disk partitioning information and recreate them if necessary.

Ci joint dans la version imprimée de ce cours, un article sur les sauvegardes.

Adresse web du document fourni : « `http://www.???.com/???/???` »

# Analysing Backup Systems

Elizabeth D. Zwicky
Information, Telecommunications, and Automation Division
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
zwicky@erg.sri.com

Abstract

Backup systems are absolutely essential ; every site must have some method of doing backups to avoid tragedies. They are also highly individual. The backup system that is well suited to one site may be completely impossible at another, because of differences in staffing, funding, or usage patterns on the machines. While descriptions of several backup systems are available, guidelines to choose between them are not. This paper attempts to present a practical and theoretical framework in which to analyse backup systems, so that you can choose between them, and develop your own if you have to.

## 1 Introduction

No collection of system administration disasters is complete without a backup disaster story. In fact, a reasonable collection has to have at least two, the punchline to one of which is "Backups ? What backups ?" and the other one of which is more subtle. These stories come about because backup systems are complex, and they're complex in particularly nasty ways.

This document is intended to help you avoid accumulating these stories through personal experience. It is not guaranteed to be comprehensive, although I do what I can ; the topic is very, very, large. I have a strong bias towards Berkeley `dump`, which is what I know best, although I try to keep matters as general as possible. This is not a cookbook that will tell you exactly how you should do backups.

## 2 Terminology

Backup systems must be distinguished from archive systems. A backup system is a precaution ; a method of replicating data in case a disaster, minor or major, occurs. An archive system is an extension ; a method of saving data off-line so as to increase storage capacity. The two purposes do overlap somewhat, and many people use their backup system as an archive system. On the other hand, backup systems don't make very good archive systems, because providing for rapid error recovery means keeping more data than an archive needs.

For convenience, I will use the concept of levels drawn from `dump`. A level 0 dump backs up every file on a file system ; each higher level backs up all the files that have changed since the last backup with a lower number. So, for instance, a level 1 backs up everything that has changed since the most recent level 0 ; a level 2 backs up everything that has changed since the most level 1 or level 0. `dump` itself has 9 levels available. There are also what many people call "true incrementals", which are incremental to the last backup, no matter what level it was done at.

A backup system consists of some sort of scheduler, using some sort of transport program to write data to some sort of media. If you currently run `dump` by hand every so often to floppy disks, you're the scheduler, `dump` is the transport program, and the floppies are the media. (You are probably also making a horrible mistake.) If you run `dd` from `cron` every two hours to copy filesystems from one disk to another, `cron` is the scheduler, `dd` is the transport program, and the disks are the media. (You are also probably making a horrible mistake.) Many commercial programs provide both a scheduler and a transport program.

## 3 Why Getting Backups Right Is Hard

Pretty much every UNIX site in the world does backups pretty much every day, so you would think that it would be a solved problem by now. It isn't. Bad backup systems are very persistent.

Bad backup systems are hard to detect. Most of those millions of backups are never even read, much less used. In one famous example at SRI, an operator decided that actually doing backups reduced the time available to him to do amusing things, and simply labelled blank tapes and hung them on the wall. He did this for nearly 6 months without incident, and was only discovered because a new employee insisted on getting back a deleted file (the existing employees had been trained not to ask). When a backup system that bad can last that long, imagine how much longer it can be before subtle problems are detected.

Even though those subtle problems may be hard to detect, they are nonetheless almost guaranteed to exist. A backup system is exposed to every possible kind of file system abuse at some point in its existence. It is exposed to most of them much more often than people intuitively believe. A one in a million chance sounds small, but several years ago I figured that we backed up more than a million files a week - by now, a one in a million event happens every few days at our site. (The situation is somewhat better if it's one in a million directories, instead of files, and is probably livable if it's one in a million runs of dump, which is a comforting 68 years.)

Furthermore, sites differ from each other in unexpected ways. It took me some time to figure out why it is that our site shows a large number of files changed per day - much larger than most published data. We run the MH mailer, which keeps each message in a separate file. It is not unusual for people who receive a lot of mail to modify 50 to 70 percent of the files in their directory every working day. This sort of behaviour would be pathological at other sites which have the same sorts of comupters and the same amount of disk space, but use other applications. This means that a system designed and tested in one place may have unsuspected flaws waiting for, for instance, a site that uses a graphical user interface file manager, and therefore has path names over 100 characters long.

Finally, filesystems and programs are rarely designed with ease of backup as a primary goal. Instead, backup is an add-on. This doesn't work very well. Standard UNIX file systems cannot be reliably backed up completely in a finite amount of time while users also have access to them, because there are no locking primitives that would allow you to insure that things don't change at critical moments. No amount of aftermarket fooling around can fix this ; all you can do is fix the filesystem design in one way or another.

All of this makes backup systems hard to build, hard to test, and very very easy to screw up critically without noticing for months or years.

# 4 Goals of a Backup System

There are 6 main goals to keep in mind when designing a backup system :

1. Minimize data loss in the event that a file or a disk is destroyed.
2. Minimize the resources used to restore data in the event that a file or a disk is destroyed.
3. Minimize the resources used for the backup system in normal operation.
4. Maximize the reliability of the system.
5. Maximize the flexibility of the system.
6. Maximize the predictability of the system.

# 5 Minimize Data Loss

This is the most obvious goal. If something gets lost, you want it back, as close to how it was the moment it disappeared as possible. The older the version that you can get back, the more work somebody has to do. Before you can reduce the amount of risk, however, you need to be able to figure out what the risk is.

## 5.1 Calculating Loss, No Complicating Factors

In the straightforward case where a file is lost, and you want to get it back immediately, the maximum amount of work that can be lost is the longest time between two backups of the same file system. So, for instance, if you do a backup every working day, starting sometime between 8 :00 am and 4 :00 pm, the worst possible case is that on Friday you wanted to go home early, so you did backups first thing in the morning. On Monday, however, you got swamped and were just about to do them at 4 :00 when the disk crashed, taking with it 80 hours of work, including almost two full working days. Doing the backups at the same time every day, on working days only, would reduce your maximum

loss to 72 hours. Whether or not that is acceptable to you probably depends on how much work your users do on weekends ; at a business where almost everybody works 9-5 Monday-Friday, that 72 hours includes only one working day, and that may be acceptable. If your usage on weekends is heavy, that's three working days, and you may need to consider weekend backups.

Users who are consulted about the amount of loss that is acceptable to them generally claim that the maximum they can live with is very low. Unfortunately, the amount of money and inconvenience they are willing to expend on a backup system is generally equally low. You need to be very careful to present both the costs of lost data and the costs of backups. The relevant managers, at least, should always know what the maximum loss is **before** any data is lost.

One working day of loss is a number that many sites feel they can live with most of the time. If it is not acceptable to you, you will need to figure out how to fit in extra backups. You will want to schedule them so as to catch relatively even amounts of work (not relatively even amounts of time). At most sites, backing up at 8 :00 am and 8 :00 pm every day is not a significant improvement over backing up at once a day, since the entire working day falls between the two backups. If you lose a disk at 8 :00 pm, you lose all of that day's work, either way. Backing up at 12 noon and 12 midnight is much better, since it splits the day in half. (It has another advantage in that noon and midnight are usually lower use times than 8 :00 am and 8 :00 pm.)

The best you can possibly do is to run backups back-to-back ; that makes your maximum data loss the amount of time a backup cycle takes. On the other hand, if your site is large enough, and the resource you are backing up to is limited enough, this may be one working day of loss. See section 7 for suggestions on how to reduce the time a backup cycle takes. If you cannot reduce the time a cycle takes to the maximum amount of loss you can afford, you need to supplement your backup system with some form of redundant disk.

## 5.2 Calculating Loss, Multiple Errors

There are, unfortunately, several more complex calculations of maximum loss that need to be done before you can fully comprehend your situation. The case that we have been considering is the case in which a file is gone, but its backup is good and the loss is discovered immediately. (If you lose a whole filesystem, the loss is likely to be discovered immediately. Individual files which are lost or corrupted may remain unnoticed for some time.) Backups are not always good. In particular, if you choose to back up disks to other disks, you increase the likelihood that your backup is lost at the same time your original is. Earthquakes, lightning, and floods all have an unfortunate tendency to destroy **all** magnetic disks attached to the system when they happen. Therefore, you should have removable media backups at the maximum data loss interval you can put up with if such a natural catastrophe happens. This is probably a longer interval than your normal one ; people are more willing to put up with losses incurred through disaster than losses incurred due to more normal causes, like human stupidity.

Even removable media backups are not completely reliable. They are considerably less likely to be destroyed when the disks are, but they are unfortunately likely not to have been completely good in the first place. See section 8 for information on how to increase their reliability. You should also calculate how much data you can lose, worst case, if a single tape turns out to be bad. (I estimate that approximately 10 percent of the backup tapes I have had to restore from have had some sort of errors, ranging from being effectively blank, to giving read errors but successfully retrieving the particular data I was interested in.) These calculations are much more complex, because unlike the previous calculations, they depend not only on when backups are done, but also on what level the backups were done at, and how long the tapes are saved for.

If you back up a given machine with a level 0 at 3 :00 am on Monday morning, and a level 7 at 6 :00 pm every weekday, your normal maximum data loss is 57 hours (from 6 :00 pm on Friday to 3 :00 am on Monday). Those are also fairly low usage hours. If you lose a single tape, things are more confusing. The worst level 7 you could lose is Friday's ; that would lose 81 hours of data, including all of Friday's work. If you lose the level 0, what happens ? The answer depends heavily on what and when you are trying to restore, and what backup program you used to write the data, and how lucky you are. Once a tape is gone, it may not be possible to apply incremental tapes that come after it.

Suppose that it is Wednesday, and Monday's level 0 is no good. If you try to restore an individual file, you will be fine unless it changed between 6 :00 pm Friday and 3 :00 am Monday ; if it changed before that, it will be on Friday's level 7, and if it changed after that, it will be on a subsequent level 7. In this case, what has happened is that the data loss has increased by 57 hours, in a rather unpredictable way. If you try to restore an entire directory and that directory existed before 3 :00 am Monday, life is apt to be rather more confusing. Normally, you would restore the directory from the level 0, and then again from the most recent level 7. You would expect the restore program to apply all the changes, including deleting files that were removed between the time the 0 was done and the time that the 7 was done. With

the level 0 missing, it may not be capable of doing that (will not, if it is in fact `restore`). You will be able to restore the previous level 0 and Friday's level 7 normally ; you will then lose changes between Friday and Monday, and you may have to make special efforts to pull in the files that were added after Monday. If you lose a whole file system, this will be particularly painful. Not only has the data loss increased by 57 hours, the effort required to recover has also increased, and the resulting system may not be identical to the one you started with.

If you do not have the previous level 0 (for instance, you reused it to write the most recent level 0), you are in big trouble. Instead of having lost a countable number of hours of work, you only have saved a countable number (you have the changes between 3 :00 am last Monday and 6 :00 pm last Friday, and the changes between 3 :00 am this Monday and 6 :00 pm this Tuesday). Everything else on the file system is gone. This is probably not an acceptable level of risk ; you should not design a system where there is only one full dump in existence at any given point.

### 5.3 Calculating Loss, Late Discovery

When you lose an entire disk, generally you notice before you go back to do another backup. Smaller disasters, on the other hand, may go unnoticed for quite some time. Users may delete or corrupt files and then go days or weeks without noticing. There is obvious a limit to the the extent to which you can cope with this ; the user who wants a file that was deleted 3 years ago should not be surprised if you cannot get it. Before you get asked for that file, decide how long you're going to keep backups for, and then make certain that you do effectively keep them that long.

Many people keep only selected backups. If you keep daily backups for 2 weeks, weekly backups for 2 months, and full backups (done every month) for 2 years, then you are increasing maximum possible loss as time passes. A file needs to have existed for at least a week to be available 2 months later, and for at least a month to be available 2 years later. This is probably reasonable. Keeping backups that are much farther apart than a month probably isn't, since you rapidly become unable to predict whether a file is going to be there or not. Keeping incrementals without the backups they are incremental to also creates a random hit-or-miss effect, since you are back to having only files modified between specific dates.

### 5.4 Reducing the Loss

You can reduce maximum loss in the following ways ;

1. Increase the frequency with which you do backups.
2. Increase reliability of dumps (see 8)

Increasing the frequency of backups also increases the resources necessary to do them. However, it minimizes data loss in all cases. You can compensate to some extent for the increased resources by keeping fewer backups. For instance, you might choose to do two backups every day, but keep one of them for only one day. This will complicate the schedule. If you use media that has a limited capability for re-writing, you will want to avoid keeping the same tapes in a pool that are being frequently re-written, which also complicates the schedule. Magnetic tapes in general are vulnerable to too much re-writing, and 8mm tapes are much more vulnerable than half inch tapes.

## 6 Minimize Resources Used to Restore Data

This is a rather tricky goal, since things that minimize time to retrieve individual files often make it more difficult to restore entire file systems. (For instance, dump(8) is very good for restoring whole file systems, but can be extremely annoying if you need to retrieve an individual file.) Most sites will be most interested in minimizing time to restore individual files (if you lose whole file systems frequently, you have a deeper problem than can be fixed by your backup system.) On the other hand, you need to reduce the amount of time required to restore a full file system to the amount of time you can afford to be down. Restoring is usually slower than backing up (creating files is slower than reading them, and harder to do in parallel). Furthermore, if you are restoring an entire filesystem, the restore will tend to get slower as it goes, since creating a file takes longer on a full file system than on an empty one. File systems are designed to minimize this effect, but they also make assumptions about patterns of usage and creation that are not met when you create all the files in a filesystem one after another.

Ways to meet this goal :

1. Keep catalogues of which files are on what tape.

2. Make it possible for users to restore their own files.
3. Use fast media.
4. Keep backups on-line.
5. Use a system capable of deleting restores.
6. Reduce the number of different levels you use.
7. Increase the frequency of level 0s.

## 6.1 Keeping Catalogues

Most of the overhead in restoring an individual file is finding it. Users are rarely certain enough of what they changed when to let you choose a tape immediately. (Many times they don't even know what the file was named or what directory it was in for sure.) A catalogue of what was dumped when allows you to quickly figure out what tape the file is on. Most commercial backup programs will do this for you ; if you are using `dump`, `tar`, `cpio`, or some variant, you will have to roll your own. The main drawback is that you have to keep the catalogue somewhere, presumably on disk, and it can run into large amounts of space quite fast.

## 6.2 Letting Users do Their Own Restores

A system where users can restore files themselves, safely, reduces wear-and-tear on everybody. The users do not have to wait for system administrators to get around to them, and the system administrators do not have to explain why it's taking them three days to find the time to restore a .newsrc. Remember, however, that this happy state of affairs only fully applies when the user can find and mount any necessary media. Otherwise, you have simply shifted the problem from a request for a restore to a request to have a tape found and mounted. In general, letting the users wander through the backup tapes at will is not a good idea.

## 6.3 Using Fast Media

If your restore program is limited by the media speed, increasing that speed will make all restores, full or partial, faster. The question to consider is whether faster media will make a cost-effective difference. For `restore` in particular, the difference between a normal half-inch tape drive and a fast half-inch tape drive is probably irrelevant ; the tape drive is only going to be a limiting factor for small restores. The speed difference between a DAT drive and an Exabyte drive is primarily in the fast seek speed, and will make a significant difference if and only if the restoring program is capable of using fast seeks. If you are restoring a full file system that's at the beginning of a tape, it's probably irrelevant.

## 6.4 Keeping Backups On-Line

For speed, nothing beats having the backup on-line as a file system. Straight disk-to-disk copy is about as fast as it gets. Unfortunately, it requires lots of disk space, and it makes your backups vulnerable to many of the same problems as your originals. Furthermore, copying file systems disk-to-disk is not as easy as it looks ; you basically have to do a backup piped to a restore. Doing a full restore loses some of its speed in the pipeline

Most people who use on-line backups simply run their backup program to a disk file instead of a tape, often compressing the file afterwards, and then copy the result to a tape. Using a disk file instead of a tape buys some speed. Compressing it loses speed and adds vulnerability. Moving the compressed backup from disk to tape almost always results in a backup that is much slower to restore, particularly if you go the route of backing up the partition full of backups. Once you have backed up a partition full of compressed backups, getting a file requires doing a restore, an uncompress, and another restore. Unless you know that most of your restores are going to be from files still on disk, this is probably not worth it.

## 6.5 Deleting Restores

When you do a full restore from multiple levels of backups, there will be files that have been removed or renamed between the backups. A deleting restore is one that is capable of removing files that have gone away, as well as adding

files that have been created. `restore` is capable of deleting restores ; programs like `tar` and `cpio` are not. If you have to do a restore that involves multiple levels without deleting, you are likely to have to do a lot of juggling by hand, since there will be more files restored than originally existed.

### 6.6 Reducing the Number of Levels in the Scheme

If you do a level 0 on Monday, a level 1 on Tuesday, a level 2 on Wednesday, and so on through the week, and your disk collapses into a heap of slag on Sunday night, you will have to restore all seven backup tapes onto the replacement disk. This is one of the shortcomings of the true incremental scheme. If the per-tape overhead (the time it takes you to notice that a new tape is needed, find the new tape, unmount the old tape, mount the new tape, and the time it takes the program to read the initialization information off the new tape) is reasonably low, this may not be a problem. If you are using `restore`, you are in for the long haul. Making the dumps less incremental uses more tape, but may make a dramatic difference in the time the backup takes. (Even when it doesn't make a large difference in the time, at 2 :00 am the difference between three tape mounts with three hours between them and nine tape mounts with one hour between them is the difference between three pretty good naps and nine hours of misery.)

### 6.7 Increasing the Frequency of Level 0s

The more frequently you do full dumps, the less data has to be added to them from incrementals. Not only does this usually reduce the number of tapes mounted, but in many cases there is extra overhead in restoring incrementals. An incremental restore may need to remove files, for instance.

## 7 Minimize the Resources Used For the Backup System

This includes minimizing the time that backups take, the amount of systems staff attention, and the amount of backup media used. Time in particular becomes a limiting factor on very large systems ; once you have 14 gigabytes of disk space, it is all too easy to design a system where you cannot finish one set of backups in time to start the next. In these systems you can also easily run into a limitation in physical space, i.e. more tapes than places to put them.

Ways to meet this goal :

1. Make the backups run automatically.
2. Compress the backups in some fashion.
3. Run multiple dumps to the same tapes.
4. Use fast media.
5. Use media with a small form-factor.
6. Minimize network usage ; keep backup media local to the machine being backed up.
7. Schedule the backups for times of particularly low network and CPU load.
8. Use a transport program that has been optimized.

## 8 Maximize the Reliability of the System

Backup systems are prone to failure at several points, including damaged/lost media, forgetful human beings, and files that the transport programs cannot cope with. These problems come in two forms ; transient failures, and systemic failures. When the tape drive catches fire, that's a transient failure. When you write multiple dumps onto the same tape, rewinding the tape in between so that they all write on top of each other, that's a systemic failure.

Transient failures, while nasty, are both less worrisome and less fixable than systemic failures. You can compensate for them adequately by adding redundancy, although you will of course also want to avoid as many as possible. Systemic failures should be searched out and avoided ; if this is not possible, users should at least be warned about them.

Ways to reduce transient errors :

1. Verify backups after they are written. If possible, verify them on a device other than the one they were written on.

2. Do backups at times of low filesystem usage, or forcibly prevent file system usage while machines are being backed up.
3. Run `fsck` before doing backups.
4. Increase redundancy.
5. Add multiple checks and balances to the system. For instance :
   (a) Add a check of whether the backups have been done that runs separately from the backups themselves.
   (b) (Particularly in the case of automated backups) Inform multiple people of the status of backups as they complete.
   (c) Label the media with both human-readable and machine-readable labels, and read them both.
6. Use devices that you have more than one of ; use formats that can be read on more than one system.
7. Automate backups.

## 8.1 Verifying Backups

In a perfect world, you would always be able to write a tape, and then verify that the tape had all the right data on it. This is an imperfect world, and it's almost impossible. You can only check a backup against a file system if the file system is the same now as it was when you wrote the backup – that is, it has to be an idle file system, and it has to be idle long enough to do the backup and the verify, which will take as long as the backup did. If you are in this situation, you can certainly create a verification program, even if your transport program doesn't have one (which it probably doesn't).

Most of us cannot actually keep file systems idle for that long on any regular basis. In this situation, the best you can possibly do is to verify that the tape has valid data. The best way to do this is to actually read the entire tape. Some people attempt to do this by getting a table of contents of the tape, or by trying to read the last file on the tape ; whether these accomplish anything useful depends a lot on the transport program. Using `dump`, for instance, the entire table of contents is at the head of the tape, and reading the last file skips the intervening ones completely ; even if you check both, much of the tape could be garbage without your noticing. It is slower, but more reliable, to read every single file.

If possible, you should verify removable media on a drive different from the one that wrote them. Heads can drift out of alignment slowly, making drives that write media only they can read, and that only for a little while after they wrote them. This is almost impossible to automate, which makes it extremely burdensome.

## 8.2 Doing Backups on Idle File Systems

This is the good old "active dumps" question. The design of the UNIX file system makes it almost impossible to do reliable backups of a file system that is changing. The primitives that would allow you to lock things down simply don't exist. There are some file systems for UNIX machines that don't have this problem (AFS, for instance), but they aren't in wide use.

If you are using `dump`, the only truly safe way to do backups is to ensure that the file system is idle, either by unmounting it, or by bringing the machine into single-user mode. On a machine that may have local processes using the file system, the only way to guarantee that you can unmount it is to bring the machine into single-user mode. (If all uses of the file system are always through NFS, it will be unmountable.) In order to do automatic unattended backups in single-user mode, you create a file in the root partition, and you have the startup files check for the existence of that file, and first delete it and then run the backups if it exists. Be sure not to create the file until you are ready to have the dumps done ; you don't want the machine to do backups after an accidental crash.

`dump` is by no means the only program that has problems with active file systems. Programs differ in what they do when they encounter active files, but most of them will either create corrupt archives, or skip the file. Do not assume that active backups are OK because you are not using `dump`, or because a salesman told you so.

It is obviously impractical to do all backups on idle file systems, and it is perfectly reasonable to back up active file systems, as long as you are aware that it will introduce transient errors, in the form of incomplete and/or skipped files, from time to time. The lower the activity on the file system, the less the risk, so schedule backups at low-usage times. Speeding up the backups also helps, by narrowing the window during which activity matters. (It may also have second-order effects ; Ohio State's speed-modified dump often took so much memory, CPU, and disk controller capacity that nobody else was capable of modifying the disk while it ran.)

## 8.3 Increasing Redundancy

If you have copies of a file on multiple tapes, you can afford to be calm when an operator decides to juggle them, drops one, and shatters the case. The obvious way to achieve this effect is to duplicate tapes. Tape duplication between tape drives is slow, and the duplicate will have all the errors on the original, plus any introduced in the duplicating process. If you happen have a tape duplicator handy, it may be reasonable to use it, but going from drive to drive probably isn't.

Doing more backups is the next most obvious gambit, and is usually a better idea. Doing repeat backups at the same level not only gets the same files on multiple tapes, it also decreases maximum loss. In fact, simply reducing the number of levels you use, without increasing frequency, will usually increase redundancy.

When you are doing multiple backups at the same level, you need to keep careful track of what is incremental to what. For instance, if your original schedule is level 0, level 2, level 4, level 6, changing it to level 0, level 2, level 2, level 4 makes the level 4 incremental to the second level 2. If you lose that level 2, you may not be able to do a normal restore of the level 4, even though you have a perfectly good level 2 left. Doing two level 0s back-to-back is not a major win in redundancy because of this effect. The simple way to avoid this effect is to flatten from the highest level down, giving you level 0, level 2, level 4, level 4 ; the level 4s are redundant, since nothing is incremental to them.

There is also a complex way to avoid it. You can juggle things so that the incrementals are incremental to different previous dumps. If you do level $0_1$, level $0_2$, level $2_1$, level $2_2$, level $4_1$, level $4_2$, you minimize the effect of losing any individual dump. If the level $2_1$ is missing, you can use the level $2_2$ and its corresponding $4_2$. This system can be implemented with `dump` by interchanging /etc/dumpdates files, and provides a nice additional redundancy. It also confuses people nearly to death, which is a major drawback.

## 8.4 Adding Checks and Balances

Rather than putting your trust in one person or one program, provide double-checks. For instance, run a separate program that checks to see whether or not backups have been done, and reports when they have not. (It's important that it only report exceptions ; if it always says something, people will drop out of the habit of reading it.) This program should send reports to not only the people who are directly responsible for doing the backups, but also other people.

The backups should also send out status messages to multiple people. These messages should report what happened, even if there are no exceptions. Not only does this help give people a warm fuzzy feeling, it protects against failure of the checker. (If the checker fails altogether and says nothing, it appears to have approved things.)

## 8.5 Systemic Errors, and Avoiding Them

The most common causes of systemic errors are :

1. Dumping active file systems ; if you dump a file system that is in use at the same time every day, you risk being synchronized with programs or people that modify the same file at the same time every day. Most people assume that a file that is damaged because it is being modified will not always be in that state. That is, they assume that doing backups of active file systems always creates transient errors. Check this before believing in it.
2. Flaws in the transport program, including problems with :
   (a) Long file names ; some transport programs are incapable of dealing with pathnames over 100 or 128 characters long. Since most UNIX versions will allow pathnames of up to 1024 characters, it is not difficult for users to create whole subtrees that will not get backed up. People who are using shells with filename completion or icon-oriented programs are more apt to do this.
   (b) Files with holes in them ; most UNIX file systems allow programs to create files which represent large stretches of nulls without writing them to disk. This feature is often used by core files, and by dbm databases. It is extremely important that programs used for backups do not fill in these holes, either in writing to tape, or in restoring from tape. Since the / file system is often smaller than the memory size of the machine, and frequently contains core dumps which are the same size as the memory, filling in the holes will result in being unable to restore the file system.
   (c) Other peculiar filenames ; some programs have difficulty restoring files with names that are legal but unusual (names containing spaces, or 8-bit characters, for instance).

(d) Symbolic links ; some programs will copy the target of a symbolic link, rather than the link itself. This may result in a backup that is too large to restore (and will certainly cause confusion and unhappiness).

(e) Large numbers of hard links.

(f) Long targets of symbolic links.

3. Filenames that include "/" ; "/" is not a legal character in a UNIX filename, but many implementations of NFS allow clients to create such files. They may or may not get backed up, but they can't be restored except by extreme measures.

4. Design errors in the system, including :

   (a) Failing to allow for complete failures ; requiring a restore program which is not shipped with the operating system and which is only stored on a tape that it must be used to a restore. If you lose everything, you can't read the tapes to restore them. This is easier to do than you might think.

   (b) Rewinding tapes between writes.

Avoiding systemic errors is difficult. Knowing that they're possible is the first step ; testing by doing unnecessary restores is the next one. You don't want to find out where the problems are when the disk dies.

## 8.6 Maximize the Flexibility of the System

There are two sorts of flexibility that you need ; temporary, to deal with extraordinary events, and permanent, to deal with changes in your computer system.

Extraordinary events do occur. Sometimes you may be unable to meet the required schedule (to take an example from life, you may have to postpone all backups one day because all your machines are shut down due to an earthquake). Sometimes you may need to increase the frequency of backups on one particular set of files. You will inevitably need to back up more disk space than originally planned for. You may also have occasions on which very large amounts of data change at once (in the worst case, you may lose a file system and restore it). Your system should be able to cope easily with all of these.

Expansion is also inevitable. There is no such thing as enough disk space. There is also no such thing as enough CPU power, so people will buy more machines, and newer ones. When you put together your backup system, you should know what you will need to figure out how far it will expand as is, and what will need to be done when you reach the end of that tolerance.

Ways to meet this goal :

1. Use widely separated levels ; do not normally use the highest level. (For instance, rather than using levels 0, 1 and 2 or levels 0, 5, and 9, use 0, 3 and 7 so that you have spare levels you can use at every point in the hierarchy.) If you are designing the levels, write in more than you need.

2. Do not design to exactly meet capacity. For instance, if you are scheduling machines to do things on a cycle with one machine per day, make the cycle large enough to leave empty days.

3. Put in override options, so that people can do extra backups on individual machines or filesystems within the system.

4. Always have a redundant device to back up to.

## 8.7 Maximize the Predictability of the System

It should be possible to summarize the conditions under which you can restore a file for an average user. Any given file should be saved on backup for the same amount of time as any other file on the same file system. This is easy for simple systems ; on the other hand, if you do disk-to-disk backups followed by copies to tape, the interaction of the the disk-to-disk schedule with the disk-to-tape schedule may create unexpectedly good or bad days to modify files on.

# 9 Issues in the Selection of a Transport Program

## 9.1 Through the File System vs. Below the File System

Some programs, like `tar`, `cpio`, and `bru`, access files to back them up by using normal UNIX access methods (going through the file system) ; others, like `dump` and `dd` access files to back them up by reading the disk more-or-less directly (going below the file system). Going through the file system is somewhat more flexible, since you can back up anything you can get a standard file system interface to (including NFS or RFS mounted remote file systems). On the other hand, it exposes you to whatever oddities and bugs there are in the file system interface, and it may cause interesting interaction with programs (like some NFS server implementations) that do not themselves work through the file system. Going below the file system avoids most of the problems with odd features of the file system, but makes programs more susceptible to active file systems. Both ways are equally vulnerable to vendors who re-arrange the file system. (For instance, Hewlett-Packard's new access control lists are not backed up by either `dump` or `tar` as supplied by Hewlett-Packard ; `dump` believes that all disk blocks are either inodes or data pointed to by inodes, and can't see the access control lists at all, whereas `tar` writes a format that has no place to put the information. For reasons unconnected to the file system, `dump` is fixable but `tar` is not.)

Benefits of going through the file system :

- Allows backup of file systems other than locally-connected disk.
- Allows backup by directory tree instead of by file system ; you can backup part of a filesystem, or several file systems at once.
- Reduces sensitivity to active file systems (the file system itself is designed to present a consistent view of an active file system).
- Usually increases portability.

Benefits of going beneath the file system :

- Avoids sensitivity to odd facets of the file system. (Devices, long file names, funny characters in file names, holes in files, etc. etc.)
- Allows incrementals to work correctly. In order to keep archive runs from causing all files on the system to appear to be active, most archive programs modify the file access time after they have accessed the file. This keeps access time accurate, but updates the inode change time. There are other operations, like changing the owner or permissions of a file, which update only the inode change time. Programs that go through the file system cannot avoid this problem ; programs that go beneath the file system do not have any effect on the access time in the first place.
- Allows backups by users other than root ; permission to backup a file system depends on the device permissions, not the permissions on individual files.

Most people find the advantages of backing up through the file system more comprehensible, and in general more convincing. In theory, the problems with permissions and with the use of mtime instead of ctime for incrementals are sufficiently minor that a robust and well-designed program working through the file system should be preferable to one that works beneath it. In practice, such programs are rare if not mythical. The well-known programs that work through the file system, including `tar`, `cpio`, and `pax` are not even mildly robust. Furthermore, even a program that was as perfect as possible would still run into problems with the file system. The file system is designed to conceal some information that is necessary in order to do effective backups (for instance, it is designed to conceal the existence of holes).

Ci joint dans la version imprimée de ce cours, un exemple de plainte pour vol.

RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'INTÉRIEUR, DE LA SÉCURITÉ INTÉRIEURE ET DES LIBERTÉS LOCALES
DIRECTION GÉNÉRALE DE LA POLICE NATIONALE

PREFPOL/DPUP
SVP / SPQ 13ème arrdt
Tél : 01-40-79-05-05

Code INSEE du service : 075 651/113

# COMPTE RENDU D'INFRACTION INITIAL

VICTIME/DECLARANT N°

| | |
|---|---|
| VICTIME | **UNIVERSITE PARIS 7 - BIBLIOTHEQUE DE MATHEMATIQUE**, société FRANCAISE<br>établi(e) 16, RUE CLISSON 75013 PARIS<br>Téléphone : 01.44.27.37.23 |
| PREJUDICE | Butin : A EVALUER Dégâts : AUCUN |
| INFRACTION | **VOL SANS EFFRACTION**<br>Entre le 28/07/2006 à 19 H 00 et le 31/07/2006 à 08 H 00 (INDT), PERIODE CONGES SCOLAIRES.<br>16, RUE CLISSON à PARIS (13 arrdt).<br>Nature du lieu : LIEU PRIVE<br>Eléments d'enquête :<br>- Personnes remarquées : néant<br>- Mode opératoire : VOL SANS EFFRACTION DE DEUX ORDINATEURS. |

P.V. : 2006/ /001
Affaire contre X
Pièces jointes :
NEANT
Scellés : non

Transmis à :
MONSIEUR LE PROCUREUR DE LA REPUBLIQUE
tribunal de grande instance de paris

Date :
Nom : BIUNDO Stéphanie
COMMISSAIRE DE POLICE

# PROCES VERBAL

L'an deux mille six,
le trente et un juillet à quinze heures cinquante

Nous, **Laurent** [redacted]
GARDIEN DE LA PAIX
Agent de Police Judiciaire
en fonction U.P.Q PARIS 13ème
en résidence PARIS 13ème

---Etant au service, -------
---Constatons que se présente devant nous la personne ci-dessous dénommée qui nous déclare:---
**********
▪ --- **SUR SON IDENTITE** : ----------
--- "Je me nomme [redacted] **Odile.**" ----------
--- "Je suis née le 25/10/[redacted] à AULNAY SOUS BOIS (SEINE SAINT DENIS)." ----------
--- "Je suis de nationalité FRANCAISE." ----------
--- "J'exerce la profession de : BIBLIOTHECAIRE." ----------
--- "Je suis domiciliée 16, RUE CLISSON 75013 PARIS." --------
--- "Mon numéro de téléphone professionnel est le ----------
[redacted]" ----------
**********
**---SUR LES FAITS:---**
---"Je me présente à vous au nom de *la Bibliothèque de Mathématiques recherche, propriété de l'université Paris 7, sise*

.../...

*16 Rue Clisson à Paris 13ème*, que je représente en tant que bibliothècaire pour signaler qu'elle a été victime d'un vol sans effraction.---
---En fait j'ai fermé l'établissement vendredi dernier le 28.07.2006 vers 19 heures.---
---J'ai verrouillé la porte d'entrée à clé.---
---La bibliothèque ferme tout le week-end.---
---Ce matin le 31.07.2006 à 08 heures 00, au moment ou j'ai voulu j'ai ouvrir la porte d'entrée, j'ai constaté qu'elle n'était plus verrouillée.---
---Elle était simplement claquée.---
---Cela m'a véritablement étonné.---
---J'ai alors visité l'intérieur de l'établissement salle, par salle et j'ai constaté qu'il manquait deux ordinateurs.---
---Les deux machines manquantes se trouvaient à la base dans la salle de lecture principale.---
---Il s'agit de *deux ordinateur Imac, 2.1 Ghz,*
*le premier numéro de série : SW8547329TAR*
*Le deuxième numéro de série : SW8546Y2TTAR*
---Il s'agit d'ordinateurs complets, le moniteur sert d'unité centrale.---
---Concernant le mode opératoire, je n'ai pas d'explication ni de doute.---
---Beaucoup de monde à la clé des lieux, car en ce moment il y a des travaux et beaucoup de va et vient.---
---Il m'est impossible de soupçonner qui que ce soit.---
---Je n'ai pas pensé à faire intervenir les fonctionnaires de Polices, mais vu la configuration des lieux et vu la fréquentations des lieux toutes recherches d'empreintes s'avère impossible.---
---Je n'ai aucun soupçon quant à ou aux auteurs des faits.---
---Je dépose plainte au nom de l'université Paris 7 contre inconnu pour les faits relatés ci-dessus.-----
---Je prends acte des dispositions de l'article 53-1 du Code de Procédure Pénal visant à obtenir réparation pour le préjudice commis et/ou a être assisté par un organisme d'aide aux victimes.---
---Je prends acte que je ne serais avisé des suites de l'affaire qu'en cas de recherches positives.---
---Je n'ai rien d'autre à ajouter ou à retrancher à mes déclarations".---
---Après lecture faite par lui-même le déclarant persiste et signe avec nous le présent Procès-Verbal.---

LA DECLARANTE L'A.P.J.

RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'INTÉRIEUR, DE LA SÉCURITÉ
INTÉRIEURE ET DES LIBERTÉS LOCALES
DIRECTION GÉNÉRALE DE LA POLICE NATIONALE
PREFPOL/DPUP
SVP / SPQ 13ème arrdt
144 BOULEVARD DE L'HOPITAL
PARIS (13 arrdt)
Tél. : 01-40-79-05-03

# RECEPISSE DE DECLARATION

**UNIVERSITE PARIS 7 - BIBLIOTHEQUE DE MATHEMATIQUE**
établi(e) 16, RUE CLISSON 75013 PARIS

a déclaré avoir été victime de l'infraction suivante :
**VOL SANS EFFRACTION**

survenue :
Entre le 28/07/2006 à 19 H 00 et le 31/07/2006 à 08 H 00,
16, RUE CLISSON à PARIS (13 arrdt)

**MODE D'OPERER :** VOL SANS EFFRACTION DE DEUX ORDINATEURS.

**PREJUDICES :** Butin : A EVALUER Dégâts : AUCUN

Plainte déposée le 31 Juillet 2006
Sous le numéro 2006/ /001

Article 441-6 du Code Pénal. Le fait de se faire délivrer indûment par une administration publique ou par un organisme chargé d'une mission de service public, par quelque moyen frauduleux que ce soit, un document destiné à constater un droit, une identité ou une qualité ou à accorder une autorisation est puni de deux ans d'emprisonnement et de 30000 euros d'amende. Est puni des mêmes peines le fait de fournir une déclaration mensongère en vue d'obtenir d'une administration publique ou d'un organisme chargé d'une mission de service public une allocation, un paiement ou un avantage indu.

Fait à PARIS, le 31 Juillet 2006

L'Agent de Police Judiciaire,
Laurent [redacted]
GARDIEN DE LA PAIX,

DEPOT DE PLAINTE
contre un auteur inconnu

Date de dépôt de la plainte : 31/07/2006

Nom et prénom du plaignant : UNIVERSITE PARIS 7 - BIBLIOTHEQUE DE MATHEMATIQUE
Objet de la plainte : VOL SANS EFFRACTION.

Date des faits : Entre le 28/07/2006 et le 31/07/2006

Parquet du procureur de la République tribunal de grande instance de paris

Service : SVP / SPQ 13ème arrdt
144 BOULEVARD DE L'HOPITAL - PARIS (13 arrdt)

## INFORMATION SUR LES DROITS DES VICTIMES

Vous avez été victime d'une infraction pénale. Cette note est destinée à vous informer sur la teneur et les conditions d'exercice de vos droits.

### Association ou service d'aide aux victimes

Vous pouvez obtenir des informations complémentaires et être aidé dans vos démarches par l'association ou le service d'aide aux victimes suivant :

### Permanence gratuite des avocats

Des consultations gratuites sont en outre organisées par l'ordre des avocats :

L'enquête effectuée sur l'infraction dont vous avez été victime sera transmise au procureur de la République qui peut donner différentes suites à la procédure :

### Poursuites

Le procureur de la République peut décider d'exercer des poursuites en saisissant directement une juridiction de jugement ou en ouvrant une information judiciaire devant le juge d'instruction.
Pour obtenir réparation du préjudice subi, vous serez en droit de vous constituer partie civile afin de demander des dommages et intérêts à l'auteur de l'infraction.
Si votre demande de dommages et intérêts a déjà été recueillie par procès-verbal lors de votre audition par les services d'enquête, elle vaut constitution de partie civile à condition que le procureur de la République donne son accord. Si votre demande paraît manifestement inférieure au préjudice que vous avez subi et si cet accord n'a pas été donné, ce magistrat vous permettra de reformuler votre demande. Vous en serez alors avisé.
Si une information est ouverte par le procureur de la République, la constitution de partie civile sera possible à tout moment devant le juge d'instruction. Ce magistrat vous en informera du reste personnellement.
Si l'affaire est audiencée devant le tribunal correctionnel, le tribunal pour enfants ou le tribunal de police, vous serez avisé de la date d'audience et vous pourrez vous constituer partie civile (1) :
- soit en personne, en vous rendant au greffe du tribunal avant l'audience pour faire une déclaration ou en vous présentant au début de l'audience avec les pièces justificatives de votre préjudice ;
- soit par lettre recommandée avec accusé de réception ou par télécopie en utilisant le formulaire ci-joint qui doit parvenir au tribunal vingt quatre heures au moins avant la date d'audience, en joignant les pièces justificatives de votre préjudice ;
- soit par l'intermédiaire d'un avocat.

Chapitre 12

# *Gestion de la mémoire virtuelle*





UNIX $\equiv$ système multi-utilisateurs $+$ multi-tâches

$\Rightarrow$ nécessité d'une gestion sophistiquée de la mémoire de tous les processus

Sur une machine à base de processeur Intel x86 : 4 GB de ram maximum

Grâce au mécanisme PAE (« Physical Address Extension ») : 64 GB maximum

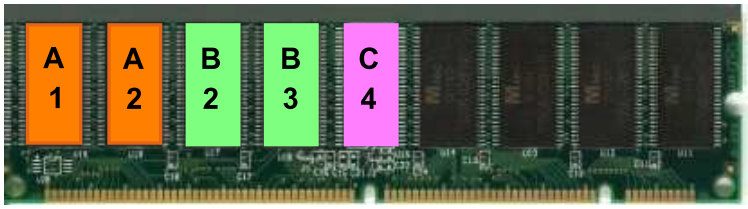
Principe approximatif de la mémoire virtuelle :

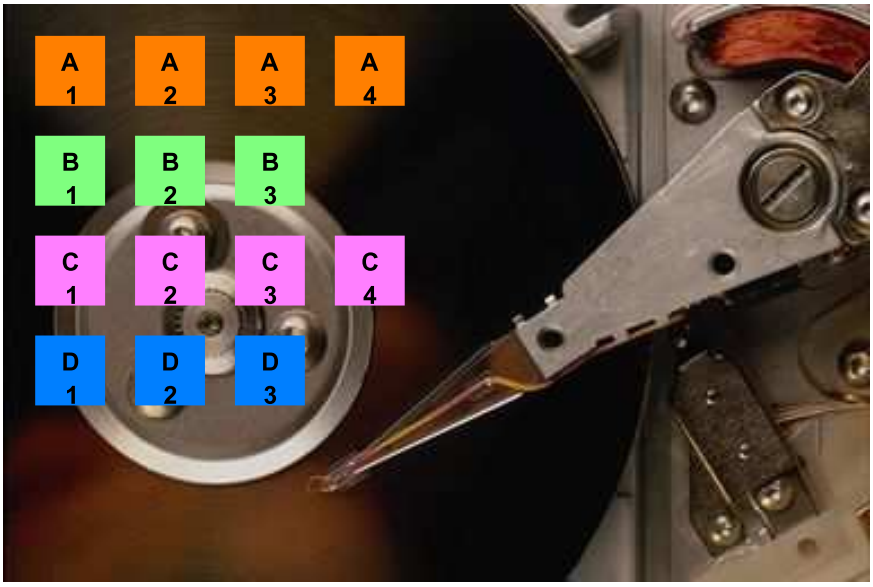
- Un processus a l'impression d'avoir un espace mémoire > RAM.
  C'est la **mémoire virtuelle**.
- L'espace mémoire est découpée en segments de taille donnée : les pages.
  On charge les pages d'un processus en RAM au fur et à mesure des besoins.
  C'est la **pagination**.
- Quand la RAM est « pleine » et que l'on doit charger des pages, on copie sur disque de vieilles pages et on les élimine de la RAM pour les remplacer par les nouvelles pages.
  C'est ce que l'on appelle **swapper**.

Le noyau gére ce mécanisme de mémoire virtuelle.







Le fonctionnement d'une machine UNIX standard nécessitera donc une partition disque pour le swap.

La partition de swap est créée comme les autres partitions régulières.

Par contre, on n'a pas besoin de faire « `newfs` » sur la partition de swap.





Comment connaître la taille du swap ?

Sur LINUX : utiliser « `free` »

```
# free
             total       used       free     shared    buffers
Mem:        129488     121716       7772       4136     104824
-/+ buffers:            16892     112596
Swap:       259768          0     259768
```

Comment connaître la taille du swap ?
Sur SOLARIS, 2 possibilités.

Utiliser « swap -l »

```
# /usr/sbin/swap -l
swapfile             dev  swaplo blocks   free
swapfs                 -       0 139408 123280
/dev/dsk/c0t3d0s1   32,25       8  98488  98488
```

Utiliser « swap -s »

```
# /usr/sbin/swap -s
total: 8264k bytes allocated + 3520k reserved = 11784k used,
52632k available
```





Suite de commandes à donner :

```
# free
             total       used       free     shared    buffers     cached
Mem:        512720     198032     314688       1168      13428      89168
-/+ buffers/cache:      95436     417284
Swap:      1052216          0    1052216

# mkswap -f /dev/hda5
# swapon  /dev/hda5

# free
             total       used       free     shared    buffers     cached
Mem:        512720     198128     314592       1168      13428      89180
-/+ buffers/cache:      95520     417200
Swap:      1068208          0    1068208
```

Syntaxe : `swap -a partition`

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t0d0s1   136,1      16 4198304 3455392
# swap -s
total: 601128k bytes allocated + 230824k reserved = 831952k used,
1686832k available
# swap -a /dev/dsk/c0t0d0s6
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t0d0s1   136,1      16 4198304 3455392
/dev/dsk/c0t0d0s6   136,6      16 1440224 1440224
# swap -s
total: 601128k bytes allocated + 230824k reserved = 831952k used,
2406896k available
```





Les scripts de démarrage activent automatiquement les partitions de swap au boot.

Les partitions sont indiquées dans le fichier « `/etc/fstab` » :

```
/dev/hda2          none    swap   sw        0 0
```

Les scripts de démarrage activent automatiquement les partitions de swap au boot.

Les partitions sont indiquées dans le fichier « /etc/vfstab » :

```
#device             device  mount   FS      fsck    mount   mount
#to mount           to fsck point   type    pass    at boot options
#
/dev/dsk/c0t0d0s1   -       -       swap    -       no      -
```





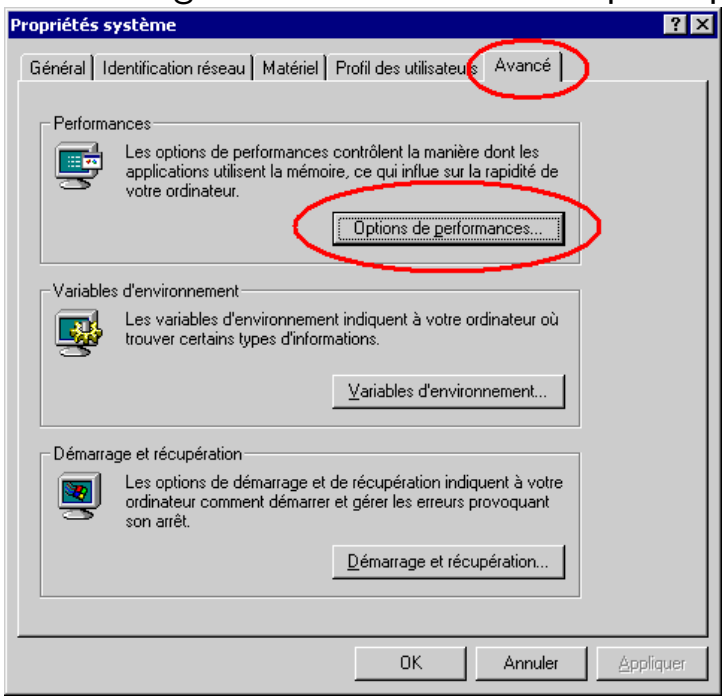
Quelle taille donner à une partition de swap ?
Par tradition on met deux fois la RAM.

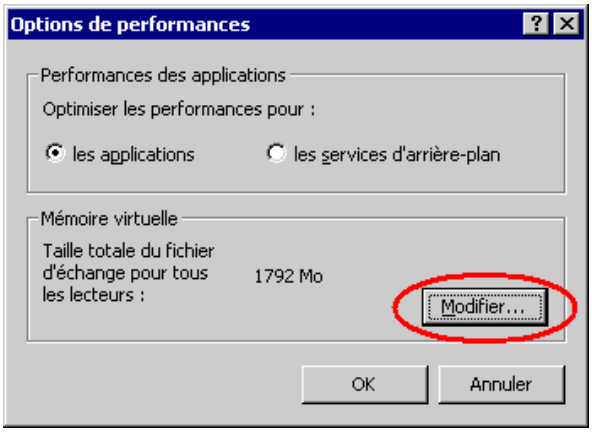
En cas de taille de swap insuffisante, aïe, problèmes :

```
# ./programme-gourmand
No swap space available

Sep 15 17:07:43 solaris.example.com /kernel: pid 335 (z),
uid 0, was killed: out of swap space
```

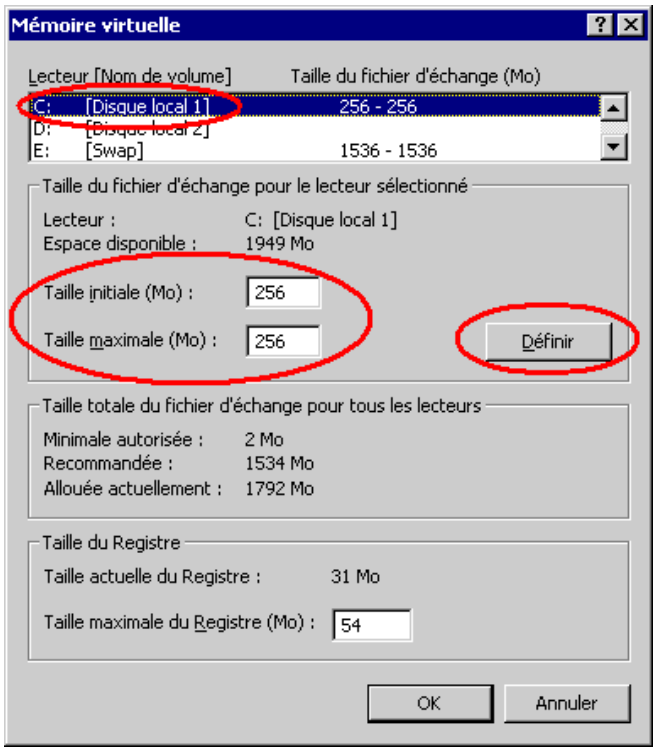
Le swap sous Windows est stocké dans un fichier à la taille variable.
Il faut régler la taille de ce fichier pour qu'elle ne varie pas.

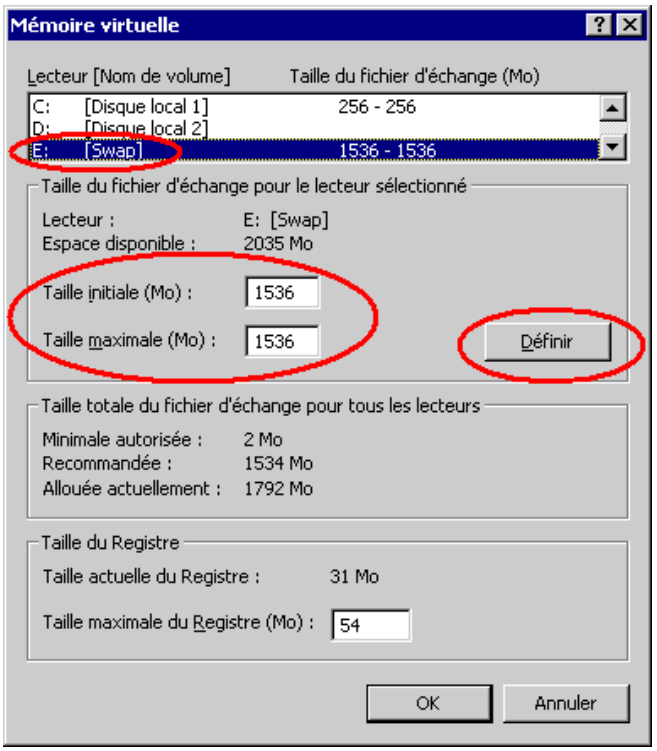






On peut utiliser des fichiers pour swapper au lieu de partitions.
Tous les systèmes UNIX ne le permettent pas.

Avantages :

- en cas de swap insuffisant, il est simple de créer un fichier de swap.

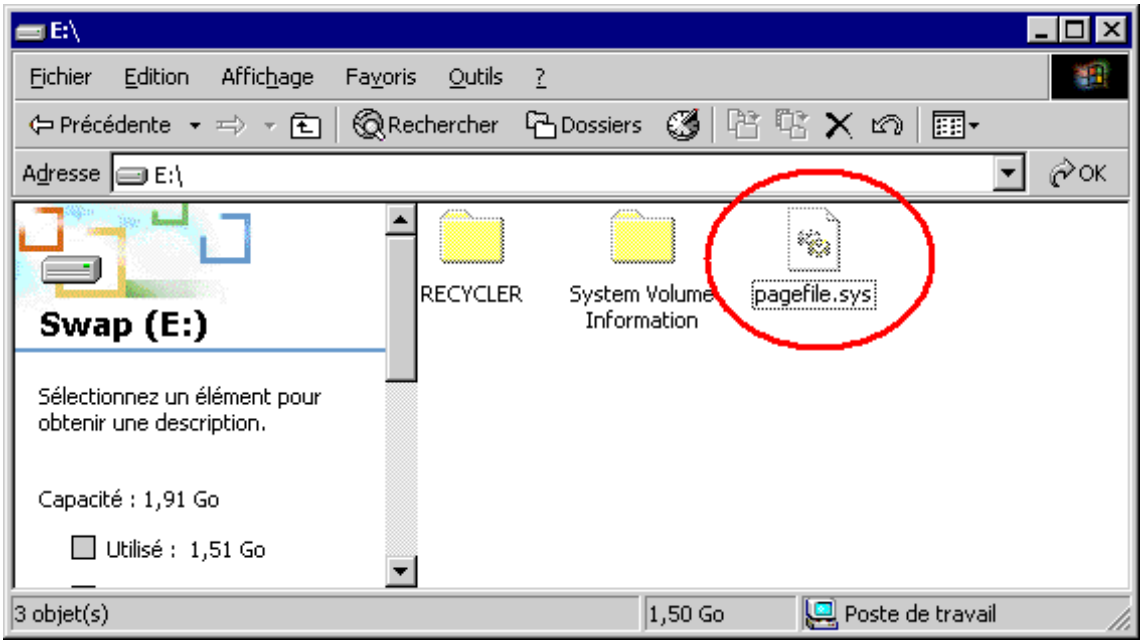
Inconvénients :

- une fois ajouté au swap, il est souvent impossible de faire machine arrière ; pour supprimer le fichier de swap, il faut rebooter.
- swap via le filesystem et non via en dessous ⇒ solution a priori lente/peu performante





Sous Windows 2000, il n'y a que des fichiers de swap :



Comment générer des fichiers de swap ?

Via la commande « mkfile » si elle est disponible sur l'UNIX :

```
# mkfile 50m grosfichier
# chmod 600 grosfichier
# ls -l grosfichier
-rw-------  1 root 52428800 Nov 16 01:27 grosfichier
```





Faire ensuite :

```
# swapon /chemin/vers/grosfichier
```

Les droits du fichier doivent être 600 pour ne pas lire dans le fichier de swap des informations sensibles.

Le fichier « /etc/fstab » peut servir à monter des fichiers de swap automatiquement au boot.

Commande `creatfil` du Resource Kit Windows 2000.
Syntaxe : `creatfil fichier nombre-de-blocs`
(la taille d'un bloc est de 1024 octets)

```
c:\>creatfil %TEMP%\grosfichier.dat 1024
c:\>dir %TEMP%\grosfichier.dat
 Volume in drive C is Windows XP
 Volume serial Number is 0C5C-E708

 Directory of C:\DOCUME~1\besancon\LOCALS~1\Temp

18/07/2004  12:35         1 048 576 grosfichier.dat
               1 File(s)      1 048 576 bytes
               0 Dir(s)   1 173 176 320 bytes free
```





Comment générer des fichiers de swap ?

Via la commande `dd` (en anglais *device to device*) :

```
# dd if=/dev/zero of=grosfichier bs=1M count=50
50+0 records in
50+0 records out
52428800 bytes transferred in 3.533640 secs (14837052 bytes/sec)
# chmod 600 grosfichier
# ls -l grosfichier
-rw-------  1 besancon 52428800 Nov 16 01:27 grosfichier
```

Faire ensuite :

```
# swapon /chemin/vers/grosfichier
```

Les droits du fichier doivent être `600` pour ne pas lire dans le fichier de swap des informations sensibles.

(en anglais *virtual memory statistics*)

Syntaxe : `vmstat [nombre-de-secondes [nombre-de-fois]]`

Exemple :

```
# vmstat 5 10
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr dd dd f0 s0   in   sy   cs us sy id
 0 0 4 1547856 126800 7   8 58  1  1  0  1  1  5  0  0  447  615  267 32  1 66
 0 0 19 596672 49568  0   1  0  0  0  0  0  0  0  0  0  403   40  112  1  2 96
 0 0 19 596672 49568  0   3  0  0  0  0  0  0  0  0  0  422  131  166  1  4 95
 0 0 19 592152 47584 50 205 102 0  0  0  0  1  0  0  0  405 2149  420 76 24  0
 0 0 19 589408 46472  0   1  0  0  0  0  0  0  1  0  0  405 2461  532 94  6  0
 0 0 19 589392 46264  0   0  0  0  0  0  0  5  2  0  0  416 2444  535 96  4  0
 0 0 19 589384 46088  0   0  0  0  0  0  0  0  1  0  0  403 1204  317 47  1 52
 0 0 19 595208 48880  0   1  0 14 14  0  0  0  0  0  0  403  396  172 14  1 85
 0 0 19 596672 49592  0   0  0  0  0  0  0  0  4  0  0  410   84  118  0  0 100
 0 0 19 596672 49592  0   0  0  0  0  0  0  0  0  0  0  403   33  117  0  0 100
```





Exemple :

```
# vmstat 5
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr dd dd f0 s0   in   sy   cs us sy id
 0 0 4 1547816 126800 7   8 58  1  1  0  1  1  5  0  0  447  615  267 32  1 66
 0 0 19 596392 48992  0   1  0  0  0  0  0  0  1  0  0  416  176  161  2  1 96
 0 0 19 595384 48736 66 315 233 0  0  0  0  9  4  0  0  428 1060  170 18  6 75
 0 0 19 589192 44464  5  23 193 0  0  0  0  3 18  0  0  444 2078  272 90  8  2
 1 0 19 589152 43536  2  14 120 0  0  0  0  8 12  0  0  442 1768  248 93  7  0
 0 0 19 589096 42648  2   5 149 0  0  0  0  1 17  0  0  438 2084  264 94  4  1
 1 0 19 589056 42040  1   1 56  0  0  0  0  1  7  0  0  417 1434  223 97  3  0
 1 0 19 589144 42984 41 195 112 24 24 0  0  2  5  0  0  416 2310  234 92  8  0
 1 0 19 589168 43288  0  13  2  2  2  0  0  8  2  0  0  421 1923  230 96  4  0
 1 0 19 589112 42504  2   7  0  0  0  0  0  0  1  0  0  406 2030  229 94  6  0
 1 0 19 589040 42056  1   1  0  2  2  0  0 12  1  0  0  427 1568  231 98  2  0
 0 0 19 592072 43896 16  71 172 35 35 0  0 12  4  0  0  434 1049  205 54  3 43
 0 0 19 596640 47032  0   0  0  0  0  0  0  0 15  0  0  498  190  286  0  0 99
 0 0 19 596640 47032  0   0  0  0  0  0  0  0  0  0  0  420   67  158  0  0 100
^C
```

Fichier « /proc/sys/vm/swappiness » : détermine quelle quantité de mémoire virtuelle sera mise en swap.

Par défaut : 60 ⇒ 60% des données seront mises dans le cache et 40 % seront stockées en RAM.

Chapitre 13

# *Processus et exécutables*





UNIX est synonyme de multi-tâches.
L'aspect multitâche se voit par la commande « `ps` ».

```
% ps -aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  1120   52 ?        S    Oct23   0:06 init
root         2  0.0  0.0     0    0 ?        SW   Oct23   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   Oct23   0:01 [kupdate]
...
nobody     476  0.0  0.1  1300   44 ?        S    Oct23   0:01 [identd]
daemon     490  0.0  0.0  1144    0 ?        SW   Oct23   0:00 [atd]
xfs        636  0.0  0.3  2820  120 ?        S    Oct23   0:18 xfs -droppriv -da
root     14703  0.0  0.0  2256    0 tty1     SW   Oct25   0:00 [login]
root      9813  0.0  0.0  6912    0 ?        SW   Oct31   0:09 [kdm]
idiri    20810  0.0  0.0  6552    0 ?        SW   15:13   0:01 [kwm]
idiri    20837  0.0  0.0  2140    0 ?        SW   15:13   0:00 [tcsh]
idiri    20863  0.0  0.0  1996    0 pts/0    SW   15:13   0:00 [tcsh]
besancon 21785  0.0  1.3  1732  416 pts/1    S    15:25   0:00 -bash
idiri    23600  0.0  0.0  1844    0 tty1     SW   16:26   0:00 [vi]
idiri    23660  0.2  1.5  1860  472 tty2     S    16:39   0:01 vi probleme6.c
...
```

Comment bien utiliser « ps » ?

Une bonne utilisation de « ps » repose sur la connaissance des différentes options employables.

La difficulté est que les options sont différentes selon que le système est de type BSD ou AT&T. On retiendra :

(BSD) `ps -aux` ≡ `ps -edf` (AT&T)





◇ **ps** : les processus associés à son terminal

```
% ps
  PID  TT  STAT      TIME COMMAND
24578  p7  Ss     0:00.33 -bash (bash-2.01)
24600  p7  S      0:00.52 xdvi.bin partie-03.dvi
24602  p7  I      0:00.99 gs -sDEVICE=x11 -dNOPAUSE -dSAFER -q - (gs-5.10)
24876  p7  R+     0:00.00 ps
```

◇ **ps -x** : tous ses processus

```
% ps -x
  PID  TT  STAT      TIME COMMAND
24578  p7  Ss     0:00.34 -bash (bash-2.01)
24600  p7  S      0:00.52 xdvi.bin partie-03.dvi
24602  p7  I      0:00.99 gs -sDEVICE=x11 -dNOPAUSE -dSAFER -q - (gs-5.10)
24899  p7  R+     0:00.00 ps -x
24881  p8  Ss+    0:00.27 -bash (bash-2.01)
```

## ◇ ps -ax : tous les processus de la machine

```
% ps -ax
  PID  TT  STAT      TIME COMMAND
    0  ??  DLs    0:00.59  (swapper)
    1  ??  Ss     0:01.18 /sbin/init --
    2  ??  DL     1:25.50  (pagedaemon)
    3  ??  DL     0:00.00  (vmdaemon)
    4  ??  DL     2:35.03  (update)
...
24496  p6  I+     0:00.01 rlogin alpha
24578  p7  Ss     0:00.35 -bash (bash-2.01)
24600  p7  S      0:00.55 xdvi.bin partie-03.dvi
24602  p7  I      0:00.99 gs -sDEVICE=x11 -dNOPAUSE -dSAFER -q - (gs-5.10)
24935  p7  R+     0:00.00 ps -ax
24881  p8  Is+    0:00.27 -bash (bash-2.01)
 3869  p9  Is     0:00.16 -tcsh (tcsh-6.07)
...
```





## ◇ ps -aux : tous les processus de la machine avec les noms de login associés

```
% ps -aux
USER       PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED      TIME COMMAND
satish   24499  3.7  5.8  9320 7504  p5  SN    4:31PM   0:39.34 /usr/local/net
root         1  0.0  0.1   484   76  ??  Is   Wed04PM   0:01.18 /sbin/init --
root         2  0.0  0.0     0   12  ??  DL   Wed04PM   1:25.50  (pagedaemon)
root         3  0.0  0.0     0   12  ??  DL   Wed04PM   0:00.00  (vmdaemon)
...
besancon 24578  0.0  0.7   784  944  p7  Ss    4:44PM   0:00.36 -bash (bash-2.
besancon 24600  0.0  1.9  1108 2456  p7  S     4:44PM   0:00.60 xdvi.bin parti
```

### ◇ ps -w

Parfois la commande est tronquée par « ps -ax » :

```
% ps -ax
  PID  TT  STAT      TIME COMMAND
...
  368  ??  Is     0:04.36 sendmail: accepting connections on port 25
(sendmail-
```

On utilise alors « ps -axw » :

```
% ps -axw
...
  368  ??  Is     0:04.36 sendmail: accepting connections on port 25
(sendmail-8.8.6)
```

◇ **ps -ww** Encore plus de caractères affichés par rapport à « ps -w ».





### ◇ ps -e : tous les processus de la machine

```
% ps -e
   PID TTY      TIME CMD
     0 ?        0:01 sched
     1 ?        0:02 init
     2 ?        0:00 pageout
     3 ?        0:52 fsflush
   181 ?        0:06 automoun
   306 ?        0:00 sac
    49 ?        0:00 devfseve
    51 ?        0:00 devfsadm
   123 ?        0:01 rpcbind
   137 ?        0:01 rpc.nisd
...
```

### ◇ ps -l : affichage sous forme longue

```
% ps -l
 F S   UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN TTY      TIME CMD
 8 S   999 16463 16461  0  40 20        ?    172        ? pts/0    0:00 csh
```

### ◇ ps -f : full listing

```
% ps -edf
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0 09:09:47 ?        0:01 sched
    root     1     0  0 09:09:47 ?        0:02 /etc/init -
    root     2     0  0 09:09:47 ?        0:00 pageout
    root     3     0  0 09:09:47 ?        0:52 fsflush
    root   181     1  0 09:12:07 ?        0:06 /usr/lib/autofs/automountd
...
  daemon   283     1  0 09:12:12 ?        0:11 /usr/sbin/lpd
    root   291     1  0 09:12:13 ?        0:00 /usr/local/apache/bin/httpd
    root   296     1  0 09:12:14 ?        0:00 /usr/local/admin/lib/idled
  nobody 15130   291  0 23:30:56 ?        0:00 /usr/local/apache/bin/httpd
besancon 16463 16461  0 00:12:26 pts/0    0:00 -csh
...
```





La commande « ps » permet d'afficher parfois les variables d'environnement d'un processus.

- Non disponible sur Solaris
- Disponible sous FreeBSD : utiliser l'option « -e »
- Disponible sous LINUX : utiliser l'option « -e »

Exemples pages suivantes.

## ◇ FreeBSD

```
% ps
  PID  TT  STAT      TIME COMMAND
...
47935  p2  Is     0:00.01 -bash (bash)
...

% ps e
  PID  TT  STAT      TIME COMMAND
...
47923  p1  Is     0:00.01 USER=besancon LOGNAME=besancon HOME=/home/ad
min/besanTronqué !
...

% ps ew
  PID  TT  STAT      TIME COMMAND
...
47923  p1  Is     0:00.01 USER=besancon LOGNAME=besancon HOME=/home/ad
min/besancon MAIL=/var/mail/besancon PATH=/sbin:/bin:/usr/sbiTronqué !
...
```





## Suite...

```
% ps eww
  PID  TT  STAT      TIME COMMAND
...
47923  p1  Is     0:00.01 USER=besancon LOGNAME=besancon HOME=/home/ad
min/besancon MAIL=/var/mail/besancon PATH=/sbin:/bin:/usr/sbin:/usr/bi
n:/usr/games:/usr/local/sbin:/usr/local/bin:/usr/X11R6/bin:/home/admin
/besancon/bin TERM=vt100 BLOCKSIZE=K FTP_PASSIVE_MODE=YES SHELL=/bin/b
ash SSH_CLIENT=134.157.13.223 33022 22 SSH_CONNECTION=134.157.13.223 3
3022 134.157.13.123 22 SSH_TTY=/dev/ttyp1 DISPLAY=localhost:10.0 SSH_A
UTH_SOCK=/tmp/ssh-rcv9LXt0N9/agent.47922 -bash (bash)
...
```

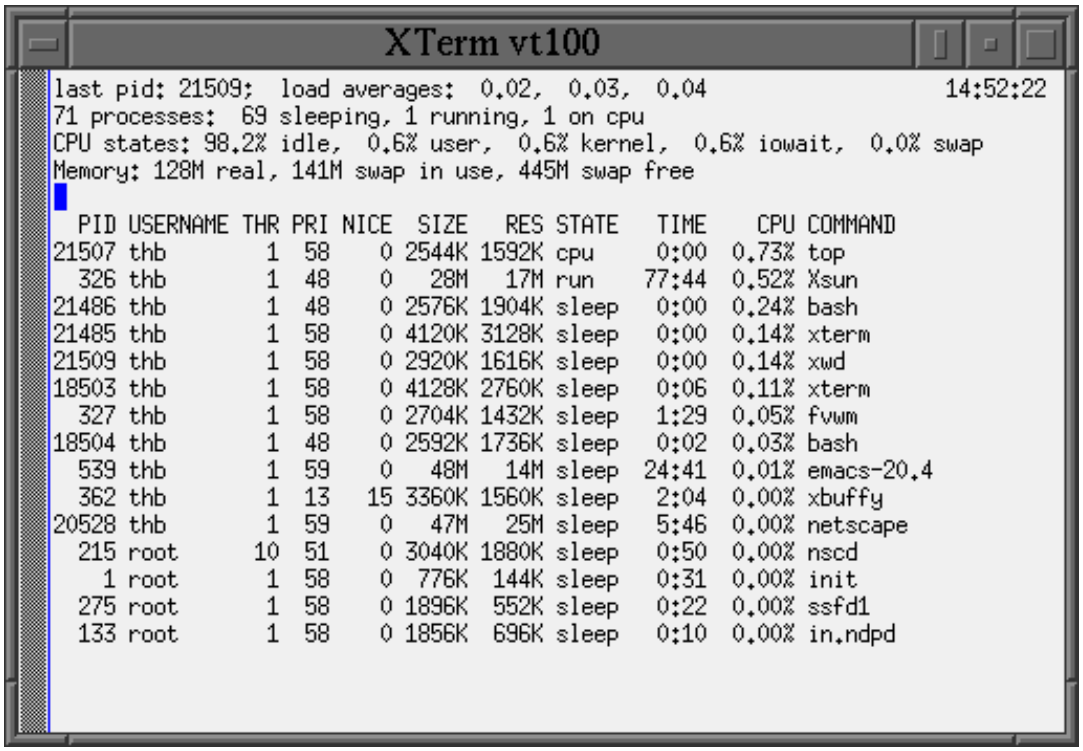
Inconvénient de « ps » : c'est la liste des processus à un instant $t$.
On ne pourra jamais sous UNIX avoir la liste des processus en cours : le temps de chercher les processus et de faire le rapport à l'écran, certains processus peuvent avoir disparu.

Amélioration de « ps » : la commande « top » qui n'est cependant pas standard sur tous les UNIX.
Son intérêt : elle affiche une liste des processus toutes les N secondes

URL : `ftp://ftp.groupsys.com/pub/top/`







La commande « **kill** » sert à communiquer avec des processus :

- arrêt de processus
- demande au processus de se reconfigurer
- passage en mode verbeux du processus
- etc.

La commande « **kill** » existe sur tous les UNIX et il n'y a pas de différence de fonctionnement selon les UNIX.

2 syntaxes possibles :

- syntaxe numérique : « `kill -9 2878` »
- syntaxe symbolique : « `kill -KILL 2878` »





| Nom en langage C | Nom pour la commande `kill` | Valeur | Comportement | Sens |
|---|---|---|---|---|
| SIGHUP | HUP | 1 | Exit | Hangup |
| SIGINT | INT | 2 | Exit | Interrupt |
| SIGQUIT | QUIT | 3 | Core | Quit |
| SIGILL | ILL | 4 | Core | Illegal Instruction |
| SIGTRAP | TRAP | 5 | Core | Trace or Breakpoint Trap |
| SIGABRT | ABRT | 6 | Core | Abort |
| SIGEMT | EMT | 7 | Core | Emulation Trap |
| SIGFPE | FPE | 8 | Core | Arithmetic Exception |
| SIGKILL | KILL | 9 | Exit | Killed |
| SIGBUS | BUS | 10 | Core | Bus Error |
| SIGSEGV | SEGV | 11 | Core | Segmentation Fault |
| SIGSYS | SYS | 12 | Core | Bad System Call |
| SIGPIPE | PIPE | 13 | Exit | Broken Pipe |
| SIGALRM | ALRM | 14 | Exit | Alarm Clock |
| SIGTERM | TERM | 15 | Exit | Terminated |
| SIGUSR1 | USR1 | 16 | Exit | User Signal 1 |

| Nom en langage C | Nom pour la commande kill | Valeur | Comportement | Sens |
|---|---|---|---|---|
| SIGUSR2 | USR2 | 17 | Exit | User Signal 2 |
| SIGCHLD | CHLD | 18 | Ignore | Child Status Changed |
| SIGPWR | PWR | 19 | Ignore | Power Fail or Restart |
| SIGWINCH | WINCH | 20 | Ignore | Window Size Change |
| SIGURG | URG | 21 | Ignore | Urgent Socket Condition |
| SIGPOLL | POLL | 22 | Exit | Pollable Event |
| SIGSTOP | STOP | 23 | Stop | Stopped (signal) |
| SIGTSTP | TSTP | 24 | Stop | Stopped (user) |
| SIGCONT | CONT | 25 | Ignore | Continued |
| SIGTTIN | TTIN | 26 | Stop | Stopped (tty input) |
| SIGTTOU | TTOU | 27 | Stop | Stopped (tty output) |
| SIGVTALRM | VTALRM | 28 | Exit | Virtual Timer Expired |
| SIGPROF | PROF | 29 | Exit | Profiling Timer Expired |
| SIGXCPU | XCPU | 30 | Core | CPU time limit exceeded |
| SIGXFSZ | XFSZ | 31 | Core | File size limit exceeded |
| SIGWAITING | WAITING | 32 | Ignore | Concurrency signal reserved by threads library |





Les signaux les plus utiles sont :

- SIGHUP
  Cela envoie l'équivalent du Ctrl-C du clavier.

- SIGKILL
  Cela envoie un signal que le processus est obligé de suivre et qui se traduira inélectablement par la mort du processus.

(en anglais *processus grep*)

Permet de rechercher des processus désignés par leur nom ou par d'autres attributs.

Syntaxe : `pgrep [options] regexp`

Options intéressantes :

- option « `-f` » : recherche parmi la commande et ses paramètres
- option « `-l` » : affiche PID + nom de la commande (+ paramètres si option « `-f` »)
- option « `-u username` »
- option « `-u username1[,username2,etc]` »

En gros équivalent de « `ps -aux | grep regexp` » mais plus pratique.

Disponible sur LINUX, SOLARIS





◇ Exemple 1

```
% pgrep -f mingetty
3778
3779
3780
3781
3782
3783
```

◇ Exemple 2

```
% pgrep -f -l mingetty
3778 /sbin/mingetty tty1
3779 /sbin/mingetty tty2
3780 /sbin/mingetty tty3
3781 /sbin/mingetty tty4
3782 /sbin/mingetty tty5
3783 /sbin/mingetty tty6
```

(en anglais *processus kill*)

Permet d'envoyer des signaux à des processus désignés par leur nom ou par d'autres attributs.

Syntaxe : `pkill [options] regexp`

Options intéressantes :

- option « `-f` » : recherche parmi la commande et ses paramètres
- option « `-u username` »
- option « `-u username[,username,etc]` »
- option « `-signal` » : numéro du signal ou nom du signal (voir page 732)

Disponible sur LINUX, SOLARIS





◇ Exemple

```
# pkill -TERM -u clamav
```

(en anglais *kill all*)
Syntaxe : `killall [options] nom`
Options intéressantes :

- option « `-i` » : attente de réponse oui/non pour tuer le processus
- option « `-signal` » : numéro du signal ou nom du signal (voir page 732)

```
# pgrep -l -u apache
3664 httpd
3665 httpd
3666 httpd
3667 httpd
3668 httpd
3669 httpd

# killall httpd
# pgrep -l -u apache
#   <--- les processus httpd sont bien morts
```





Un processus UNIX posséde une priorité d'exécution : son « nice number »

- $-20 \leq$ nice number $\leq 20$
- $-20$ correspond à la priorité maximale
- 20 correspond à la priorité minimale

Un utilisateur peut descendre la priorité de ses processus.
Seul l'administrateur peut augmenter la priorité de n'importe quel processus.

Pour lancer un processus à la priorité N (N pouvant être négatif) :
« `nice [-N] commande` »

Une fois un processus démarré, on peut changer sa priorité par la commande : « `renice priorité PID` »

Exemple de la commande « nice » lancée par un utilisateur normal :

```
% nice -10 ./programme
% ps -aux
USER       PID %CPU %MEM    SZ   RSS TT   STAT CPU START   TIME COMMAND
besancon 17641 94.6  0.1    16   144 p1     R N   0 23:42    0:07 ./programme
...
% ps -ax
  PID TT   STAT CPU TIME COMMAND
...
17641 p1    R N   2  0:47 ./programme

% nice --10 ./programme
nice: cannot set priority: Permission denied
```





Exemple de la commande « nice » lancée par l'administrateur :

```
# nice --10 ./programme
# ps -ax
  PID TT   STAT CPU TIME COMMAND
...
17663 p1    R <   0  0:04 ./programme
```

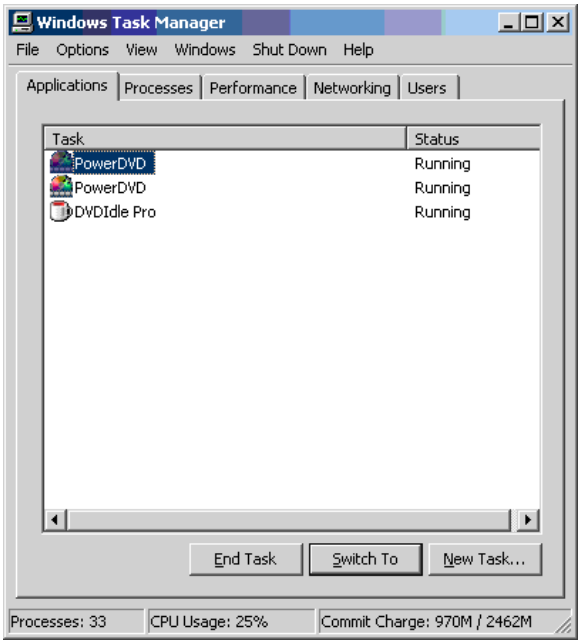
Exemple de la commande « renice » :

```
% nice -10 ./programme
% ps -al
      F   UID   PID  PPID CP PRI NI  SZ  RSS WCHAN   STAT CPU TT  TIME CC
...
20008001 4332 17665   224211  97 10  16  144           R N  2 p1  0:29 ./
...
% renice 15 17665
17665: old priority 10, new priority 15
% ps -al
      F   UID   PID  PPID CP PRI NI  SZ  RSS WCHAN   STAT CPU TT  TIME CC
...
20008001 4332 17665   224255 102 15  16  144           R N  0 p1  1:44 ./
...
```

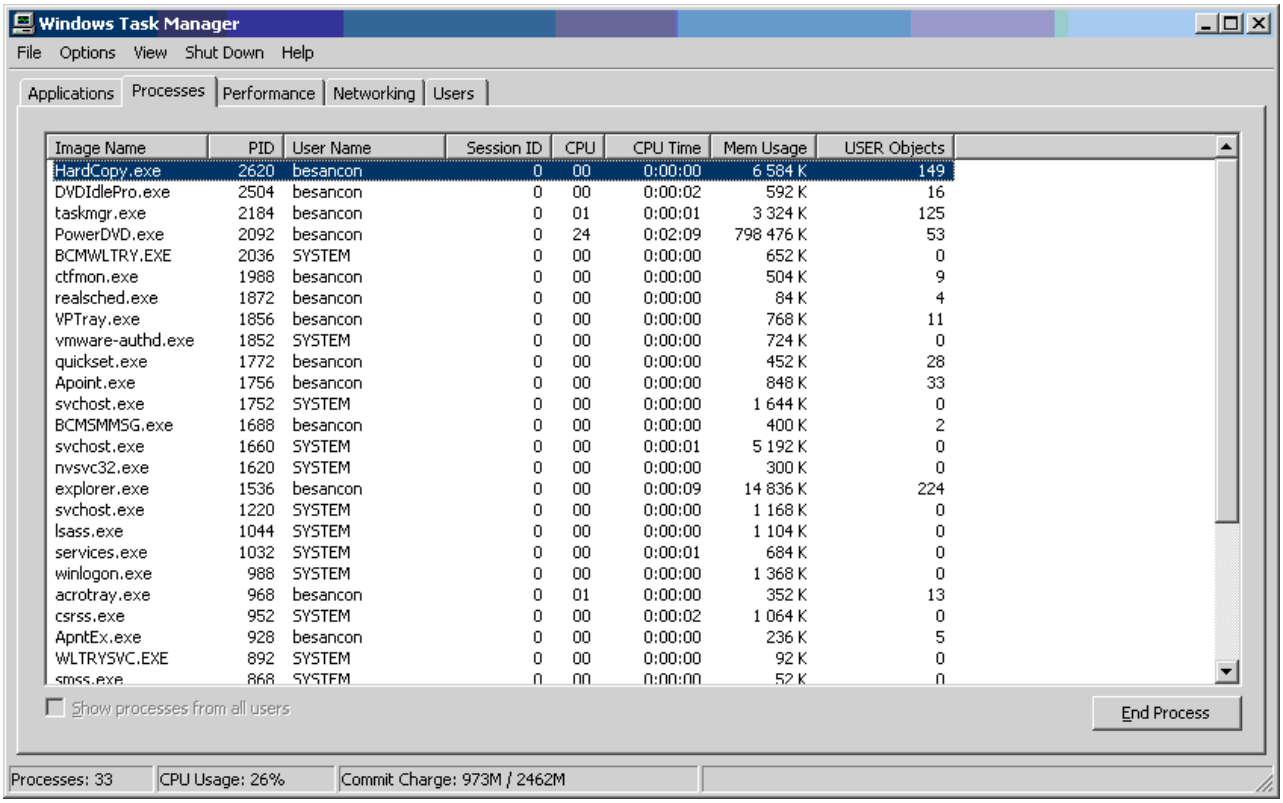




Utilitaire taskmgr.exe de chez Microsoft :

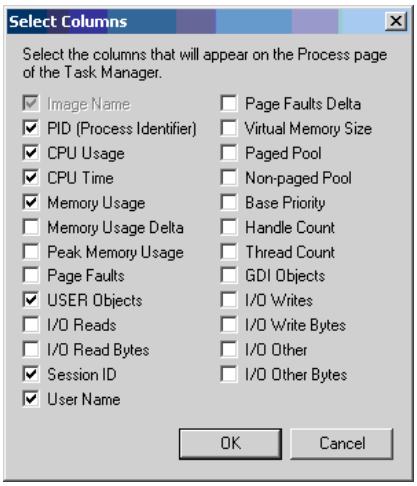


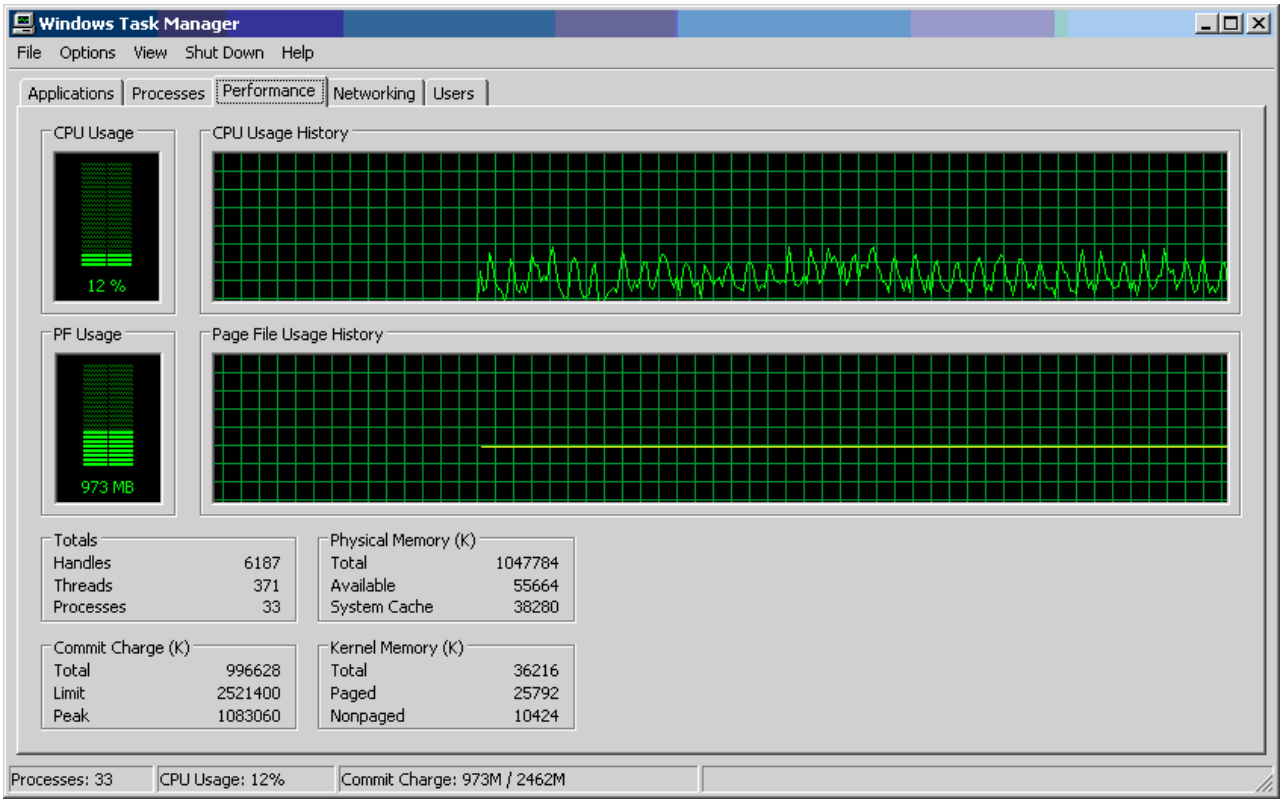

C'est le programme appelé via Ctrl - Alt - Del


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Image Name PID User Name Session ID CPU CPU Time Mem Usage USER Objects
HardCopy.exe 2620 besancon 0 00 0:00:00 6 584 K 149
DVDIdlePro.exe 2504 besancon 0 00 0:00:02 592 K 16
taskmgr.exe 2184 besancon 0 01 0:00:01 3 324 K 125
PowerDVD.exe 2092 besancon 0 24 0:02:09 798 476 K 53
BCMWLTRY.EXE 2036 SYSTEM 0 00 0:00:00 652 K 0
ctfmon.exe 1988 besancon 0 00 0:00:00 504 K 9
realsched.exe 1872 besancon 0 00 0:00:00 84 K 4
VPTray.exe 1856 besancon 0 00 0:00:00 768 K 11
vmware-authd.exe 1852 SYSTEM 0 00 0:00:00 724 K 0
quickset.exe 1772 besancon 0 00 0:00:00 452 K 28
Apoint.exe 1756 besancon 0 00 0:00:00 848 K 33
svchost.exe 1752 SYSTEM 0 00 0:00:00 1 644 K 0
BCMSMMSG.exe 1688 besancon 0 00 0:00:00 400 K 2
svchost.exe 1660 SYSTEM 0 00 0:00:01 5 192 K 0
nvsvc32.exe 1620 SYSTEM 0 00 0:00:00 300 K 0
explorer.exe 1536 besancon 0 00 0:00:09 14 836 K 224
svchost.exe 1220 SYSTEM 0 00 0:00:00 1 168 K 0
lsass.exe 1044 SYSTEM 0 00 0:00:00 1 104 K 0
services.exe 1032 SYSTEM 0 00 0:00:01 684 K 0
winlogon.exe 988 SYSTEM 0 00 0:00:00 1 368 K 0
acrotray.exe 968 besancon 0 01 0:00:00 352 K 13
csrss.exe 952 SYSTEM 0 00 0:00:02 1 064 K 0
ApntEx.exe 928 besancon 0 00 0:00:00 236 K 5
WLTRYSVC.EXE 892 SYSTEM 0 00 0:00:00 92 K 0
smss.exe 868 SYSTEM 0 00 0:00:00 52 K 0
Show processes from all users
End Process
Processes: 33
CPU Usage: 26%
Commit Charge: 973M / 2462M





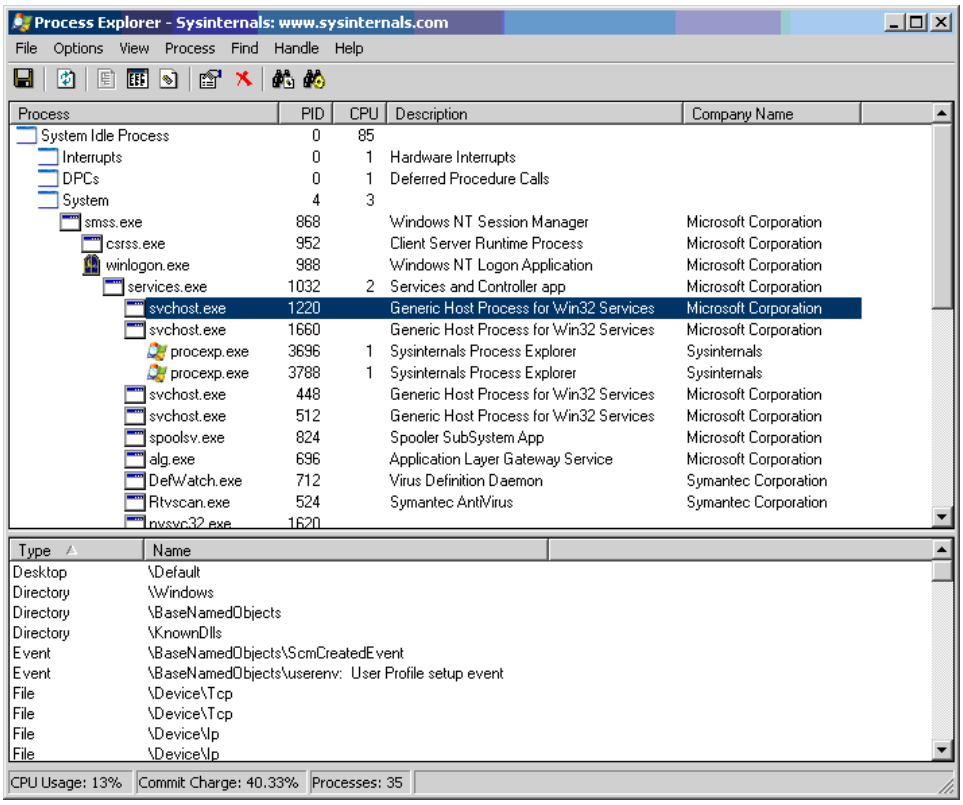

Select Columns
Select the columns that will appear on the Process page of the Task Manager.
Image Name
PID (Process Identifier)
CPU Usage
CPU Time
Memory Usage
Memory Usage Delta
Peak Memory Usage
Page Faults
USER Objects
I/O Reads
I/O Read Bytes
Session ID
User Name
Page Faults Delta
Virtual Memory Size
Paged Pool
Non-paged Pool
Base Priority
Handle Count
Thread Count
GDI Objects
I/O Writes
I/O Write Bytes
I/O Other
I/O Other Bytes
OK
Cancel








Utilitaire « `procexp.exe` » de chez `http://www.sysinternals.com` :



Utilitaires `pstools` de chez `http://www.sysinternals.com` :

- `PsExec.exe` - execute processes remotely
- `PsFile.exe` - shows files opened remotely
- `PsGetSid.exe` - display the SID of a computer or a user
- `PsKill.exe` - kill processes by name or process ID
- `PsInfo.exe` - list information about a system
- `PsList.exe` - list detailed information about processes
- `PsLoggedOn.exe` - see who's logged on locally and via resource sharing
- `PsLogList.exe` - dump event log records
- `PsService.exe` - view and control services
- `PsShutdown.exe` - shuts down and optionally reboots a computer
- `PsSuspend.exe` - suspends processes
- `PsUptime.exe` - shows you how long a system has been running since its last reboot



# Chapitre 13 • Processus et exécutables

## §13.12 • Temps d'exécution d'un processus : `time`

A un programme sont associés :

- le temps d'exécution de code de bas niveau du système (par exemple écrire sur disque)
- le temps d'exécution de code de haut niveau de l'utilisateur (par exemple calculer une fonction mathématique)

La commande « `time` » fournit ces temps.
Syntaxe : `time commande parametres`

Par exemple :

```
% time gzip -v archive.tar
archive.tar:                 26.8% -- replaced with archive.tar.gz

real    0m0.485s
user    0m0.390s
sys     0m0.070s
```

Attention, subtilité :

- il existe un builtin du shell appelé « time »
- il existe une commande « time » : « /bin/time »

```
% echo $SHELL
/bin/bash
% type time
time is a shell keyword
% time gzip -v archive.tar
archive.tar:                    26.8% -- replaced with w.tar.gz

real    0m0.485s
user    0m0.390s
sys     0m0.070s

% /bin/time gzip -v archive.tar
archive.tar:                    26.8% -- replaced with w.tar.gz

real        0.4
user        0.3
sys         0.0
```





Pour obtenir la charge du système, utiliser la commande « uptime » :

```
% uptime
 11:15pm  up  4:44,  4 users,  load average: 0.12, 0.02, 0.00
```

Quels sont les renseignements renvoyés :

- **le temps de fonctionnement du système**
- **le nombre d'utilisateurs connectés**
- **la charge du système** : 3 nombres représentant :
  - la moyenne de processus actifs durant la minute précédente
  - la moyenne de processus actifs durant les 5 dernières minutes
  - la moyenne de processus actifs durant les 15 dernières minutes

Difficulté de définir une valeur «normale» :

- de nombreux processus en attente d'une entrée sortie donnent un fort indice de charge alors que le système est en parfaite mesure de répondre efficacement
- un seul processus sollicitant fortement la mémoire virtuelle de la machine en swappant sans cesse peut à lui seul empêcher la machine de répondre alors que l'indice de charge est bas
- on traite indifféremment les processus renicés et ceux qui ne le sont pas

On prendra comme valeur «normale» une valeur de 3.

On préférera les renseignements renvoyés par « `ps` » pour se faire une idée de la charge réélle du système.





Pour information :

| Taux de disponibilité | Durée d'indisponibilité |
|---|---|
| 97 % | 11 jours |
| 98 % | 7 jours |
| 99 % | 3 jours et 15 heures |
| 99.9 % | 8 heures et 48 minutes |
| 99.99 % | 53 minutes |
| 99.999 % | 5 minutes |
| 99.9999 % | 32 secondes |

Il arrive occasionnellement qu'un processus soit non tuable par la commande « `kill` » même avec le signal `SIGKILL` (-9).

Ces processus tombent dans 3 catégories :

- Processus en attente d'écriture à travers NFS
- Processus en attente d'opération physique sur un périphérique
- Processus zombie, marqués par un « **Z** » ou « <**defunct**> » lorsque l'on fait « `ps` », généré lorsqu'un processus se termine et ne reçoit pas à ce moment là d'acknowledgement de la part de son parent (par exemple parce que le processus père est déjà mort).





Les processus zombies ne sont pas gênant : ils ne consomment plus de ressouces. Ils disparaitront au prochain reboot de la machine.

La seule gêne est la consommation d'une entrée processus dans le tableau interne du noyau de gestion des processus.

## Programme C de création de processus zombie (auteur : Pierre.David@crc.u-strasbg.fr)

```
main (int argc, char *argv [])
{
    switch (fork ())
    {
        case -1 :       /* erreur */
            perror ("fork") ;
            exit (1) ;
        case 0 :        /* fils */
            exit (0) ;  /* => zombification */
        default :       /* père */
            pause () ;  /* => père indigne : il se fiche de la mort du fils */
    }
}
```

## L'exécution du programme donne :

```
% ./zombie &
[1] 16274
% ps -edf | grep 16274
besancon 16274 18533  0 17:31:44 pts/3    0:00 ./zombie
besancon 16275 16274  0                   0:00 <defunct>
```

Le processus 16275 est un processus zombie.





## Commande SOLARIS pour tuer les processus zombies :

```
% ps -edf
     UID   PID  PPID  C    STIME TTY      TIME CMD
...
besancon 19763 13971  0 08:32:31 pts/7    0:00 ./zombie
besancon 19764 19763  0                   0:00 <defunct>
...

% ls -l /usr/bin/preap
-r-xr-xr-x  42 root     bin         5416 Apr  7  2002 /usr/bin/preap

% preap 19764
19764: exited with status 0

% ps -edf | grep 19764
     UID   PID  PPID  C    STIME TTY      TIME CMD
...
besancon 19763 13971  0 08:32:31 pts/7    0:00 ./zombie
```

Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setuid** (**4000** en octal).

Avec ce bit positionné, le programme est exécuté avec les droits de l'utilisateur propriétaire.

```
% ls -lg prog1.exe prog2.exe
-rwxr-xr-x   1 besancon ars      249 Sep 20 22:43 prog1.exe
-rwxr-xr-x   1 besancon ars      249 Sep 20 22:43 prog2.exe
% chmod u+s prog1.exe
% chmod 4711 prog2.exe
% ls -lg prog1.exe prog2.exe
-rwsr-xr-x   1 besancon ars      249 Sep 20 22:43 prog1.exe
-rws--x--x   1 besancon ars      249 Sep 20 22:43 prog2.exe
```





◇ Exemple : commande `passwd`

Les fichiers contenant les informations sur les comptes utilisateurs sont :

```
-rw-r--r--   1 root     sys          946 Sep 21 22:36 /etc/passwd
-rw-------   1 root     sys          488 Jun 21 22:43 /etc/shadow
```

Notez leurs droits !
Pourtant un simple utilisateur change son mot de passe et modifie donc ces fichiers !
A PRIORI, IMPOSSIBLE ! ! ! et pourtant si...

Explication : bit setuid de la commande `passwd` :

```
-r-sr-sr-x   1 root     sys        21964 Apr  7  2002 /bin/passwd
```

Attention à l'affichage du bit setuid !
Classiquement :

```
% gcc exemple.c -o exemple.exe
% ls -l exemple.exe
-rwxr-xr-x   1 besancon ars      6204 Jan 24 20:22 programme.exe
% chmod u+s exemple.exe
% ls -l exemple.exe
-rwsr-xr-x   1 besancon ars      6204 Jan 24 20:22 programme.exe
```





Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars         0 Jan 24 20:21 exemple.txt
% chmod u+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required for
set-ID on execution
% ls -l exemple.txt
-rw-r--r--   1 besancon ars         0 Jan 24 20:21 exemple.txt
% chmod 4644 exemple.txt
% ls -l exemple.txt
-rwSr--r--   1 besancon ars         0 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « S » ≡ « bit 04000 seul »
- affichage « s » ≡ « bit x + bit S »

# ATTENTION : le bit setuid ne fonctionne pas avec un shell script.

**Il ne fonctionne qu'avec un exécutable binaire.**

Une solution sera proposée dans le tôme 3 (commande « `sudo` »).





Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setgid** (**2000** en octal).

Avec ce bit positionné, le programme est exécuté avec les droits du groupe propriétaire

```
% ls -lgF prog1.exe prog2.exe
-rwxr-xr-x   1 besancon ars     249 Sep 20 22:43 prog1.exe
-rwxr-xr-x   1 besancon ars     249 Sep 20 22:43 prog2.exe
% chmod g+s prog1.exe
% chmod 2711 prog2.exe
% ls -lgF prog1.exe prog2.exe
-rwxr-sr-x   1 besancon ars     249 Sep 20 22:43 prog1.exe
-rwx--s--x   1 besancon ars     249 Sep 20 22:43 prog2.exe
```

◇ Exemple : commande `write`

La console d'une machine UNIX appartient à l'utilisateur qui se connecte dessus :

```
crw--w----   1 besancon tty        0,   0 Aug 28 13:06 /dev/console
```

Notez leurs droits !
Pourtant un simple utilisateur peut écrire sur la console via la commande `write` (si l'on est en `mesg y`) !
A PRIORI, IMPOSSIBLE ! ! ! et pourtant si...

Explication : bit setgid de la commande `write` :

```
-r-xr-sr-x   1 root     tty        11484 Apr  7  2002 /usr/bin/write
```





Attention à l'affichage du bit setgid !
Classiquement :

```
% gcc exemple.c -o exemple.exe
% ls -l exemple.exe
-rwxr-xr-x   1 besancon ars      6204 Jan 24 20:22 exemple.exe
% chmod g+s exemple.exe
% ls -l exemple.exe
-rwxr-sr-x   1 besancon ars      6204 Jan 24 20:22 exemple.exe
```

Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon ars         0 Jan 24 20:21 exemple.txt
% chmod g+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required for set-I
% ls -l exemple.txt
-rw-r--r--   1 besancon ars         0 Jan 24 20:21 exemple.txt
% chmod 2644 exemple.txt
% ls -l exemple.txt
-rw-r-lr--   1 besancon ars         0 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « l » ≡ « bit 02000 seul »
- affichage « s » ≡ « bit x + bit l »





# ATTENTION : le bit setgid ne fonctionne pas avec un shell script.

**Il ne fonctionne qu'avec un exécutable binaire.**

Une solution sera proposée dans le tôme 3 (commande « `sudo` »).

`/proc` ≡ **system process**
Il contient une représentation sous forme de fichiers virtuels et directories virtuels d'informations de la machine UNIX.

Utilité ?
⇒ pour le programmeur système qui utilisera les appels standard (et non pas des appels spécifiques comme la `libkvm` ou des `ioctl()` spécifiques)

Implémentation de « `/proc` » propre à chaque UNIX.

Représentation virtuelle de :

- informations sur les processus
- informations sur le hardware
- informations sur le noyau





◇ Exemple : informations sur les processus

```
% echo $$
23247
% cd /proc/23247
% ls -l
total 0
-r--r--r--   1 besancon sysadmin    0 Sep 22 14:30 cmdline
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 cwd -> [0001]:15235153
-r--------   1 besancon sysadmin    0 Sep 22 15:45 environ
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 exe -> [0801]:10251
dr-x------   1 besancon sysadmin    0 Sep 22 15:45 fd
pr--r--r--   1 besancon sysadmin    0 Sep 22 15:45 maps
-rw-------   1 besancon sysadmin    0 Sep 22 15:45 mem
lrwx------   1 besancon sysadmin   64 Sep 22 15:45 root -> [0801]:2
-r--r--r--   1 besancon sysadmin    0 Sep 22 14:29 stat
-r--r--r--   1 besancon sysadmin    0 Sep 22 15:45 statm
-r--r--r--   1 besancon sysadmin    0 Sep 22 15:45 status
```

## ◇ Exemple : informations sur le hardware

```
% ls -l /proc/pci
-r--r--r--    1 root     root            0 Nov  2 18:52 /proc/pci

% cat pci
PCI devices found:
  Bus  0, device   0, function  0:
    Host bridge: VIA Technologies, Inc. VT82C693A/694x [Apollo PRO133x] (rev 196).
      Prefetchable 32 bit memory at 0xd0000000 [0xd3ffffff].
  Bus  0, device   1, function  0:
    PCI bridge: VIA Technologies, Inc. VT82C598/694x [Apollo MVP3/Pro133x AGP] (re
      Master Capable.  No bursts.  Min Gnt=12.
...
```





```
% cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 8
model name      : Pentium III (Coppermine)
stepping        : 10
cpu MHz         : 1002.214
cache size      : 256 KB
Physical processor ID   : 0
Number of siblings      : 1
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pa
bogomips        : 1989.22
...
```

## ◇ Exemple : informations sur le noyau et sa version

```
% cat /proc/version
Linux version 2.4.18-27.7.xsmp (bhcompile@stripples.devel.redhat.com)
(gcc version 2.96 20000731 (Red Hat Linux 7.3 2.96-112)) #1 SMP Fri
Mar 14 05:52:30 EST 2003
```

## ◇ Exemple : informations sur le uptime du noyau

```
% cat /proc/uptime
810311.03 800120.37
```

## ◇ Exemple : informations sur des statistiques du noyau sur l'emploi de la machine

```
% cat /proc/stat
cpu  1276780 2094 1821586 158965456
cpu0 681135 1171 903015 79447636
cpu1 595645 923 918571 79517818
page 19636553 18178406
swap 3965 4959
intr 460335474 414888745 32 0 4 4 3635073 6 0 1 0 35855460 1382047 897324 0 11 367
disk_io: (3,0):(3,3,24,0,0) (22,2):(695266,373120,8363128,322146,6352160) (22,3):(
ctxt 233821067
btime 1066985101
processes 201147
```





## ◇ Exemple : informations sur la table ARP du noyau

```
% cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
134.157.46.254   0x1         0x2         00:02:7E:21:F7:9C     *        eth0
192.168.4.18     0x1         0x2         00:48:54:6B:E5:B0     *        eth3
```

## ◇ Exemple : informations sur les sockets unix du noyau

```
% cat /proc/net/unix
Num       RefCount Protocol Flags    Type St Inode Path
db7f5580: 00000002 00000000 00010000 0001 01  9782 /tmp/.font-unix/fs7100
e9fcc580: 00000002 00000000 00010000 0001 01  9940 /tmp/.X11-unix/X0
d3e23a80: 00000002 00000000 00010000 0001 01  8308 /var/lib/mysql/mysql.sock
ce1dd080: 00000002 00000000 00010000 0001 01  9600 /dev/gpmctl
d75f5080: 00000014 00000000 00000000 0002 01  5050 /dev/log
cb91e080: 00000002 00000000 00000000 0002 01 1088373
d4f46080: 00000002 00000000 00000000 0002 01 1087696
f679a580: 00000002 00000000 00000000 0002 01 1085719
f1204080: 00000002 00000000 00000000 0002 01 1057186
```

◇ Exemple : informations sur les utilisations de la mémoire par le noyau

```
% cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  1055293440 1040719872 14573568        0 107249664 692142140
Swap: 4294901760 37588992 4257312768
MemTotal:      1030560 kB
MemFree:         14232 kB
MemShared:           0 kB
Buffers:        104736 kB
Cached:         654592 kB
SwapCached:      21328 kB
Active:         547860 kB
Inact_dirty:    197696 kB
Inact_clean:    119996 kB
Inact_target:   173108 kB
HighTotal:      131008 kB
HighFree:         1024 kB
LowTotal:       899552 kB
LowFree:         13208 kB
SwapTotal:     4194240 kB
SwapFree:      4157532 kB
Committed_AS:   281672 kB
```



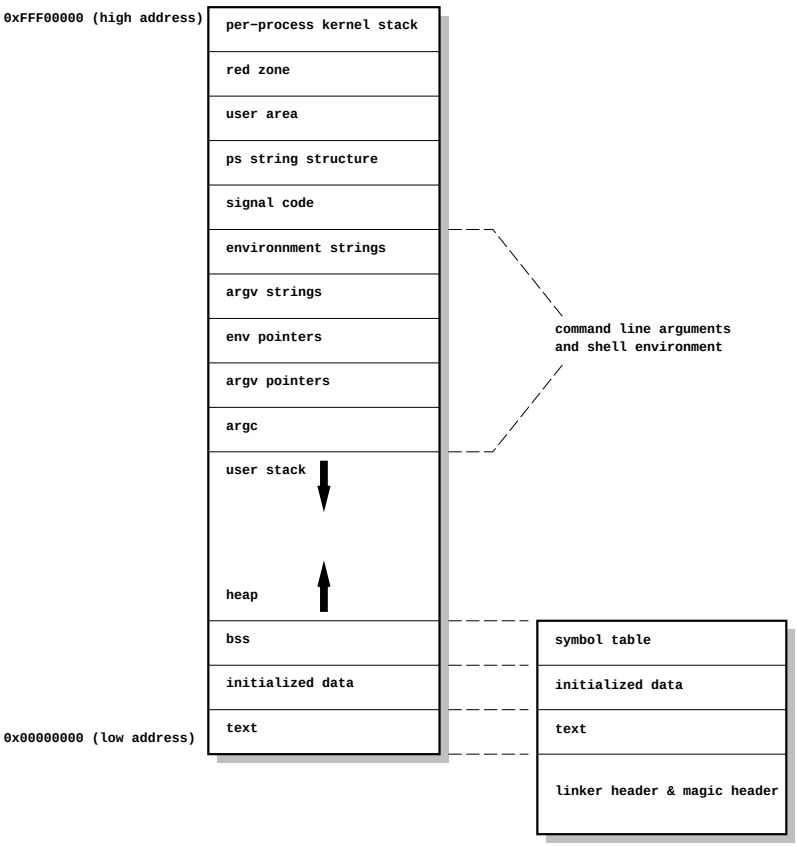


Un processus UNIX est une tâche définie par :

- le code machine à exécuter (*text*)
- une pile d'exécution (*user stack*)
- différentes zones mémoire stockant les structures manipulées par le code machine du programme (*initialized data*, *bss*, *heap*)
- différentes zones mémoire stockant les informations système sur le processus (*argc*, *argv*, *user area*, *red zone*, *kernel stack*)

(cf `<sys/proc.h>`, `<sys/user.h>`)

Cela donne le schéma de la page suivante :

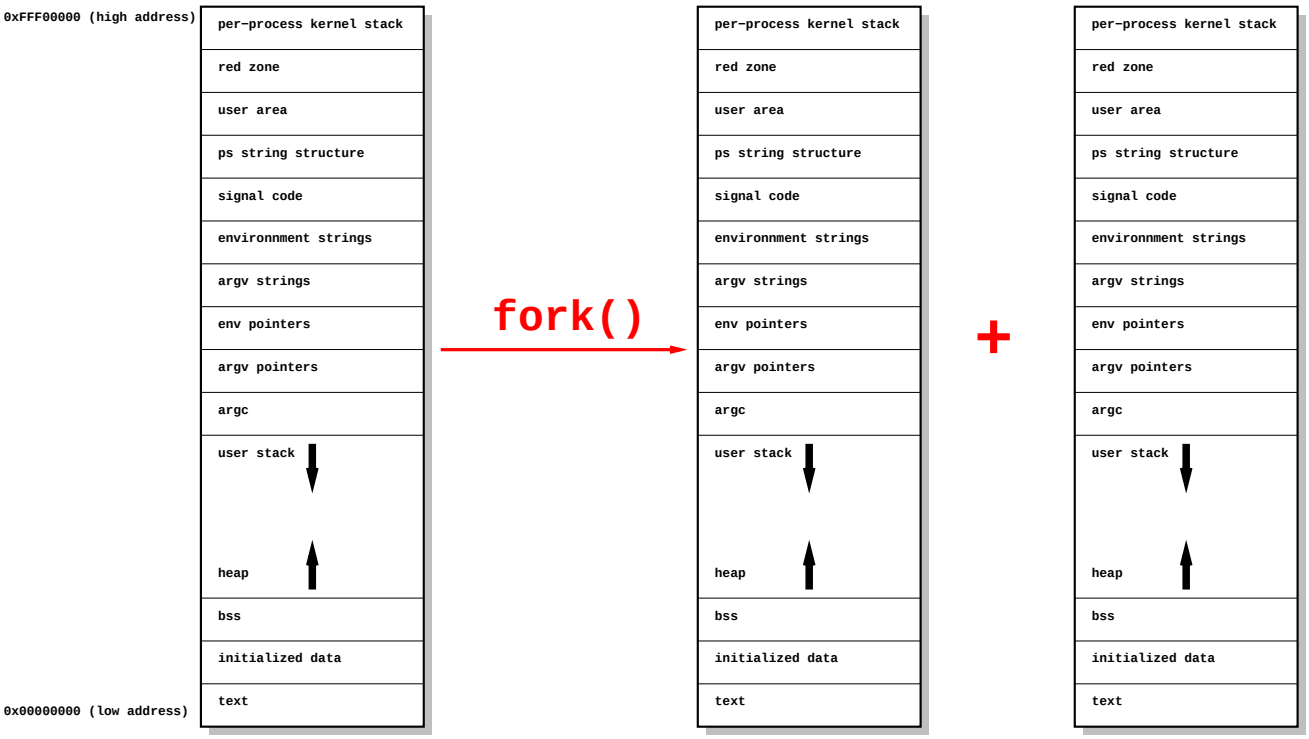







Créer de nouveaux processus passe par la fonction C « `fork()` »

Créer un nouveau processus, c'est dupliquer toutes ces structures mémoire.




Un « `fork()` » est un mécanisme couteux.

## Au niveau de l'exécution du code :

Avant le fork()

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```

Apres le fork()

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```





## Principale utilisation : `fork()` + `exec()` :

Avant le fork()

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```

Apres le fork()

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```

```
...
if ( fork() )
  {
    ... /* parent */
    ... /* code */
  }
else
  {
    exec(...); /* child */
    ...        /* code */
  }
...
```

Apres le exec()

```
main(...)
{
  ... /* autre code */
}
```

## Code d'un exemple de `fork()` + `exec()` :

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>

main(int argc, char *argv[])
  {
    pid_t pid1 ;
    pid_t pid2 ;

    switch( (pid1 = fork()) )
      {
        case -1 :
          fprintf(stderr, "Can't fork()\n");
          exit(1);
          break;
```

*(suite page suivante)*



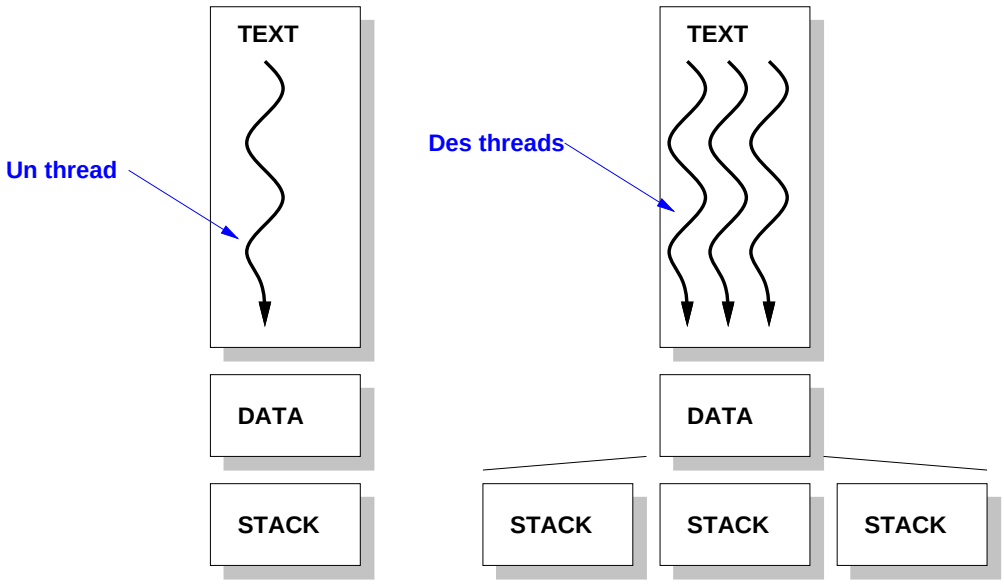


```
        case 0 :
          /* Dans le fils */
          printf("Je suis le fils %d du père %d\n", getpid(), getppid());
          sleep(10) ; /* pour faire attendre le wait() du père */
          execl("/bin/ls", "ls", "-l", "-i", "/etc/motd", NULL);
          break;
        default :
          /* Dans le père */
          pid2 = wait(NULL);
          printf("Je suis le père %d du fils %d\n", getpid(), pid1);
          printf("Mon fils %d est terminé\n", pid2);
          break;
      }
  }
```

## L'exécution donne :

```
Je suis le fils 20285 du père 20284  <-- attente de 10 secondes
  345674 -rw-r--r--   1 root   sys   49 Apr  7  2002 /etc/motd
Je suis le père 20284 du fils 20285
Mon fils 20285 est terminé
```

Un thread ≡ *Lightweigth process* ≡ un programm counter, une stack, un jeu de registres, **tout le reste est partagé avec le processus**




Un processus traditionnel  Un processus multithreadé





Développement commencé sous UNIX dans les années 1980.

Les threads sont un héritage du système **MACH** qui y empruntent la terminologie : **task**, **thread**

Un processus UNIX ≡ une task + un seul thread

Standards Posix : P1003.1b final draft 14 (realtime) et P1003.1c final draft 10 (multithreading)

Avantages des threads :

- création **light** des threads (60 % plus rapide de créer un thread qu'un processus)
- possibilité de créer des applications concurrentes sur des machines mono ou multi processeurs en étant complétement transparentes pour l'utilisateur
- possibilité de faire des applications non bloquantes (le plus souvent en termes d'I/O)
  ⇒ amélioration du temps de réponse
  exemple : serveur web Apache 2.x pour traiter plus de connexions en parallèle ; cf `http://httpd.apache.org`





Inconvénients des threads : **penser en profondeur le code à programmer** :

- attention aux *race conditions* (les threads sont ordonnancés et exécutés au hasard par l'OS)
- attention à la non réentrance du code (variables statiques ou globales a priori bannies)
  ⇒ le code doit être *thread safe*
- utilisation de « `mutex` » ou de *POSIX condition variable* pour synchroniser les threads entre eux, avec risque de deadlocks malgré tout

Sur Solaris, utiliser l'option « -L » de « ps » ou les options d'affichage « lwp » et « nlwp » :

```
% ps -e -o pid,user,lwp,nlwp,comm
  PID     USER    LWP NLWP COMMAND
...
  361     root      1    8 /usr/sbin/nscd
  361     root      2    8 /usr/sbin/nscd
  361     root      3    8 /usr/sbin/nscd
  361     root      4    8 /usr/sbin/nscd
  361     root      5    8 /usr/sbin/nscd
  361     root      6    8 /usr/sbin/nscd
  361     root      7    8 /usr/sbin/nscd
  361     root      8    8 /usr/sbin/nscd
...
```

Ici on voit que le processus UNIX 361 comporte en fait 8 threads.





Sur LINUX, utiliser l'option « -m » :

```
% ./mkthread2 &
[1] 17049
Thread number 16386 / pid 17051
Thread number 32771 / pid 17052
Thread number 49156 / pid 17053
Thread number 65541 / pid 17054

% ps ax | grep mkt
17049 pts/32   S      0:00 ./mkthread2
17062 pts/32   S      0:00 grep mkthread2

% ps axm | grep mkt
17049 pts/32   S      0:00 ./mkthread2
17050 pts/32   S      0:00 ./mkthread2
17051 pts/32   S      0:00 ./mkthread2
17052 pts/32   S      0:00 ./mkthread2
17053 pts/32   S      0:00 ./mkthread2
17054 pts/32   S      0:00 ./mkthread2
```

La librairie pour le link s'appelle traditionnellement par «`-lpthread`» (pthread ≡ *Posix Thread*)

Si votre système n'a pas de thread intégrés, se reporter aux GNU Portable Threads ou `http://www.gnu.org/software/pth/` (download sur `ftp://ftp.lip6.fr/pub/gnu/pth/`)

Il y a des threads non Posix.
Par exemple les threads Solaris, cf «`-lthread`».

Pour un manuel de programmation, se reporter par exemple au manuel SUN intitulé *Multithread Programming Guide* d'URL `ftp://192.18.99.138/802-5938/802-5938.pdf`

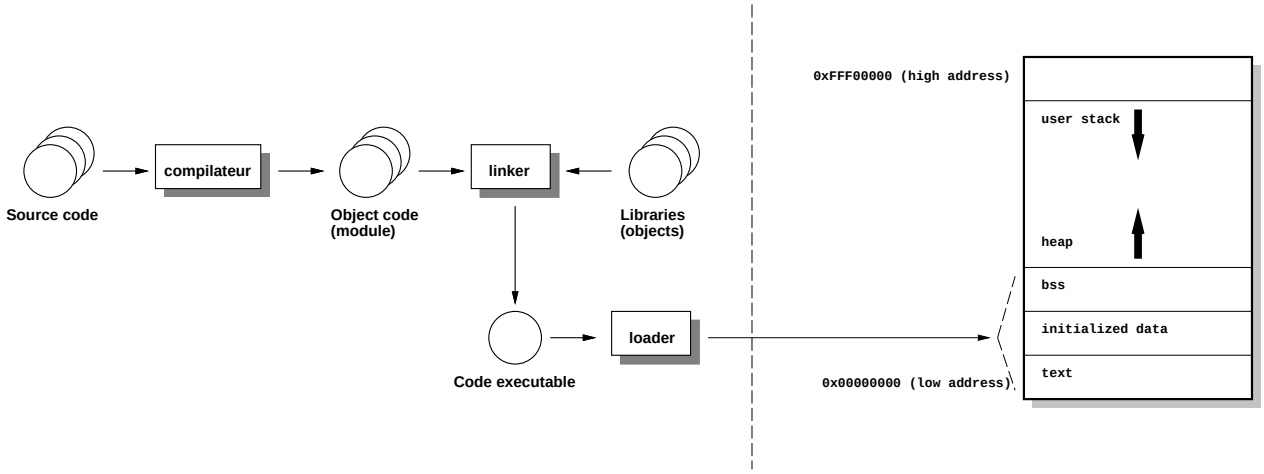




Apparemment la compilation se résume à

```
% gcc programme.c
% ls -lF a.out
-rwxr-xr-x   1 besancon ars    24576 Sep 28 23:25 a.out*
```

En fait, c'est beaucoup plus complexe :



Le même exemple de compilation mais en ne masquant plus rien (exemple sur SunOS) :

```
% cc -v programme.c
/lib/cpp -undef -Dunix -Dsun -Dsparc programme.c >/tmp/cpp.392.0.i
/lib/ccom - </tmp/cpp.392.0.i >/tmp/ccom.392.1.s
rm /tmp/cpp.392.0.i
/bin/as -o programme.o -Q /tmp/ccom.392.1.s
rm /tmp/ccom.392.1.s
/bin/ld -dc -dp -e start -X -o a.out /usr/lib/crt0.o programme.o -
rm programme.o
% ls -lF a.out
-rwxr-xr-x   1 besancon ars    24576 Sep 28 23:26 a.out*
```





On distingue donc plusieurs passes formant la chaîne de compilation :

1. Préprocesseur : commande `cpp`

   ```
   /lib/cpp -undef -Dunix -Dsun -Dsparc programme.c >/tmp/cpp.392
   ```

2. Compilateur C générant le code assembleur : commande `ccom`, `cc1`, ou autre

   ```
   /lib/ccom - </tmp/cpp.392.0.i >/tmp/ccom.392.1.s
   rm /tmp/cpp.392.0.i
   ```

3. Assembleur générant le code machine : commande `as`

   ```
   /bin/as -o programme.o -Q /tmp/ccom.392.1.s
   rm /tmp/ccom.392.1.s
   ```

4. Edition de liens (assemblage du code machine avec les bibliothèques système) : commande `ld`

   ```
   /bin/ld -dc -dp -e start -X -o a.out /usr/lib/crt0.o programme
   rm programme.o
   ```

On peut bien sûr lancer chaque étape manuellement.

Le langage C consiste en :

- des types atomiques (int, long, etc.)
- une syntaxe ; par exemple :
  - `a = b + c ;`
  - `i++ ;`
  - 
```
int ma_fonction(int a)
{
  return( a / 3 );
}
```

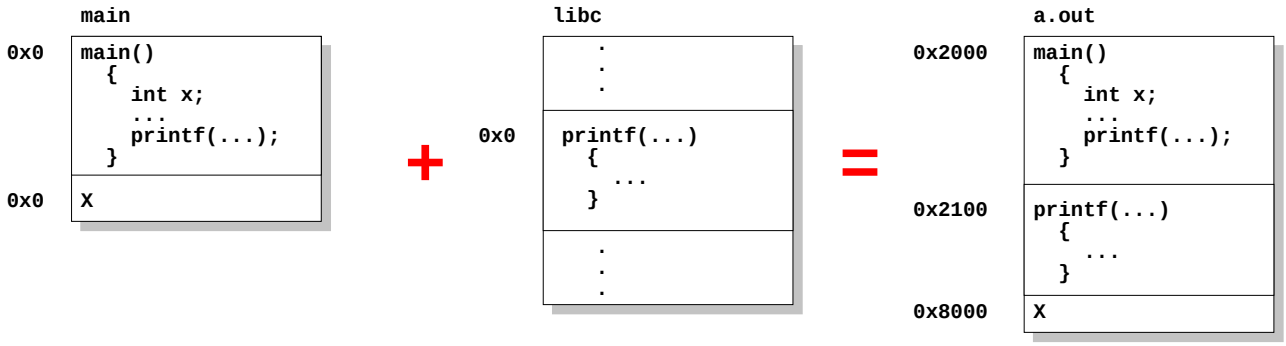
Ne font pas partie du langage C :

- les directives du préprocesseur : « `#include` », « `#ifdef` », etc.
- les fonctions : « `printf()` », « `scanf()` », etc.





L'édition de liens consiste en la complémentation du code utilisateur par le code précompilé des fonctions système utilisées. ⇒ emploi de **bibliothèques**.




Comment précise-t-on une bibliothèque à la compilation ?

```
% gcc programme.c -lm
```

C'est toujours de la forme « -**lbibliothèque** ».

Où chercher les bibliothèques de fonctions système ?

Traditionnellement dans : `/lib`, `/usr/lib`, `/usr/local/lib`

C'est précisable au niveau de la ligne de compilation :

```
% gcc programme.c -Lrépertoire -lutilitaires
```





Quels fichiers stockent-ils les bibliothèques ?

Par exemple :

```
% gcc programme.c -lrpcsvc -lm
```

utilise les bibliothèques
`/usr/lib/librpcsvc.a`
`/usr/lib/libm.a`

**Attention : cela dépend en fait du type d'édition de liens** :

- édition de liens statique
- édition de liens dynamique

Edition de liens statique : Le code de toutes les fonctions système utilisées est chargé dans l'exécutable.

Inconvénient : l'exécutable est figé, monolithique, auto-suffisant, gros. En cas de problème avec une fonction système (incorrectement codée par exemple), si une nouvelle bibliothèque est disponible auprès du constructeur, il faudra recompiler.

```
% gcc -static programme.c -lrpcsvc
% ls -l a.out
-rwxr-xr-x  1 besancon  ars  32768 Oct  8 21:42 a.out
```

⇒ emploi de « `/usr/lib/librpcsvc.a` »





Comment connaître le type d'édition de liens d'un exécutable a posteriori ?

Utiliser la commande **file** :

```
% gcc -static programme.c
% file a.out
a.out: FreeBSD/i386 compact demand paged executable not stripped
```

Par défaut les compilateurs travaillent en mode dynamic.

Une bibliothèque statique est une collection de modules objet « .o » avec un index.

La commande UNIX pour assembler les modules objet et créer la table d'index est : « **ar** »

```
% ar cq libutilitaires.a module1.o module2.o ...
```

ou plus généralement

```
% ar [option] bibliotheque module1.o module2.o ...
```





Selon les systèmes UNIX, il faut utiliser la commande **ranlib** pour mettre à jour la table des matières (problème de date) :

```
# touch /lib/libm.a
% gcc -Bstatic programme.c -lm
ld: /lib/libm.a: warning: table of contents for archive is out
of date; rerun ranlib(1)
```

Parfois, l'édition de liens échoue uniquement par absence d'un ranlib non fait.

La solution :

```
# ranlib /lib/libm.a
```

Les binaires de « /sbin » sont compilés en static en théorie :

```
% file /sbin/*
/sbin/dump:          FreeBSD/i386 compact demand paged executable
/sbin/fastboot:      FreeBSD/i386 compact demand paged executable
/sbin/fasthalt:      FreeBSD/i386 compact demand paged executable
/sbin/fdisk:         FreeBSD/i386 compact demand paged executable
/sbin/fsck:          FreeBSD/i386 compact demand paged executable
...
/sbin/ifconfig:      FreeBSD/i386 compact demand paged executable
/sbin/init:          FreeBSD/i386 compact demand paged executable
...
/sbin/mknod:         FreeBSD/i386 compact demand paged executable
...
/sbin/mount:         FreeBSD/i386 compact demand paged executable
...
```

Raison : on fait appel à ces binaires à un moment dans la vie de la station UNIX où le mécanisme des bibliothèques partagées n'est pas encore activé.





Edition de liens dynamique : On ne chargera **qu'à l'exécution** le code des fonctions système utilisées.
Avantage : l'exécutable n'a pas besoin d'être remis à jour en cas de changement de bibliothèques système.
Inconvénients :

1. l'exécutable est **dépendant** vis-à-vis des bibliothèques système
2. pseudo ralentissement au moment du lancement de l'exécutable ?

```
% gcc -dynamic programme.c -lrpcsvc
% ls -lF a.out
-rwxr-xr-x  1 besancon  ars  24576 Oct  8 21:42 a.out*
```

⇒ emploi de
/usr/lib/libc.so.2.0
/usr/lib/librpcsvc.so.3.1

Comment connaître le type d'édition de liens d'un exécutable a posteriori ?
Utiliser la commande **file** :

```
% gcc programme.c
% file a.out
a.out: FreeBSD/i386 compact demand paged dynamically linked execut
```

Par défaut les compilateurs travaillent en mode dynamic.





Comment connaître la liste des bibliothèques partagées dont a besoin un exécutable ?
⇒ Utiliser la commande **ldd**
(en anglais *list dynamic dependencies*)

```
% gcc programme.c
% ldd a.out
a.out:
        -lc.3 => /usr/lib/libc.so.3.1 (0x2001e000)

% ldd /usr/X11R6/bin/xterm
/usr/X11R6/bin/xterm:
        -lXaw.6 => /usr/X11R6/lib/libXaw.so.6.1 (0x2003e000)
        -lXmu.6 => /usr/X11R6/lib/libXmu.so.6.0 (0x20071000)
        -lXt.6 => /usr/X11R6/lib/libXt.so.6.0 (0x20080000)
        -lX11.6 => /usr/X11R6/lib/libX11.so.6.1 (0x200bc000)
        -lSM.6 => /usr/X11R6/lib/libSM.so.6.0 (0x2014c000)
        -lICE.6 => /usr/X11R6/lib/libICE.so.6.3 (0x20154000)
        -lXext.6 => /usr/X11R6/lib/libXext.so.6.3 (0x20165000)
        -ltermcap.2 => /usr/lib/libtermcap.so.2.1 (0x2016e000)
        -lc.3 => /usr/lib/libc.so.3.1 (0x20172000)
```

Bien sûr :

```
% ldd /sbin/fsck
ldd: /sbin/fsck: not a dynamic executable
```





Une bibliothèque dynamique a un numéro de version :

```
% ldd /usr/X11R6/bin/xterm
/usr/X11R6/bin/xterm:
...
        -lSM.6 => /usr/X11R6/lib/libSM.so.6.0 (0x2014c000)
        -lICE.6 => /usr/X11R6/lib/libICE.so.6.3 (0x20154000)
        -lXext.6 => /usr/X11R6/lib/libXext.so.6.3 (0x20165000)
        -ltermcap.2 => /usr/lib/libtermcap.so.2.1 (0x2016e000)
        -lc.3 => /usr/lib/libc.so.3.1 (0x20172000)
```

Exemple de message en cas de mismatch de versions d'une bibliothèque partagée :

```
% ./a.out
/usr/libexec/ld.so: warning: /usr/lib/libc.so.3.0: minor
version 0 older than expected 1, using it anyway
```

## Commande Windows proche chez http://www.sysinternals.com : « listdlls.exe »

```
ListDLLs V2.23 - DLL lister for Win9x/NT
Copyright (C) 1997-2000 Mark Russinovich
http://www.sysinternals.com

------------------------------------------------------------------------------
System pid: 4
Command line: <no command line>
------------------------------------------------------------------------------
smss.exe pid: 1304
Command line: \SystemRoot\System32\smss.exe

  Base        Size      Version         Path
  0x48580000  0xe000                   \SystemRoot\System32\smss.exe
  0x77f50000  0xa7000   5.01.2600.1217  C:\WINDOWS\System32\ntdll.dll
...
```





```
...
------------------------------------------------------------------------------
cmd.exe pid: 780
Command line: "C:\WINDOWS\System32\cmd.exe"

  Base        Size      Version         Path
  0x4ad00000  0x5e000   5.01.2600.0000  C:\WINDOWS\System32\cmd.exe
  0x77f50000  0xa7000   5.01.2600.1217  C:\WINDOWS\System32\ntdll.dll
  0x77e60000  0xe6000   5.01.2600.1106  C:\WINDOWS\system32\kernel32.dll
  0x77c10000  0x53000   7.00.2600.1106  C:\WINDOWS\system32\msvcrt.dll
  0x77d40000  0x8c000   5.01.2600.1255  C:\WINDOWS\system32\USER32.dll
  0x7e090000  0x41000   5.01.2600.1346  C:\WINDOWS\system32\GDI32.dll
  0x77dd0000  0x8d000   5.01.2600.1106  C:\WINDOWS\system32\ADVAPI32.dll
  0x78000000  0x87000   5.01.2600.1361  C:\WINDOWS\system32\RPCRT4.dll
  0x75f40000  0x1f000   5.01.2600.1106  C:\WINDOWS\system32\Apphelp.dll
------------------------------------------------------------------------------
...
```

Exemple de message en cas d'absence d'une bibliothèque partagée :

```
% ./a.out
ld.so failed: Can't find shared library "libpico.so.1.3"
% ldd a.out
a.out:
/usr/libexec/ld.so: warning: /usr/lib/libc.so.3.0: minor
version 0 older than expected 1, using it anyway
        -lpico.1 => not found (0x0)
        -lc.3 => /usr/lib/libc.so.3.0 (0x1001d000)
```

⇒ Comment ajouter des bibliothèques partagées au système ?





Méthode d'ajout de bibliothèques partagées :

1. Copier la bibliothèque partagée dans un répertoire système prévu pour cela
2. Faire prendre connaissance au système (en fait au programme « **ld.so** ») de la présence d'une nouvelle bibliothèque partagée.
   ⇒ Utiliser la commande « **ldconfig** »

La variable d'environnement « LD_LIBRARY_PATH » permet d'indiquer une liste de répertoires **non standard** où chercher des librairies dynamiques.

En temps normal, la variable d'environnement est vide.

Quand elle est utilisée, c'est via un shell script pour limiter l'action de la variable à la durée de vie du script.





Exemple : un programme utilise des librairies dynamiques non standard :

```
% ldd /usr/local/iv/bin/idraw
        libUniIdraw.so =>       (file not found)
        libUnidraw.so =>        (file not found)
        libIV.so =>             (file not found)
        libXext.so.6.4 =>       /usr/X11R6/lib/libXext.so.6.4
        libX11.so.6.1 =>        /usr/X11R6/lib/libX11.so.6.1
        libSM.so.6.0 =>         /usr/X11R6/lib/libSM.so.6.0
        libICE.so.6.3 =>        /usr/X11R6/lib/libICE.so.6.3
        libsocket.so.1 =>       /lib/libsocket.so.1
        libm.so.1 =>            /lib/libm.so.1
        libc.so.1 =>            /lib/libc.so.1
```

Alors pour lancer le programme on fait le script « /usr/local/bin/idraw » suivant :

```
#!/bin/sh
LD_LIBRARY_PATH=/usr/local/iv/lib
export LD_LIBRARY_PATH
/usr/local/iv/bin/idraw "$@"
```

En cas d'effacement de bibliothèques partagées...

1. AÏE
2. Essayer d'utiliser les binaires static pour réparer
3. Redémarrer sur CDROM
4. Avoir pensé à exporter par NFS les répertoires de bibliothèques partagées de façon à recopier les bibliothèques manquantes par NFS.
5. Avoir compilé en static les outils nécessaires à réparer. C'est possible en utilisant des sources d'UNIX du domaine public.

Dans tous les cas, c'est difficile et pas gagné d'avance...





Dynamic linking : le système UNIX complète lors de l'exécution d'une application le code binaire de l'application par le code binaire de librairies.

Dynamic loading : l'application complète elle-même son code binaire lors de son exécution par le code binaire de modules.

Avantages du dynamic loading :

- extensions possibles d'une application
- possibilité de publier une API pour écrire des modules sans pour autant dévoiler le code source de l'application ⇒ intéressant pour des ISV

## ◇ Application du dynamic loading : Modules, Plugins

- Modules Apache

  *Modules de filtrage, de réécriture de documents HTML, etc.*
  *Cf* `http://modules.apache.org/`

- Modules PAM

  *Modules d'ajout de procédures d'authentification dans des programmes système (programmes du genre «* `su` *», «* `login` *», «* `telnetd` *», etc.)*
  *Cf* `http://www.kernel.org/pub/linux/libs/pam/`
  *Cf* `http://www.solaris.com/solaris/pam/`

- Modules POSTGRESQL

  *Modules définissant de nouvelles fonctions, de nouveaux opérateurs, de nouveaux types SQL, etc.*
  *Cf* `http://gborg.postgresql.org/`

- etc.





## ◇ Exemple : application faisant du dynamic loading

```
#include <stdio.h>
#include <dlfcn.h>
#include <link.h>
#include <limits.h>
#include <unistd.h>

main()
  {
    void *module_handle;
    int module_mode = RTLD_NOW | RTLD_GLOBAL;
    void (*sym_handle)();
    int dlinfo ;

    printf("Nous sommes dans le fichier %s\n", __FILE__);
    printf("Nous sommes le processus %d\n", getpid());
    printf("sleep(20)\n");
    sleep(20);
    printf("Chargement du module...\n");
    module_handle = dlopen("./module.so.1", module_mode);
```

Suite à la page suivante. . .

```
    if ( module_handle )
      {
        printf("Module chargé en = 0x%x\n", module_handle);

        /* Recuperation de l'adresse du symbole */
        sym_handle = dlsym(module_handle, "fonction");
        printf("Symbole chargé en = 0x%x\n", sym_handle);

        /* Appel du symbole */
        (void)(*sym_handle)();

        /* On en a fini avec le module dynamique */
        printf("Déchargement du module...\n");
        dlclose(module_handle);
        printf("sleep(20)\n");
        sleep(20);
        exit(0);
      }
    else
      {
        /* Module non charge */
        printf("Module non chargé.\n");
        exit(1);
      }
  }
```





## ◇ Exemple : module pour l'application

```
#include<stdio.h>

void fonction()
  {
    printf("Nous sommes dans le fichier %s dans %s()+%d\n",
           __FILE__, __FUNCTION__, __LINE__);
    printf("Nous sommes toujours le processus %d\n", getpid());
    printf("sleep(20)\n");
    sleep(20);
    printf("Fin d'exécution de %s()\n", __FUNCTION__);
  }
```

## ◇ Exemple : lancement de l'application faisant du dynamic loading

```
% ./programme.exe
Nous sommes dans le fichier programme.c
Nous sommes le processus 3021
sleep(20)
Chargement du module...
Module chargé en = 0xff3a1dcc
Symbole chargé en = 0xff2502d0
Nous sommes dans le fichier module.c dans fonction()+6
Nous sommes toujours le processus 3021
sleep(20)
Fin d'exécution de fonction()
Déchargement du module...
sleep(20)
```





## Plus exactement :

```
% ./programme.exe
Nous sommes dans le fichier programme.c
Nous sommes le processus 3021
sleep(20)
Chargement du module...
Module chargé en = 0xff3a1dcc
Symbole chargé en = 0xff2502d0
Nous sommes dans le fichier module.c dans fonction()+6
Nous sommes toujours le processus 3021
sleep(20)
Fin d'exécution de fonction()
Déchargement du module...
sleep(20)
```

```
% pldd 3021
3021:   ./programme.exe
/usr/lib/libdl.so.1
/usr/lib/libc.so.1
/usr/platform/sun4u/lib/libc_psr.so.
```

```
% pldd 3021
3021:   ./programme.exe
/usr/lib/libdl.so.1
/usr/lib/libc.so.1
/usr/platform/sun4u/lib/libc_psr.so.
/var/tmp/dl/module.so.1
/usr/lib/libld.so.2
/usr/lib/libelf.so.1
```

```
% pldd 3021
3021:   ./programme.exe
/usr/lib/libdl.so.1
/usr/lib/libc.so.1
/usr/platform/sun4u/lib/libc_psr.so.1
/usr/lib/libld.so.2
/usr/lib/libelf.so.1
```

Le mangling est le passage du nom de fonction exprimé dans le langage de programmation au symbole interne pour la compilation.
Par exemple suivant le langage et le système, la fonction « `fonction()` » sera manglée de la façon suivante :

| Langage | Système UNIX | Fonction | Symbole manglé |
|---|---|---|---|
| Fortran | HP-UX | `fonction` | `fonction` |
| Fortran | SunOS | `fonction` | `_fonction_` |
| Fortran | AIX | `fonction` | `.fonction` |
| C | HP-UX | `fonction` | `fonction` |
| C | FreeBSD | `fonction` | `_fonction` |
| C++ | FreeBSD | `fonction` | `_fonction__Fv` |





Le problème de l'administrateur (lors du portage d'un soft par exemple) :

```
% gcc programme.c
/var/tmp/ccl134891.o: Undefined symbol `_pintf' referenced from
text segment
```

Il faut donc :

- savoir identifier la fonction en cause dans le source C.
  La fonction en cause existe-t-elle sur cet UNIX ?
- Si la fonction est implémentée, dans quelle bibliothèque la trouve-t-on ?

La commande **nm** répond au point 2.

```
% nm /usr/lib/libm.a
i387_s_tan.o:
00000038 t ___arch_select_tan
00000090 D ___arch_tan
         U ___generic_tan
         U ___get_hw_float
0000005c T ___i387_tan
00000054 T _tan
...
```





Pour diverses raisons, il y a au sein d'un exécutable une table interne d'informations qui ne servent pas la plupart du temps : **la table des symboles**
Inconvénient : l'exécutable est plus gros que nécessaire à cause de la table des symboles.
Pour supprimer cette table, utiliser la commande **strip**.

```
% gcc programme.c -lm
% file a.out
a.out:     sparc demand paged dynamically linked executable not stripped
% ls -l a.out
-rwxr-xr-x   1 besancon ars    32768 Oct  8 22:21 a.out
% strip a.out
% file a.out
a.out:     sparc demand paged dynamically linked executable
% ls -l a.out
-rwxr-xr-x   1 besancon ars    24576 Oct  8 22:21 a.out
```

Inutile de lancer « strip » plusieurs fois sur le même binaire :

```
% file a.out
a.out:     sparc demand paged dynamically linked executable
not stripped
% strip a.out
% strip a.out
strip: a.out already stripped
```





Soit un programme C :

```
#include <stdio.h>
main()
{
  int i = 3;
  printf ("Hello world %d !\n", i);
}
```

Alors :

```
% gcc programme.c
% ./a.out
Hello world 3 !
% strings a.out
Hello world %d !
```

Intérêts de la commande **strings** :

- elle permet de trouver certaines informations dans un exécutable sans avoir recours aux sources de l'exécutable
- elle s'applique à tout fichier

La commande sera particulièrement utilisée sur des programmes récupérés sur Internet, sur des applications suspectes. Exemple sur un programme suspect, en fait un mouchard de mots de passe déposé par un pirate informatique :

```
# /usr/ucb/gcore 26704
/usr/ucb/gcore: core.26704 dumped
# strings core.26704
...
log.log
...
```

⇒ révélation du nom du fichier de log ; plus qu'à faire un « `find` »





Quand une application se termine anormalement, elle génére une image de son espace mémoire sous la forme d'un fichier « **core** » :

```
% cat buggy.c
#include <string.h>
main()
{
  char *p = NULL;
  strcpy(p, "boum !");
}

% gcc buggy.c
% ./a.out
Segmentation fault (core dumped)
```

A noter que selon les systèmes UNIX, le fichier « core » prend des noms différents :

Hypothèse : le programme en question s'appelle « `programme` »

| UNIX | Nom de l'image |
|---|---|
| SOLARIS | `core` |
| FreeBSD | `programme.core` |
| LINUX | `core.PID` |





De par sa nature (image mémoire), le fichier « core » est plein de trous :

```
% df .
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sd2g            1791382 1519809    92435     94%   /users/adm

% gcc big-buggy.c
% ./a.out 10
Allocation de 10 Mo...
So far so good :-)
Bye bye...
Segmentation fault (core dumped)

% ls -ls core
10336 -rw-r--r--   1 besancon ars 18911664 Oct  8 23:49 core
% df .
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sd2g            1791382 1530081    82163     95%   /users/adm
```

On constate que $92435 - 18468 \neq 82163$ ! $\Rightarrow$ le fichier contient des trous.
La taille du fichier ne correspond pas à ce qui est utilisé en réalité.
Raison : les trous de l'espace mémoire n'allouent pas de bloc disque.
On parle de **fichier sparse**.

## Comment créer soi-même un fichier sparse ?

```
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>

main()
{
  int    fd;
  off_t  offset;
  char   *buf = "bof";

  /*
   * Attention ce programme est quick & dirty car on ne verifie
   * aucune valeur de retour des fonctions.
   */
  fd = open("/tmp/sparse", O_WRONLY | O_CREAT, 0644);
  offset = 16 * 1024 * 1024 ;
  lseek(fd, offset, SEEK_SET);
  write(fd, buf, strlen(buf));
  close(fd);
}
```





Après avoir fait tourner le programme précédent, on voit que les blocs intermédiaires ne sont pas alloués :

```
% ls -ls /tmp/sparse
16 -rw-r--r--  1 besancon   16777219 Sep 21 19:17 /tmp/sparse
```

Le caractère « sparse » n'est pas conservé par recopie :

```
% cp /tmp/sparse /tmp/sparse2
% ls -ls /tmp/sparse /tmp/sparse2
   16 -rw-r--r--  1 besancon   16777219 Sep 21 19:17 /tmp/sparse
16400 -rw-r--r--  1 besancon   16777219 Sep 21 19:17 /tmp/sparse2
```

## ◇ Empêcher la création de fichiers core

Plusieurs solutions :

- créer un répertoire « core » dans le rèpertoire où un core peut être créé
- commande « ulimit » de BASH (à mettre dans « $HOME/.profile ») :

```
% echo $SHELL
/bin/bash
% ulimit -c 128
% ulimit -a
core file size (blocks)     128
data seg size (kbytes)      unlimited
file size (blocks)          unlimited
open files                  256
pipe size (512 bytes)       10
stack size (kbytes)         8192
cpu time (seconds)          unlimited
max user processes          9845
virtual memory (kbytes)     unlimited
```





- commande « limit » (selon le shell) :

```
% echo $SHELL
/bin/csh
% limit coredumpsize 128
% limit
cputime         unlimited
filesize        unlimited
datasize        unlimited
stacksize       8192 kbytes
coredumpsize    128 kbytes
descriptors     256
memorysize      unlimited
```

Pour debugger une application qui ne marche pas bien, il peut être intéressant de lancer un suivi du runtime : analyse des appels système (fonctions de la section 2 du manuel) faits par le programme.

En pratique :

- programme « `trace` »
- programme « `truss` » sur SOLARIS
- programme « `ktrace` » et « `kdump` » sur FreeBSD
- autres ?

Disponibilité de la commande sur la quasi totalité des systèmes soit de base soit via `http://www.liacs.nl/~wichert/strace/`





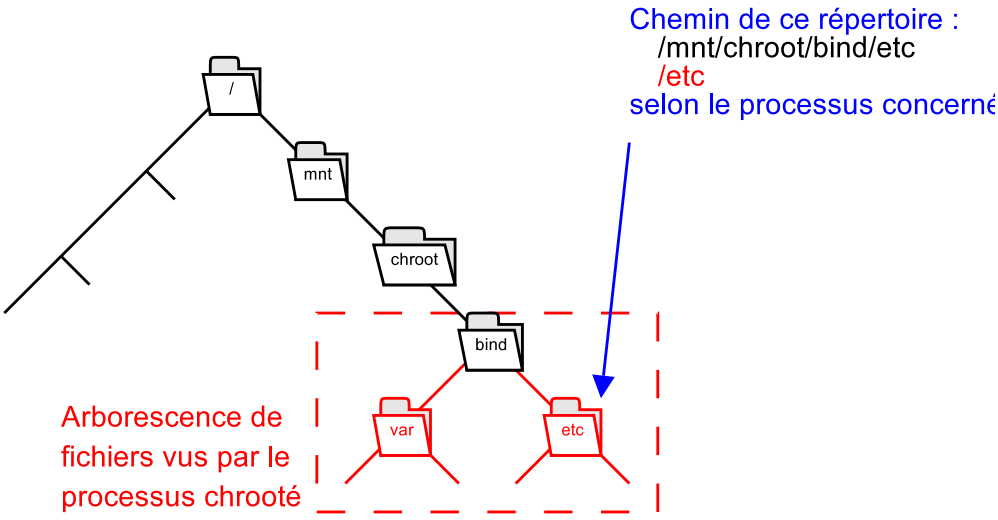
## Exemple 1 : Connaître le type d'actions de base réalisées par un programme

```
close(3)                                = 0
open("/usr/lib/libc.so.1.9.1", O_RDONLY) = 3
read(3, "\201\3\1\v\0\7\240\0\0\0@\0\0\000"..., 32) = 32
```

## Exemple 2 : Pouvoir comprendre les dysfonctionnements de programmes

```
25487 xmgr-3.01pl7.LIN RET   execve 0
25487 xmgr-3.01pl7.LIN CALL  getitimer(0x1e6d7e)
25487 xmgr-3.01pl7.LIN NAMI  "/compat/linux/lib/ld.so"
25487 xmgr-3.01pl7.LIN NAMI  "/lib/ld.so"
25487 xmgr-3.01pl7.LIN RET   getitimer JUSTRETURN
25487 xmgr-3.01pl7.LIN CALL  write(0x2,0xefbfdd9c,0x14)
25487 xmgr-3.01pl7.LIN GIO   fd 2 wrote 20 bytes "./xmgr-3.01pl7.LINUX"
25487 xmgr-3.01pl7.LIN CALL  write(0x2,0xefbfdd20,0x2a)
25487 xmgr-3.01pl7.LIN GIO   fd 2 wrote 42 bytes ": can't load dynamic linker
'/lib/ld.so' \0"
```

« Chrooter » un processus consiste à faire tourner un processus avec une origine de l'arborescence des fichiers différente de « / ».




« chroot » est un appel système : « chroot(2) »





Pour le programme chrooté, les fichiers visibles se limitent à ceux contenus dans la « cage » du chrootage.

Principaux programmes chrootés : des programmes réseau exposés à Internet vers lesquels viennent des requêtes hostiles.

Dans le cadre de « chroot », on distingue deux catégories de processus qui rendent les choses plus ou moins complexes à configurer :

- processus nés hors de la cage puis vivant et mourant dans la cage
- processus nés, vivant et mourant dans la cage.

◇ Processus nés hors de la cage puis vivant et mourant dans la cage

Par exemple le logiciel BIND (cf `http://www.isc.org/products/BIND/`)
Le principe général de la programmation en pseudo langage de ce type de processus :

1. vérification que l'on tourne sous l'identité de root
2. faire un « `chroot()` » dans la cage
3. lecture des fichiers de configuration dans la cage
4. mise en écoute sur certains ports réseau $< 1024$
5. abandon des privilèges système via le passage sous l'identité d'un utilisateur dédié (pour BIND, il s'agit de l'utilisateur « `bind` » du groupe « `bind` »)

Le binaire lancé se limite à accèder à des fichiers de données (en lecture ou en écriture) dans la cage lors de son fonctionnement en mode régime de croisière (c'est-à-dire que l'on passe sous silence la phase d'initialisation de l'application peut-être avec des fichiers hors de la cage mais cette étape est très limitée dans le temps).





◇ Processus nés, vivant et mourant dans la cage

Par exemple un serveur FTPD qui lancerait un binaire « `ls` » pour répondre à la commande FTP « `dir` ».
Ce programme « `ls` » nait, vit et meurt dans la cage.
**La difficulté : les librairies dynamiques**
Rappel :

```
% ldd /bin/ls
        libc.so.1 =>     /usr/lib/libc.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
        /usr/platform/SUNW,Sun-Blade-100/lib/libc_psr.so.1
```

Implication :

- **Toutes les librairies dynamiques utilisées par des programmes naissant dans une cage doivent exister au sein de la cage.**
- **Tous les devices utilisés par des programmes naissant dans une cage doivent exister au sein de la cage.**

◇ Aspects de sécurité

**CHROOT n'a pas été conçu pour de la sécurité !**
En théorie, un processus chrooté ayant les privilèges système peut faire :

```
# mount /dev/hda1 /mnt
```

où `/mnt` est un répertoire dans la cage
où `/dev/hda1` désigne la partition du vrai « / » de la machine.

Le problème de CHROOT : il y a partage de la définition des devices entre tous les processus, chrootés ou non chrootés.
⇒ CHROOT n'a pas été conçu pour de la sécurité. Le renforcement de la sécurité via CHROOT est un pur effet de bord de son fonctionnement.

Autres solutions apportant de la sécurité :

- JAIL sur FREEBSD
- ZONE sur SOLARIS 10





◇ Processus chrootés contre processus non chrootés

Au niveau de la commande « `ps` », rien ne distingue un processus chrooté d'un processus non chrooté.

Un processus chrooté peut obtenir la liste de tous les processus de la machine. Donc problème en cas de processus chrooté compromis tournant au nom de root car il peut connaitre les processus hors cages.

Un processus chrooté peut obtenir la liste de tous les processus de la machine et leur envoyer des signaux. Donc problème en cas de processus chrooté compromis tournant au nom de root car il peut tuer des processus hors cage.

Chapitre 14

# Horloges UNIX





L'horloge sert à dater les fichiers, lancer des applications à des moments précis voulus (cf cron), dater des événements, etc.

Problèmes classiques : mise à l'heure d'une horloge, consultation d'une horloge, an 2000, etc.




Pour UNIX, le début du temps est le 1 janvier 1970.

La fin du temps : le mardi 19 janvier 2038 à 4 h 14 min 7s (heure GMT) (c'est-à-dire 2147483647 secondes ≡ `0x7fffffff`)

Exemple de problème :

```
% date
Thu Jan 29 11:18:49 CET 2004

% tar xvf archive.tar
lspci.log
tar: lspci.log: time stamp Jan 29 11:59 2004 is 2434 s in the future
lspci-n.log
tar: lspci-n.log: time stamp Jan 29 12:04 2004 is 2724 s in the future
lspci-t.log
tar: lspci-t.log: time stamp Jan 29 12:03 2004 is 2697 s in the future
lspci-v.log
tar: lspci-v.log: time stamp Jan 29 12:04 2004 is 2734 s in the future
lspci-vv.log
tar: lspci-vv.log: time stamp Jan 29 12:03 2004 is 2689 s in the future
proc-pci.log
tar: proc-pci.log: time stamp Jan 29 12:05 2004 is 2816 s in the future
```





Syntaxe : `date [+format]`

```
% date
Thu Oct 22 22:00:51 CEST 1998
```

« `date +format` » utilise un format à la « `printf` » avec des « `%` ».
Cf la page de manuel de la fonction « `strftime` ».
Les bigrammes les plus utiles :

| | |
|---|---|
| `%H` | heure (00..23) |
| `%M` | minute (00..59) |
| `%S` | seconde (00..59) |
| `%T` | temps (hh :mm :ss) |
| `%m` | mois, (01..12) |
| `%d` | jour, (01..31) |
| `%y` | derniers chiffres de l'année (00..99) |
| `%Y` | année (1970...) |

Exemples d'utilisation :

- Avoir la date sous la forme YYYY/MM/DD :

```
% date +%Y/%m/%d
1998/02/22
```

- Avoir le jour de la semaine :

```
% date +%w
4
```

- Avoir le numéro de la semaine :

```
% date +%W
42
```

Pour des informations mathématiques sur les dates (années bissextiles, calcul du jour de la semaine etc., on pourra se reporter à :
ftp://ftp.lip6.fr/pub/doc/faqs/sci-math-faq/dayWeek.gz





Il existe une commande plus sophistiquée : le GNU date d'URL
ftp://ftp.lip6.fr/pub/gnu/sh-utils/sh-utils-2.0.tar.gz
On peut grâce à cette commande calculer des choses impossibles à calculer autrement :

```
% date
Wed Aug  5 11:09:06 MET DST 1998

% date '+%Y/%m/%d'
1998/08/05
% date '+%Y/%m/%d' --date "10 days"
1998/08/15
% date '+%Y/%m/%d' --date "-10 days"
1998/07/26
```

(suite)

```
% date
Wed Aug  5 11:09:06 MET DST 1998

% date '+%Y/%m/%d' --date "1 years 2 months 3 days"
1999/10/08
% date '+%Y/%m/%d' --date "-1 years -2 months -3 days"
1997/06/02
% date '+%Y/%m/%d' --date "1 fortnight"`
1998/08/19
```





La commande date permet de régler l'horloge de la machine.
Syntaxes :

```
date [options] [mmdd]HHMM
date [options] mmddHHMM[cc]yy[.SS]
```

Par exemple

```
# date
Fri Sep 11 11:40:33 CEST 1998

# date 09110938
Fri Sep 11 09:38:00 CEST 1998

# date
Fri Sep 11 09:38:04 CEST 1998
```

Attention aux commandes date non compatibles an 2000.

(en anglais *Daylight Savings Time*)

Le réglage de l'heure d'été, heure d'hiver se fait au niveau des fichiers suivants :

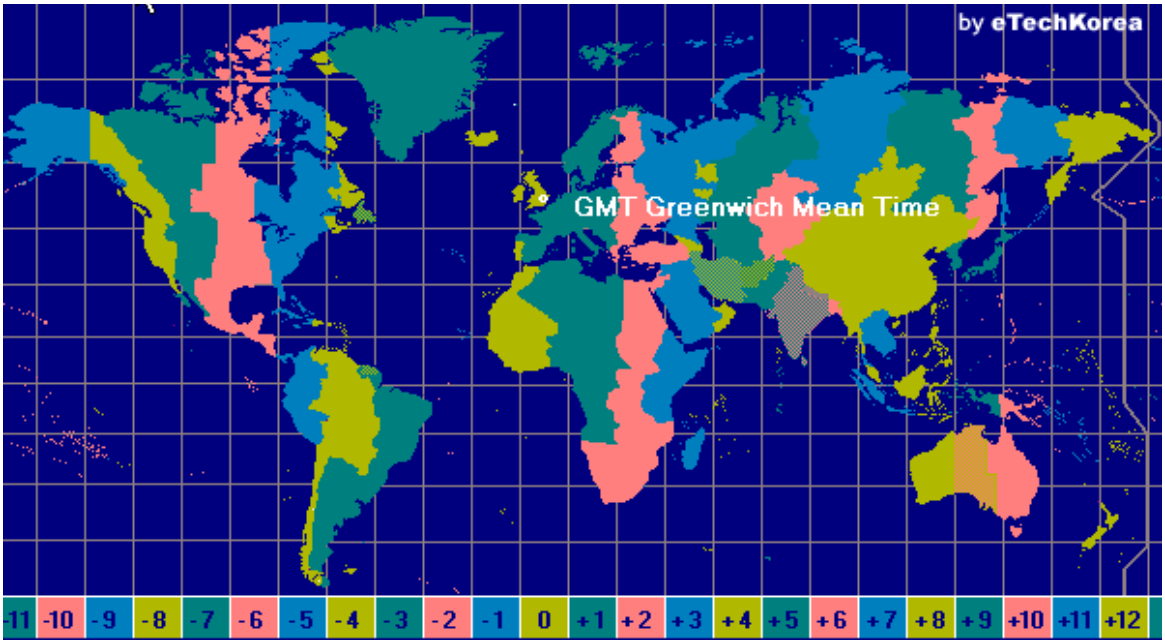
| | |
|---|---|
| AIX | `/etc/environment` |
| DEC | `/etc/zoneinfo/localtime` |
| FreeBSD | `/etc/localtime` |
| HP-UX 10 | `/etc/TIMEZONE` |
| IRIX | `/etc/TIMEZONE` |
| Linux | `/usr/lib/zoneinfo/localtime` |
| SunOS | `/usr/share/lib/zoneinfo/localtime` |
| Solaris | `/etc/TIMEZONE` |





La variable d'environnement **TZ** permet de changer de fuseau horaire lors de l'affichage de `date`.

Pour la France on prend **MET-1DST** (Middle European Time Daylight Saving Time)



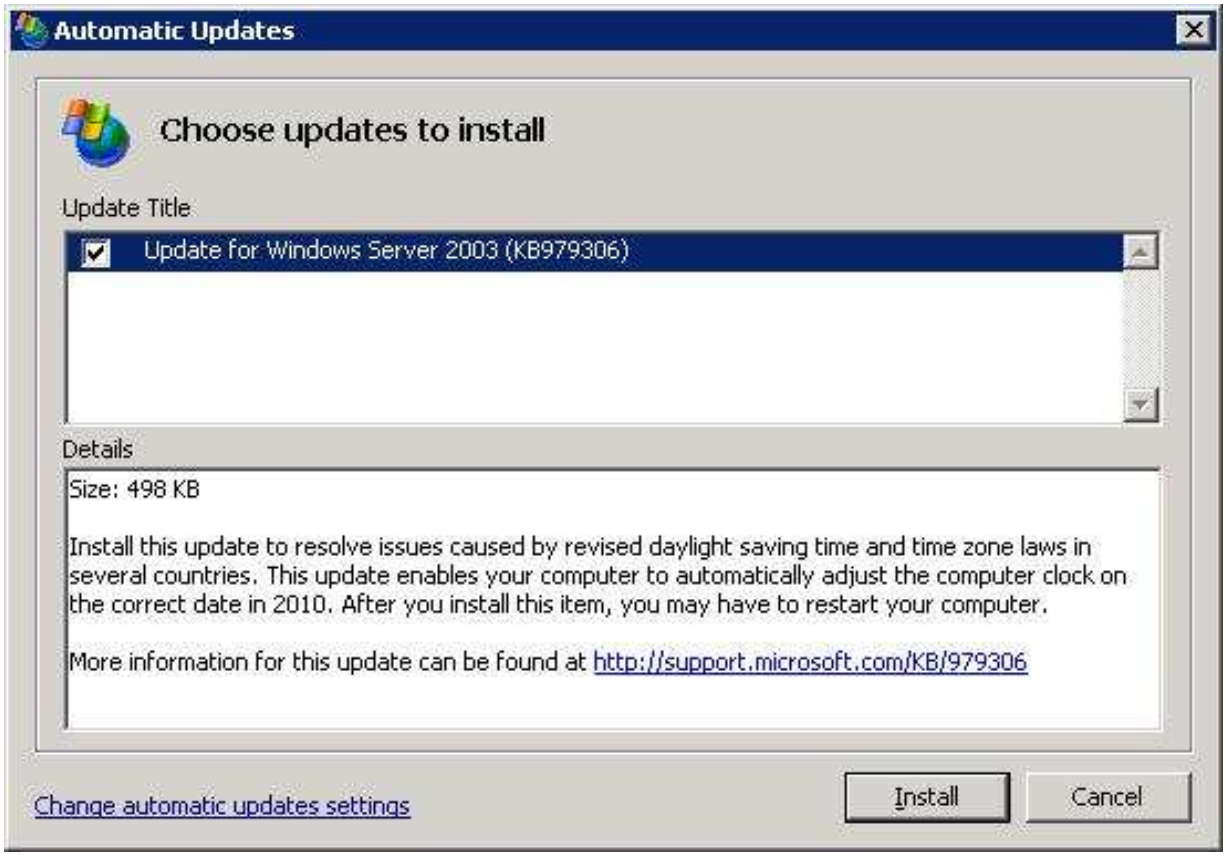
ATTENTION : les définitions des passages à l'heure d'été ou d'hiver changent ! :

- en France, septembre à octobre en 199 ?
- 
- 
- 19 juillet 2007 : Nouvelle Zélande, Haiti, Syrie, Honduras, Turks and Caicos Islands

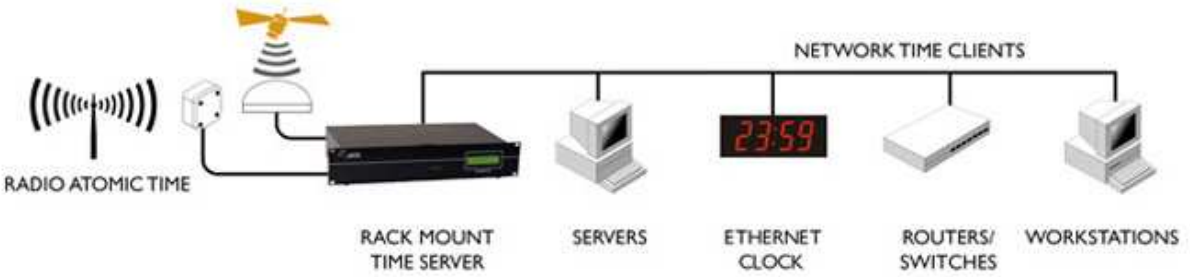
⇒ mise à jour du système indispensable si l'on est concerné !







Autre solution : Network Time Protocol.



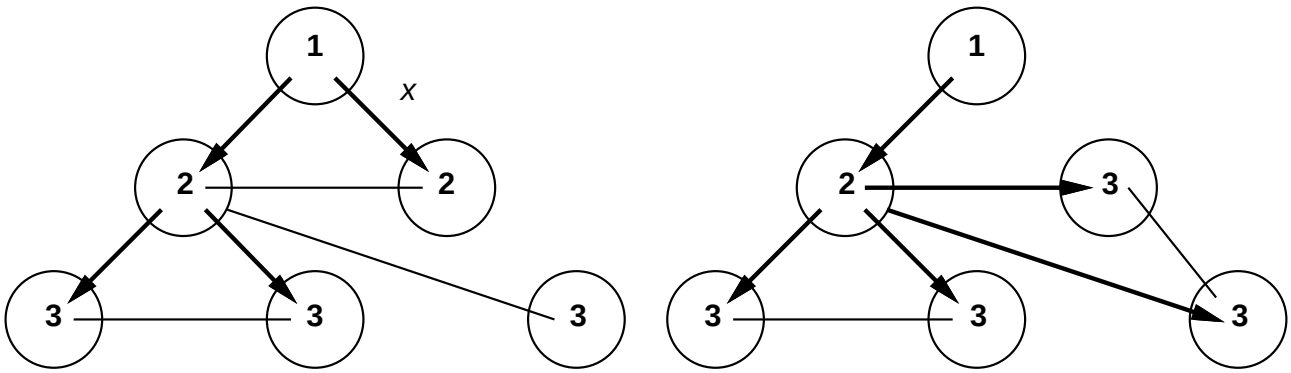

Cf :

- http://www.eecis.udel.edu/~ntp/
- http://www.cru.fr/NTP/
- http://www.sun.com/blueprints/0701/NTP.pdf
- http://www.sun.com/blueprints/0801/NTPpt2.pdf
- http://www.sun.com/blueprints/0901/NTPpt3.pdf



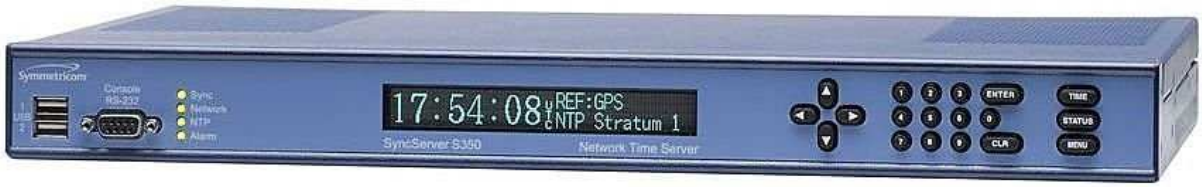


La synchronisation se fait en considérant l'heure de plusieurs machines de références (équipées par exemple d'horloges atomiques) et en tenant compte des délais de transmission via le réseau.



Pour une liste de serveurs de temps en France, se reporter à
`http://www.cru.fr/NTP/serveurs_francais.html`

Une horloge serveur de temps ressemble à ce genre de machines :











En pratique :

1. on choisit des serveurs de temps
2. on désigne une machine du service comme serveur d'heure pour le service
3. on y installe le package NTP ce qui se traduit par lancer le démon `xntpd`
4. sur les machines du service, faire "`ntpdate serveur-ntp`" à intervalle régulier

Garde fou : on ne met pas la machine à l'heure si la différence d'heures est trop grande :

```
Sep 11 02:06:47 example.com xntpd[93]: system event 4 status c621
Sep 11 02:06:47 example.com xntpd[93]: time error -7390.467777 is
large (set clock manually)
Sep 11 02:06:47 example.com xntpd[93]: system event 2 status c634
```



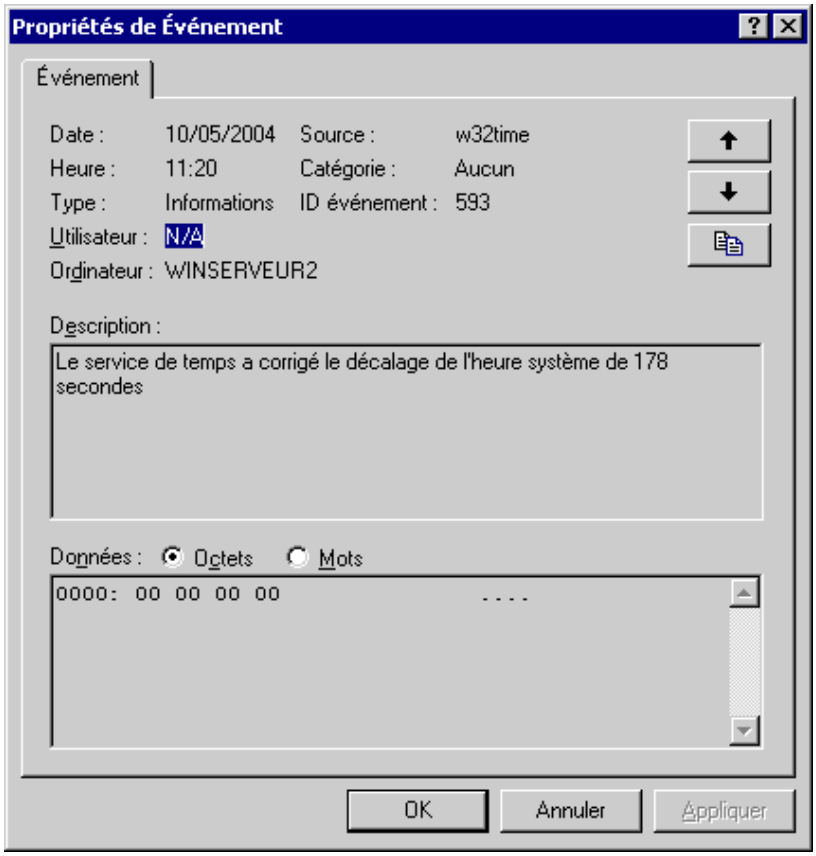


Windows 2000 comporte des outils de synchronisation d'horloges (à cause de Active Directory)

Mettre à jour se fait de la façon suivante :

```
set time /setsntp:ntp.example.com
w32tm -once
```

Trace d'une mise à l'heure :








A un programme sont associés :

- le temps d'exécution de code de bas niveau du système (par exemple écrire sur disque)
- le temps d'exécution de code de haut niveau de l'utilisateur (par exemple calculer une fonction mathématique)

La commande « `time` » fournit ces temps.

Syntaxe : `time commande parametres`

Par exemple :

```
% time gzip -v archive.tar
archive.tar:       26.8% -- replaced with archive.tar.gz

real    0m0.485s
user    0m0.390s
sys     0m0.070s
```

## Attention, subtilité :

- il existe un builtin du shell appelé « time »
- il existe une commande « time » : « /bin/time » ou « /usr/bin/time »

```
% echo $SHELL
/bin/bash
% type time
time is a shell keyword
% time gzip -v archive.tar
archive.tar:      26.8% -- replaced with w.tar.gz

real    0m0.485s
user    0m0.390s
sys     0m0.070s

% /bin/time gzip -v archive.tar
archive.tar:                    26.8% -- replaced with w.tar.gz

real        0.4
user        0.3
sys         0.0
```

Chapitre 15

# *Planification / Programmation de tâches périodiques*





(cf *chronologique*, dieu grec *Chronos*)

L'administrateur doit lancer périodiquement des actions (par exemple nettoyage de « `/tmp` », remise à l'heure par rapport à une horloge de référence, etc.). Il existe un système automatisant ces lancements : **CRON**



Un démon « `crond` » de CRON est lancé par les scripts de démarrage du boot.

◇ Démon « crond »

Sur Linux :

```
% ls -l /etc/rc.d/init.d/crond
-rwxr-xr-x 1 root root  1031 Feb  3 2000 /etc/rc.d/init.d/crond
```

Sur Solaris :

```
% ls -l /etc/init.d/crond
-rwxr--r-- 4 root sys   513 Jul 16 1997 /etc/init.d/crond
```





Les actions à lancer périodiquement sont indiquées dans un fichier texte appelé « **crontab** » au format suivant :

```
minutes heures jour-du-mois mois jour-de-semaine commande
```

- champ 1 : minutes, de 0 à 59.
- champ 2 : heures, de 0 à 23.
- champ 3 : jour du mois, de 1 à 31.
- champ 4 : mois, de 1 à 12.
- champ 5 : jour de la semaine, de 0 (dimanche) à 6 (samedi) (souvent 7 est aussi accepté pour dimanche).
- champ 6 : commande à exécuter.

### ◇ Caractères spéciaux

Le caractère « * » sert de caractère joker pour n'importe quel des champs 1 à 5.

Le caractère « , » sépare les éléments d'une liste de valeurs.
Par exemple :

```
1,3,4,6
```

Le caractère « - » désigne un intervalle entre deux valeurs.
Par exemple :

```
1-4 --> valeurs 1, 2, 3, 4
```

Le caractère « / » permet d'indiquer une fréquence (voir page d'après).
Par exemple :

```
*/10 --> valeurs 0, 10, 20, 30, 40, etc.
1/2 --> valeurs 1, 3, 5, 7, 9, 11, etc.
```

Le caractère « # » introduit un commentaire.





### ◇ Exemples

```
15  * * * * /chemin/vers/ntpdate.sh
0 * * * * /chemin/vers/check-devnull.sh
15 * * * * /chemin/vers/check-devnull.sh
30 * * * * /chemin/vers/check-devnull.sh
45 * * * * /chemin/vers/check-devnull.sh
0 21 * * 1 /chemin/vers/sauvegarde.sh
0 21 * * 3 /chemin/vers/sauvegarde.sh
0 21 * * 5 /chemin/vers/sauvegarde.sh
```

La crontab ci-dessus est donc équivalente à :

```
15  * * * * /chemin/vers/ntpdate.sh
*/15 * * * * /chemin/vers/check-devnull.sh
0 21 * * 1,3,5 /chemin/vers/sauvegarde.sh
```

### ◇ Emplacement des fichiers crontab

L'emplacement des fichiers crontab dépend des systèmes :

| Système | Program |
|---|---|
| AIX | `/var/spool/cron/crontabs/` |
| HP-UX | `/var/spool/cron/crontabs/` |
| FreeBSD | `/var/cron/tabs/` |
| Linux | `/var/spool/cron/` |
| Solaris | `/var/spool/cron/crontabs/` |

Mais il est inutile de connaitre cet emplacement car des commandes le masquent.

Il y a **un fichier crontab par utilisateur** (s'il désire utiliser CRON) dans les directories mentionnés ci-dessus.

Par exemple :

`/var/spool/cron/`**`besancon`**
`/var/spool/cron/`**`root`**





(Question à 100 sous : pourquoi ne faut-il rien programmer entre 2 h et 3h du matin ?)

Les commandes programmés s'exécutent sans terminal associé. Il n'y a donc pas de « `stdin` » ou « `stdout` » associés. Il faut réaliser soi-même les redirections requises car sinon, **tout ce qui est écrit sur « `stdout` » ou « `stderr` » est envoyé par mail au propriétaire du fichier crontab**.

Soit le shell script suivant qui est lancé via crontab :

```
#!/bin/sh
echo Je suis un message normal
echo Je suis un message d'erreur >&2
exit 0
```

Après son exécution, on trouve dans sa boite aux lettres :

```
Date: Sat, 10 Aug 2002 23:20:00 +0200 (MEST)
From: Thierry Besancon <besancon>
To: besancon
Subject: Output from "cron" command

Your "cron" job on solaris.example.com
/tmp/demo

produced the following output:

Je suis un message normal
Je suis un message d'erreur
```





◇ Extensions récentes

On peut initialiser des variables d'environnement qui seront passées aux programmes lancés via la CRONTAB.

Syntaxe simple : « `variable=valeur` »

La variable la plus intéressante à modifier : « `MAILTO` »

`MAILTO=""` **--> pas de mail envoyé**
`MAILTO="besancon@example.com"` **--> mail envoyé à une autre adresse**

Les fichiers crontab ont beau être au format texte, il ne faut pas les éditer manuellement. Il faut passer par l'intermédiaire de la commande « `crontab` ». Il y a deux façons de faire selon les versions de la commande :

1. `crontab -e`
   Cela lance l'éditeur de texte indiqué par la variable d'environnement « `EDITOR` » ou bien « `vi` » par défaut.
2. 
```
crontab -l > /tmp/mon-crontab-à-moi
vi /tmp/mon-crontab-à-moi
crontab /tmp/mon-crontab-à-moi
rm /tmp/mon-crontab-à-moi
```





Un utilisateur ne peut consulter / modifier / supprimer que son fichier CRONTAB.

L'administrateur peut consulter / modifier / supprimer tout fichier CRONTAB d'utilisateur.

Pour root, la syntaxe est donc :

- « **crontab -l [-u autre-utilisateur]** »
- « **crontab -e [-u autre-utilisateur]** »
- « **crontab -r [-u autre-utilisateur]** »

Les fichiers « cron.allow » et « cron.deny » contrôlent l'utilisation de la commande « crontab ».

| Système | Program |
|---|---|
| Linux | /etc/cron.allow<br>/etc/cron.deny |
| Solaris | /etc/cron.d/cron.allow<br>/etc/cron.d/cron.deny |

Ces 2 fichiers ont la même syntaxe : ils contiennent des noms de login.

| cron.allow | cron.deny | Utilisateurs autorisés |
|---|---|---|
| Présent | Présent | Ceux explicitement dans cron.allow |
| Présent | – | Ceux explicitement dans cron.allow |
| – | Présent | Tous sauf ceux dans cron.deny |
| – | – | Uniquement root |

Il est conseillé de ne laisser l'accès à « crontab » qu'à l'utilisateur root.





Sur un LINUX de base, on a en plus la crontab « /etc/crontab ».

Que contient de base la crontab « /etc/crontab » ?

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root nice -n 19 run-part /etc/cron.hourly
02 4 * * * root nice -n 19 run-part /etc/cron.daily
22 4 * * 0 root nice -n 19 run-part /etc/cron.weekly
42 4 1 * * root nice -n 19 run-part /etc/cron.monthly
```

Structure quasi identique sauf :

- champ 1 : minutes, de 0 à 59
- champ 2 : heures, de 0 à 23
- champ 3 : jour du mois, de 1 à 31
- champ 4 : mois, de 1 à 12
- champ 5 : jour de la semaine, de 0 (dimanche) à 6 (samedi)
- champ 6 : **login sous lequel lancer la commande**
- champ 7 : commande à exécuter (anciennement champ 6)



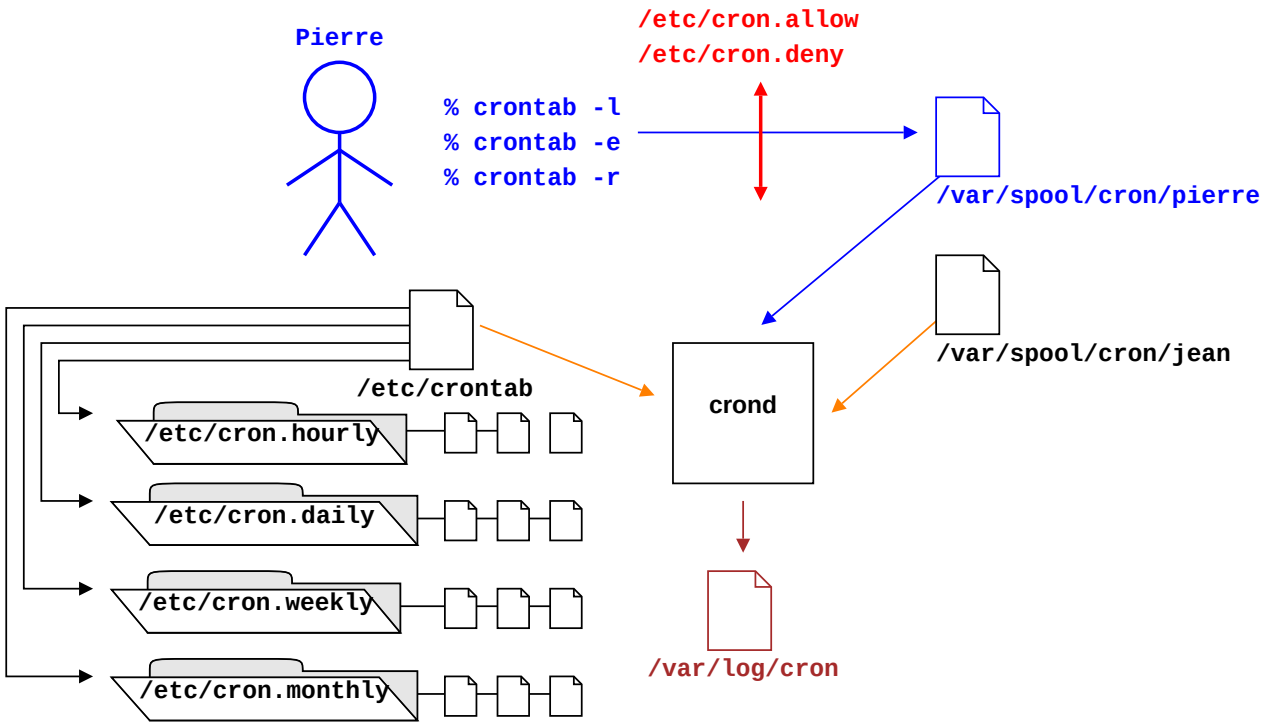


On a :

- « `/etc/cron.hourly` » : répertoire de scripts horaires
- « `/etc/cron.daily` » : répertoire de scripts quotidiens
- « `/etc/cron.weekly` » : répertoire de scripts hebdomadaires
- « `/etc/cron.monthly` » : répertoire de scripts mensuels

Intérêt : plus simple à manipuler : il y a juste à copier un script dans un de ces répertoires.

Au total :






# Chapitre 15 • Planification / Programmation de tâches périodiques

## §15.7 • Programmation de shell scripts pour CRON : `tty -s`

Rappel :

- Tout ce qu'un job CRON écrit sur « `stdout` » ou sur « `stderr` » est envoyé par mail.
- Un shell peut être interactif ou pas ; en interactif « `stdout` » et « `stderr` » sont attachés à un terminal

Comment programmer un shell script pour :

- qu'en interactif, il affiche des infos
- qu'en non interactif via CRON, il n'affiche rien qui ne serait renvoyé par mail

?

Solution : utiliser la commande « `tty` »

Syntaxe : `tty [option]`

Sans option : la commande renvoie le nom du terminal utilisé par l'utilisateur :

```
% tty
/dev/pts/4
```

L'option intéressante : « `-s` »
La commande « `tty -s` » n'affiche rien mais :

- en interactif, son code de retour vaut 0
- en non interactif, son code de retour vaut 1





La solution est donc en générale :

```
#!/bin/sh

if tty -s
then
  # interactif
  commandes
else
  # non interactif
  autres commandes
fi
```

Ordonnanceur commercial « Dollar Universe » (connu sous le nom « `$U` »).

Cf « `http://www.orsyp.com` »








Ci joint dans la version imprimée de ce cours, un article intitué « Tâches planifiées et gestion de credentials sous WINDOWS » du numéro 47 de Janvier/Février 2010 de MISC.

Adresse web : « `http://www.miscmag.com` »





# TÂCHES PLANIFIÉES ET GESTION DE CREDENTIALS SOUS WINDOWS

Roderick ASSELINEAU – rasselineau@atlab.fr

**mots-clés : WINDOWS / TÂCHES PLANIFIÉES / CREDENTIALS / REVERSE ENGINEERING**

*Sous Windows, toute tâche planifiée est authentifiée auprès du système au moment de son exécution. Cet article explique comment Windows gère les credentials associés aux tâches et présente un nouvel outil, alternative à TaskPwDmp, capable de les récupérer de façon fiable sous Windows XP et 2003.*

## 1 Introduction

L'année dernière, Ivanlef0u a présenté sur son blog une suite d'articles originaux **[1] [2]** expliquant le fonctionnement du mécanisme des tâches planifiées et comment l'exploiter pour obtenir de nouveaux credentials. L'outil TaskPwdDmp a été publié à cette même époque pour accompagner ses billets. Malheureusement, pour différentes raisons que nous expliquerons par la suite, il impose quelques lourdes contraintes aux pentesters en conditions réelles. Nous avons donc étudié plus en profondeur les mécanismes utilisés par Windows pour nous en affranchir. Cet article expose les résultats de nos recherches.

Après avoir rappelé brièvement le fonctionnement du gestionnaire des tâches (*Task Manager*) de Windows, nous expliquerons l'attaque proposée par Ivanlef0u dans ses articles et implémentée successivement dans TaskPwDmp et TaskPwDmp-ng. Nous analyserons ensuite quelques fonctions internes du task manager et présenterons alors une toute autre approche ayant permis d'aboutir à un nouvel outil, TaskPwDmp2.

## 2 Quelques rappels sur le fonctionnement des tâches planifiées dans Windows

La gestion des tâches sous Windows XP et 2003 est entièrement prise en charge par le service Task Manager. Comme bon nombre de services sous Windows, celui-ci est créé à partir d'une bibliothèque (ici `schedsvc.dll`) chargée par une instance du processus `svchost`. Tout client souhaitant créer, supprimer ou énumérer des tâches communique alors avec ce service par l'intermédiaire d'une interface RPC. Le cas de Windows 2000 est un peu particulier puisque le processus chargé de la planification des tâches est dédié (MSTask.exe) mais les interfaces RPC sont les mêmes.

Pour des raisons vraisemblablement historiques, les Windows actuels cumulent plusieurs interfaces RPC permettant une gestion plus ou moins directe des tâches. Bien que les mécanismes internes sous-jacents soient très différents d'une interface à l'autre, celles-ci manipulent un même *task store* (en général le répertoire `%SystemRoot%\Tasks`) dans lequel chaque tâche planifiée correspond à un fichier spécial muni d'une extension `.job`. Ces interfaces sont les suivantes :

- Atsvc : historiquement, la première interface. Elle fournit, à travers un jeu d'API simple, les `NetScheduleJob*()`, un contrôle assez complet et relativement intuitif des tâches. Il n'est cependant pas possible de lancer une planification de tâches sans avoir les droits administrateur, ce qui est contraignant.
- SASec : apparue avec Windows 2000, elle rend possible la planification des tâches pour un utilisateur donné. Quel que soit l'utilisateur sous le compte duquel tourne la tâche, la saisie des credentials appropriés est nécessaire lors de la programmation. Contrairement à l'interface précédente, celle-ci ne s'occupe absolument pas de la gestion (création, modification, suppression) des fichiers `.job` du task store qui doit être faite alternativement. Elle se focalise en effet essentiellement sur l'obtention et la modification d'informations liées aux tâches.

- ItaskSchedulerService : cette dernière interface est apparue avec Windows Vista. Bien que nous ne l'ayons pas vérifié, il est vraisemblable qu'elle ait été conservée dans Windows Seven. Ne faisant pas l'objet de cet article, nous ne nous attarderons pas dessus.

Nous invitons le lecteur désireux d'obtenir plus d'informations sur ces interfaces à consulter l'excellente documentation en ligne fournie par Microsoft **[3]**.

Sous Windows, il y a par défaut deux outils permettant la création de tâches planifiées : la commande en ligne **at.exe** et le shell **explorer.exe** (munie d'une extension spéciale pour la gestion du task store). Chacun de ces programmes a sa propre façon de communiquer avec le task manager et le schéma ci-dessous résume ces interactions pour Windows XP et 2003 :

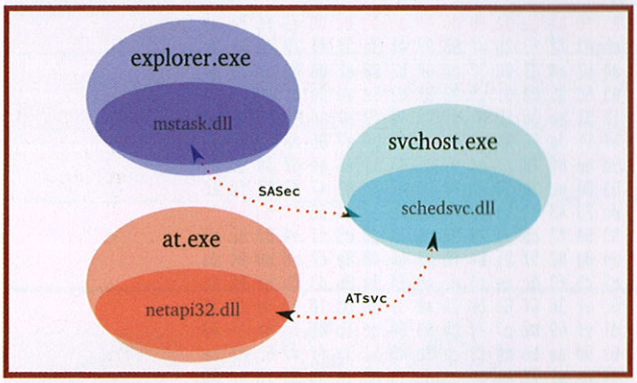



*Fig. 1 : RPC et planification des tâches sous Windows*

Dans la mesure où seule l'interface SASec est susceptible de manipuler des credentials saisis par l'utilisateur, elle est celle sur laquelle se focalise cet article.

## 3 La première méthode : DLL injection dans le processus svchost

En analysant la bibliothèque **schedsvc.dll**, on peut remarquer la présence d'une fonction au nom très évocateur, **DecryptCredentials()**, dont on sait grâce aux symboles Microsoft que le 4ème argument est un pointeur sur structure de type **_JOB_CREDENTIALS**. Comme cette structure est passée par référence, on peut imaginer sans trop risquer de se tromper que les credentials de la tâche sont stockés dedans au retour de la fonction. Un fait que l'expérience confirme. Pour obtenir les credentials de la tâche, il suffit donc de « hooker » cette fonction et de trouver un moyen de déclencher son appel.

Un examen rapide de **schedsvc.dll** montre que la fonction est appelée chaque fois qu'une tâche doit être lancée par le task manager. Plus intéressant, elle est également appelée par **SAGetAccountInformation()** et **SASetAccountInformation()**, deux fonctions de l'interface SASec permettant réciproquement l'obtention et la modification des informations liées à une tâche. En pratique, TaskPwDmp repose sur l'utilisation de la classe **Itask** dont l'une des méthodes, **GetAccountInformation()**, effectue un appel RPC indirect à **SAGetAccountInformation()**.

Plus précisément, TaskPwDmp fonctionne de la façon suivante :

- Vérification de la présence du task manager. Si le task manager est arrêté ou si le task store ne contient aucun fichier, TaskPwDmp s'interrompt.
- Chargement d'une bibliothèque dans le contexte du processus **svchost**. Après avoir déterminé le PID du processus **svchost** ayant chargé **schedsvc.dll**, TaskPwDmp fait appel à la très classique **CreateRemoteThread()** pour charger la bibliothèque **TaskPwDmp.dll** dans le contexte du task manager. Celle-ci ouvre alors un port LPC pour communiquer avec TaskPwDmp puis « hook » la fonction **DecryptCredentials()** en utilisant la bibliothèque **detours** de Microsoft.
- Création d'un thread à l'écoute du port LPC. Lorsque la fonction **DecryptCredentials()** est appelée, le hook associé permet de récupérer les valeurs des logins et mots de passe puisqu'ils sont disponibles à des offsets constants dans la structure de type **_JOB_CREDENTIALS** au retour de la fonction. TaskPwDmp crée donc un thread dédié à l'écoute de ce port LPC pour recevoir les informations envoyées par le hook.
- Déclenchement arbitraire de **DecryptCredentials()**. TaskPwDmp utilise les classes **ITask** et **ITaskScheduler** pour chacune des tâches du task store. TaskPwDmp affiche les credentials interceptés par le hook sur la sortie standard.
- Déchargement de **TaskPwDmp.dll**. TaskPwDmp crée un nouveau thread dans le processus **svchost** qui appelle **FreeLibrary()** pour décharger **TaskPwDmp.dll**. Au moment du déchargement, celle-ci retire le hook en utilisant **detours** et se décharge de la mémoire. Le processus **svchost** revient alors dans un état de fonctionnement normal.

## 4 Les limitations de TaskPwDmp

La première version de TaskPwDmp **[1]** souffre d'une imperfection majeure : l'utilisation d'offsets dans la bibliothèque **schedsvc.dll** pour calculer l'adresse de **DecryptCredentials()**. Compte tenu de la nature intrusive de l'outil (hook de fonction en mémoire) et de la criticité du processus exploité (svchost), une erreur de calcul entraîne irrémédiablement un état instable de la machine cible, chose peu souhaitable en audit. Conscient de ce problème, l'auteur a donc proposé trois mois plus tard une seconde version **[2]**, TaskPwDmp-ng,

Chapitre 16

# Traces/Journaux système : SYSLOG





Les fichiers journaux sont regroupés dans « `/var/log` » par convention.

Par exemple :

- « `/var/log/messages` »
- « `/var/log/secure` »
- « `/var/log/lastlog` » (commande « `lastlog` » associée)
- « `/var/log/wtmp` » (commandes « `last` » et « `who` » associées)
- « `/var/log/dmesg` » (commande « `dmesg` » associée)
- les applications peuvent avoir leurs propres fichiers de log :
  - « `/var/log/cups/access_log` »
  - « `/var/log/cups/error_log` »
  - etc.

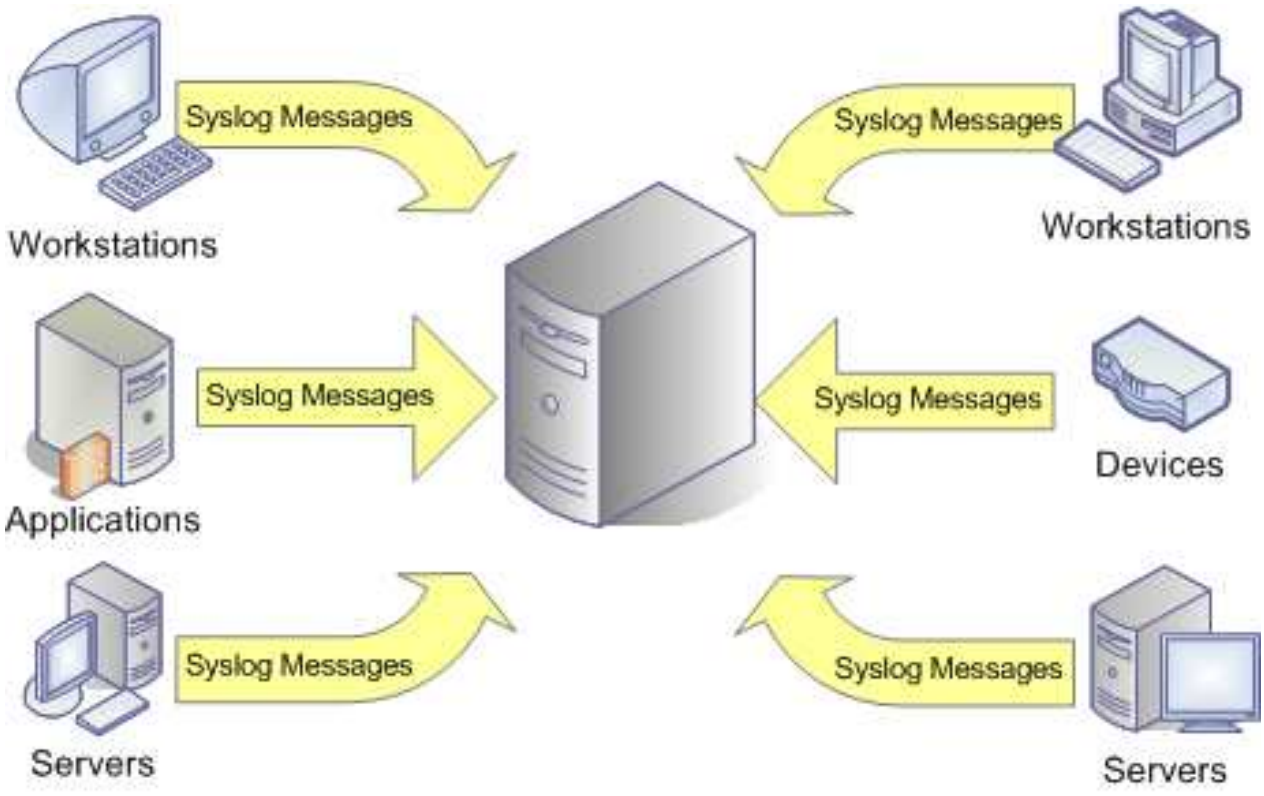
(en anglais *System Log*)

SYSLOG est le mécanisme standard de distribution des messages envoyés par le noyau et les programmes (systèmes ou autres) :

- Il fonctionne au dessus de UDP (port 514), donc en mode non connecté. Cf RFC3164.
- Il agit comme un dispatcher de messages.
- Il n'utilise aucune authentification des clients SYSLOG.
- Le programme associé s'appelle `syslogd`
- D'origine BSD. Améliorations sur certains LINUX.







```
Nov 11 02:00:02 example.org pppd: pppd 2.4.0b1 started by besancon, uid 1000
Nov 11 02:00:21 example.org pppd: Serial connection established.
Nov 11 02:00:21 example.org pppd: Using interface sppp0
Nov 11 02:00:21 example.org pppd: Connect: sppp0 <--> /dev/ttya
Nov 11 02:00:22 example.org pppd: LCP: Rcvd Code-Reject for Identification id 13
Nov 11 02:00:22 example.org pppd: Remote message: Login ok
Nov 11 02:00:22 example.org pppd: local  IP address 192.168.0.30
Nov 11 02:00:22 example.org pppd: remote IP address 192.168.0.100
Nov 11 02:05:07 example.org spppasyn: sppp0: bad fcs (len=352)
Nov 11 02:05:59 example.org pppd: Terminating on signal 15.
Nov 11 02:05:59 example.org pppd: Connection terminated.
Nov 11 02:05:59 example.org pppd: Connect time 5.7 minutes.
Nov 11 02:05:59 example.org pppd: Sent 60962 bytes (4074 packets), received 108649
(4142 packets).
Nov 11 02:06:00 example.org pppd: Exit.

Nov 11 11:05:41 example.org usba: /pci@1f,0/usb@c,3 (ohci0): getting device status
Nov 11 11:05:41 example.org usba: USB-device: storage@3, scsa2usb0 at bus address
Nov 11 11:05:41 example.org usba:  M-Systems DiskOnKey 0415A20E29003852
Nov 11 11:05:41 example.org genunix: scsa2usb0 is /pci@1f,0/usb@c,3/storage@3
Nov 11 11:05:41 example.org genunix: /pci@1f,0/usb@c,3/storage@3 (scsa2usb0) onlin
Nov 11 11:05:41 example.org scsi: sd16 at scsa2usb0: target 0 lun 0
Nov 11 11:05:41 example.org genunix: sd16 is /pci@1f,0/usb@c,3/storage@3/disk@0,0
```





```
MySelf:~# cat /var/log/syslog
Dec  7 06:25:05 localhost syslogd 1.4.1#17: restart.
Dec  7 06:39:01 localhost /USR/SBIN/CRON[928]: (root) CMD (  [ -d /var/lib/php4 ] && find /var/lib/php4
-type f -cmin +$(/usr/lib/php4/maxlifetime) -print0 | xargs -r -0 rm)
Dec  7 07:03:07 localhost -- MARK --
Dec  7 07:17:01 localhost /USR/SBIN/CRON[950]: (root) CMD (   run-parts --report /etc/cron.hourly)
Dec  7 15:02:11 localhost proftpd[1182]: 192.168.0.3 (84.90.94.31[84.90.94.31]) - FTP session opened.
Dec  7 15:02:12 localhost proftpd[1182]: 192.168.0.3 (84.90.94.31[84.90.94.31]) - mod_delay/0.4:
delaying for 64 usecs
```

**proftpd[1182]: 84.90.94.31 - FTP session opened**

```
Dec  7 15:02:12 localhost proftpd[1182]: 192.168.0.3 (84.90.94.31[84.90.94.31]) - FTP session closed.
Dec  7 15:17:01 localhost /USR/SBIN/CRON[1194]: (root) CMD (   run-parts --report /etc/cron.hourly)
Dec  7 15:39:01 localhost kernel: USB Mass Storage device found at 2
Dec  7 16:03:08 localhost -- MARK --
```

**kernel: USB Mass Storage device found at 2**

Pour trier les messages SYSLOG, ces messages texte sont transmis en interne associés à une étiquette.

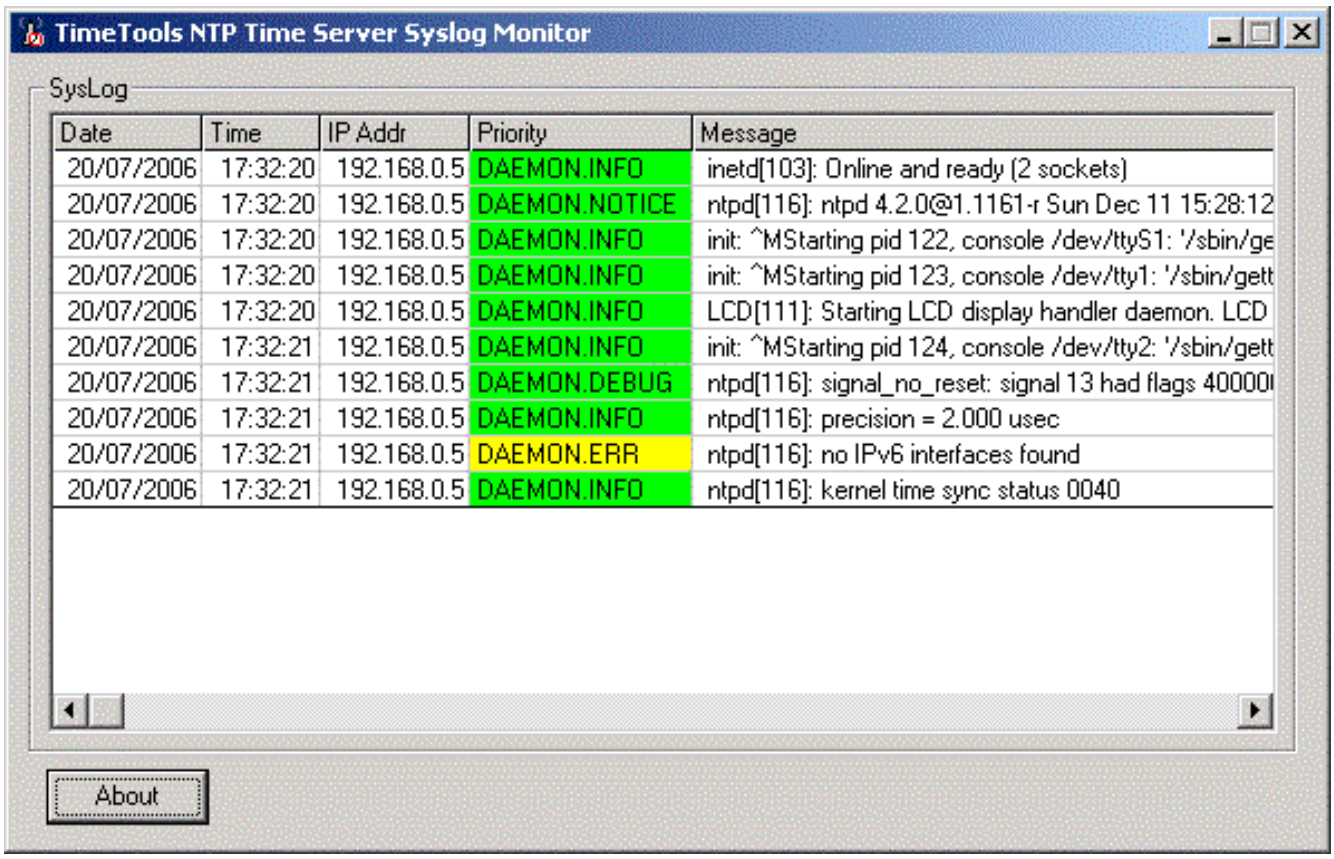
L'étiquette est composée :

- de la date
- du nom de la classe générique du programme émetteur : c'est le **facility**
- du nom du niveau de la priorité du message : c'est le **level**

Cf « `<syslogd.h>` »







◇ Facility :

| Nom | Valeur |
|---|---|
| kern | 0 << 3 |
| user | 1 << 3 |
| mail | 2 << 3 |
| daemon | 3 << 3 |
| auth | 4 << 3 |
| syslog | 5 << 3 |
| lpr | 6 << 3 |
| news | 7 << 3 |
| uucp | 8 << 3 |
| local0 à local7 | de 16 << 3 à 23<< 3 |

« * » est un caractère joker.





◇ level :

| Nom | Valeur |
|---|---|
| emerg | 0 |
| alert | 1 |
| crit | 2 |
| err | 3 |
| warning | 4 |
| notice | 5 |
| info | 6 |
| debug | 7 |

« * » est un caractère joker (non disponible sur Solaris)

**ATTENTION : Un niveau de sécurité englobe les niveaux de sécurité supérieurs (« err » englobe « crit », « alert » et « emerg » ; etc.)**

SYSLOG agit comme un routeur des messages :

- Les messages sont émis par les programmes et reçus sur chaque machine par le démon « `syslogd` »
- « `syslogd` » décide quoi faire des messages sur la base de son fichier de configuration, « verb !/etc/syslog.conf ! »

Syntaxe du fichier « `/etc/syslog.conf` » :

```
facility.level[;facility.level]*\taction
```

Ci-dessus « `\t` » indique le caractère tabulation (notation du « `printf` » du langage C).
**La tabulation est indispensable** !





◇ possibilités de distribution des messages : actions

- distribution des messages dans des fichiers
  Indiquer pour cela un nom de fichier via son path absolu (commençant donc par un « `/` ») ;
- aiguillage des messages vers le démon d'une autre machine
  Indiquer pour cela un nom de machine ou son adresse IP, précédé de « `@` » ;
- affichage des messages sur un terminal particulier
  Indiquer pour cela le path absolu du terminal ;
- affichage des messages sur les terminaux de certains utilisateurs énumérés
  Indiquer pour cela une liste des logins des utilisateurs (séparés par une virgule) ; par exemple `root, operator`
  utiliser « `*` » pour atteindre tous les utilisateurs connectés

### ◇ Exemple

```
kern.*                  /var/adm/kernel
kern.crit               /dev/console
kern.crit               @finlandia
```

Signification :

- tous les messages émis par les utilitaires de type « kernel » vont dans « /var/adm/kernel »
- tous les messages émis par les utilitaires de type « kernel » et de sévérité « crit », « alert », « emerg » (sévérités supérieures ou égales à « crit ») sont recopiés sur la console de la machine
- tous les messages émis par les utilitaires de type « kernel » et de sévérité « crit », « alert », « emerg » (sévérités supérieures ou égales à « crit ») sont envoyés à la machine « finlandia »





### ◇ Extensions non standard

Des extensions se rencontrent de plus en plus souvent :

- écriture « = » précédant le level et indiquant que seul ce niveau est pris en compte et pas les niveaux supérieurs
- écriture « ! » précédant le level et indiquant que ce niveau et les niveaux supérieurs ne sont pas pris en compte
- écriture « != » précédant le level et indiquant que ce niveau et celui-là seul n'est pas pris en compte

Personnellement, le fichier de mes machine clientes contient :

```
*.*                 @sysloghost.example.com
```

ou version Solaris :

```
*.debug             @sysloghost.example.com
```

Il redirige tout message vers la machine appelée `sysloghost.example.com` qui centralise et stocke ce message.





• Sur la machine appelée `sysloghost.example.com`, on stockera par exemple les messages de la façon suivante :

```
*.*                 /var/log/syslog
```

ou version Solaris :

```
*.debug             /var/log/syslog
```

**Il vaut mieux avoir trop de messages que pas du tout ou pas ceux qui sont intéressants.**

`syslogd` est lancé par les scripts de démarrage.

Si l'on modifie le fichier « `/etc/syslog.conf` », la modification est prise en compte en envoyant le signal « `SIGHUP` » au processus `syslogd` :

```
# ps -edf | grep syslogd
    root   339     1  0   Aug 04 ?        0:18 /usr/sbin/syslogd
# kill -HUP 339
```

ce qui se traduit par l'émission du message SYSLOG :

```
Aug 19 15:02:34 example.com syslogd: configuration restart
```





Le signal « `SIGTERM` » arrête le processus ce qui se traduit par l'émission du message SYSLOG :

```
Aug 19 15:02:54 example.com syslogd: going down on signal 15
```

(autre façon de faire : « `pkill -HUP syslogd` »)

Programme C minimal :

```
#include <syslog.h>

main(int argc, char *argv[])
  {
     openlog("ftpd", LOG_PID, LOG_DAEMON);
     syslog(LOG_INFO, "Connection from host %s", "333");
     syslog(LOG_INFO|LOG_LOCAL2, "error: %m");
     closelog();
  }
```

Cela donne les messages :

```
Nov 11 11:33:09 example.org ftpd[10658]: Connection from host 333
Nov 11 11:33:09 example.org ftpd[10658]: error: Bad file number
```





Programmer en langage C n'est pas pratique si on veut utiliser SYSLOG au niveau du shell.

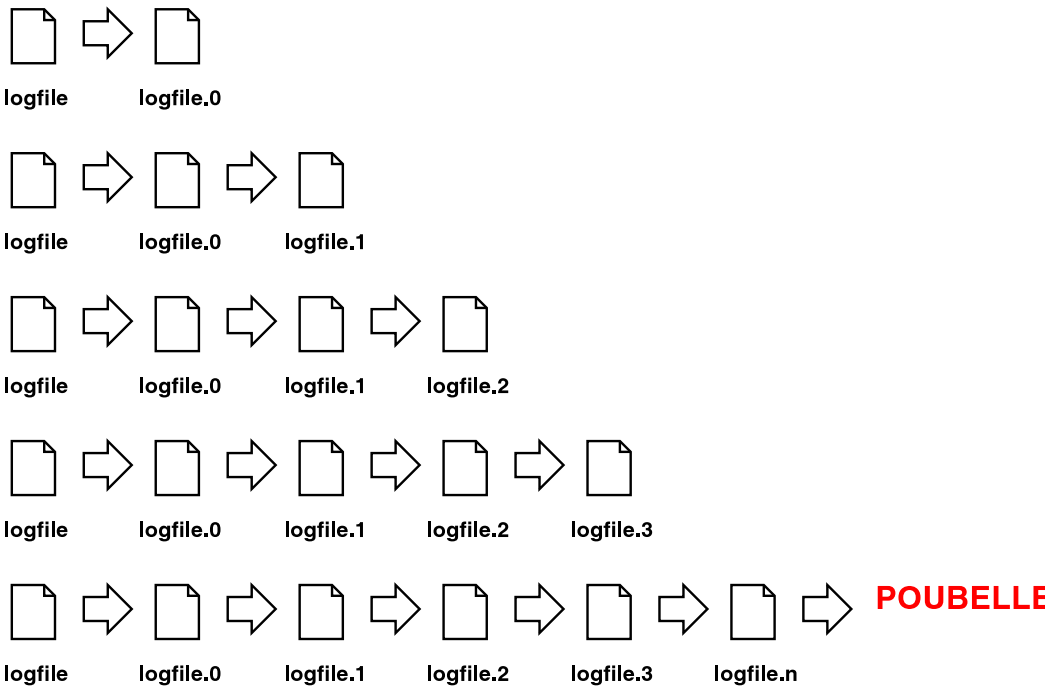
⇒ il existe la commande « `logger` » au niveau du shell.

Syntaxe : `logger [-p facility.level] message`

```
% logger -p local3.notice Essai de la commande logger

% grep Essai /var/log/messages
Nov 16 14:56:18 example.com ftpd[1098]: Essai en langage C
Nov 16 14:58:03 example.com besancon: Essai de la commande logger
```

Mécanisme traditionnel : rotation des fichiers de log :








Nettoyage régulier (via CRON) des logs : commande « `logrotate` »

Fichier de configuration « `/etc/logrotate.conf` ».

Opérations faisables :

- copie du fichier de log dans un autre fichier (en général ajout d'un suffixe numérique)
- compression du log
- transmission du log
- vidage du log
- suppression de vieux logs

Décisions prises selon :

- la taille du log
- la date (par exemple tous les jours, toutes les semaines, etc.)

Rien ne sert de configurer « `syslogd` » si l'on n'inspecte pas les logs recueillis.

Plusieurs méthodes :

- méthode artisanale via « `tail -f fichier-de-log` » pour afficher en temps réél l'arrivée des messages
- méthode plus sophistiquée via certains programmes dédiés à cela ; liste non exhautive :
  - logiciel `xlogmaster` ; cf `http://www.gnu.org/software/xlogmaster/`
  - logiciel `awacs` ; cf `http://www.gnu.org/software/awacs/`
  - logiciel `swatch` ; cf `ftp://ftp.stanford.edu/general/security-tools/swatch/`

Pour résoudre le problème des logs qui grossissent : « `trimlog` » `http://online.securityfocus.com/tools/118`





Il y a plusieurs autres implémentations de démons SYSLOG.

• Le démon « `syslogd` » de Digital Unix introduit un mot clef spécial `syslog.dated` :

```
kern.debug              /var/adm/syslog.dated/kern.log
mail.debug              /var/adm/syslog.dated/mail.log
daemon.debug            /var/adm/syslog.dated/daemon.log
```

Le mot « `syslog.dated` » sera remplacé par la date du jour ce qui donnera :

```
/var/adm/syslog.dated/13-Sep-11:57/*.log
/var/adm/syslog.dated/14-Sep-11:57/*.log
/var/adm/syslog.dated/15-Sep-11:58/*.log
```

- `nsyslogd` disponible à l'URL

`http://coombs.anu.edu.au/~avalon/nsyslog.html`

- Syslog-ng tries to fill the gaps original syslogd's were lacking :
  - powerful configurability
  - filtering based on message content
  - message integrity, message encryption (not yet implemented in 1.4.x)
  - portability
  - better network forwarding

`http://www.balabit.hu/products/syslog-ng/`



## Chapitre 16 • Traces/Journaux système : SYSLOG
### §16.13 • (Windows : : SYSLOG)

Windows NT, 2000, XP utilisent des mécanismes de journaux et non pas SYSLOG. Les événements sont des données en binaire.

Mais il existe des programmes apportant SYSLOG à Windows.

◇ BackLog




On voit bien sur UNIX :

```
Nov 11 21:53:08 dell.example.org Application 14 Tue Nov 11 21:53:06 2003
23      Norton AntiVirus        N/A     N/A     Information     VAGABOND
      Protection temps réel Symantec AntiVirus chargée.
```

Gratuit... Cf

`http://www.intersectalliance.com/projects/BackLogNT/setup.ex`



## Chapitre 16 • Traces/Journaux système : SYSLOG
### §16.14 • Législation sur les logs

**A completer...**

Ci joint dans la version imprimée de ce cours, un chapitre gratuit d'un livre traitant de SYSLOG.

Adresse web du document fourni :
« `http://www.oreilly.com/catalog/bssrvrlnx/chapter/ch10.pdf` »

TOOLS & BEST PRACTICES FOR BASTION HOSTS

# *Building* SECURE SERVERS *with* LINUX



O'REILLY®

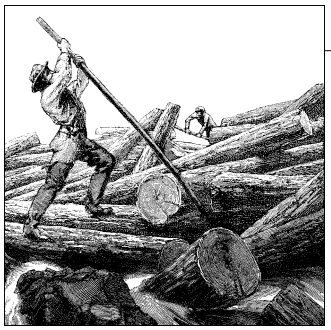
MICHAEL D. BAUER

TITLE Page iii Wednesday, October 16, 2002 6:14 PM

# Building SECURE SERVERS with LINUX

*Michael D. Bauer*

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo



CHAPTER 10

# System Log Management and Monitoring

Whatever else you do to secure a Linux system, it must have comprehensive, accurate, and carefully watched logs. Logs serve several purposes. First, they help us troubleshoot virtually all kinds of system and application problems. Second, they provide valuable early-warning signs of system abuse. Third, after all else fails (whether that means a system crash or a system compromise), logs can provide us with crucial forensic data.

This chapter is about making sure your system processes and critical applications log the events and states you're interested in and dealing with this data once it's been logged. The two logging tools we'll cover are syslog and the more powerful Syslog-ng ("syslog new generation"). In the monitoring arena, we'll discuss Swatch (the Simple Watcher), a powerful Perl script that monitors logs in real time and takes action on specified events.

## syslog

syslog is the tried-and-true workhorse of Unix logging utilities. It accepts log data from the kernel (by way of *klogd*), from any and all local process, and even from processes on remote systems. It's flexible as well, allowing you to determine what gets logged and where it gets logged to.

A preconfigured syslog installation is part of the base operating system in virtually all variants of Unix and Linux. However, relatively few system administrators customize it to log the things that are important for their environment and disregard the things that aren't. Since, as few would dispute, information overload is one of the major challenges of system administration, this is unfortunate. Therefore, we begin this chapter with a comprehensive discussion of how to customize and use syslog.

### Configuring syslog

Whenever *syslogd*, the syslog daemon, receives a log message, it acts based on the message's type (or "facility") and its priority. syslog's mapping of actions to facilities

## What About klogd?

One daemon you probably won't need to reconfigure but should still be aware of is *klogd*, Linux's kernel log daemon. This daemon is started automatically at boot time by the same script that starts the general system logger (probably */etc/init.d/syslogd* or */etc/init.d/sysklogd*, depending on which Linux distribution you use).

By default, *klogd* directs log messages from the kernel to the system logger, which is why most people don't need to worry about *klogd*: you can control the handling of kernel messages by editing the configuration file for *syslogd*.

This is also true if you use Syslog-ng instead of syslog, but since Syslog-ng accepts messages from a much wider variety of sources, including */proc/kmsg* (which is where *klogd* receives its messages), some Syslog-ng users prefer to disable *klogd*. Don't do so yourself unless you first configure Syslog-ng to use */proc/kmsg* as a source.

*klogd* can be invoked as a standalone logger; that is, it can send kernel messages directly to consoles or a log file. In addition, if it isn't already running as a daemon, *klogd* can be used to dump the contents of the kernel log buffers (i.e., the most recent kernel messages) to a file or to the screen. These applications of *klogd* are especially useful to kernel developers.

For most of us, it's enough to know that for normal system operations, *klogd* can be safely left alone (that is, left with default settings and startup options—*not* disabled). Just remember that when you use syslog in Linux, all kernel messages are handled by *klogd* first.

and priorities is specified in */etc/syslog.conf*. Each line in this file specifies one or more facility/priority selectors followed by an action; a selector consists of a facility or facilities and a (single) priority.

In the following *syslog.conf* line in Example 10-1, `mail.notice` is the selector and `/var/log/mail` is the action (i.e., "write messages to */var/log/mail*").

*Example 10-1. Sample syslog.conf line*

```
mail.notice                /var/log/mail
```

Within the selector, `mail` is the facility (message category) and `notice` is the level of priority.

### Facilities

Facilities are simply categories. Supported facilities in Linux are *auth*, *auth-priv*, *cron*, *daemon*, *kern*, *lpr*, *mail*, *mark*, *news*, *syslog*, *user*, *uucp*, and *local0* through *local7*. Some of these are self-explanatory, but the following are of special note:

*auth*
: Used for many security events.

*auth-priv*
: Used for access-control-related messages.

*daemon*
: Used by system processes and other daemons.

*kern*
: Used for kernel messages.

*mark*
: Messages generated by *syslogd* itself, which contain only a timestamp and the string `--MARK--`; to specify how many minutes should transpire between marks, invoke *syslogd* with the *-m [minutes]* flag.

*user*
: The default facility when none is specified by an application or in a selector.

*ocal7*
: Boot messages.

*
: Wildcard signifying "any facility."

*none*
: Wildcard signifying "no facility."

### Priorities

Unlike facilities, which have no relationship to each other, priorities are hierarchical. Possible priorities in Linux are (in increasing order of urgency): *debug*, *info*, *notice*, *warning*, *err*, *crit*, *alert*, and *emerg*. Note that the "urgency" of a given message is determined by the programmer who wrote it; facility and priority are set by the programs that generate messages, not by syslog.

As with facilities, the wildcards * and *none* may also be used. Only one priority or wildcard may be specified per selector. A priority may be preceded by either or both of the modifiers, = and *!*.

If you specify a single priority in a selector (without modifiers), you're actually specifying that priority *plus* all higher priorities. Thus the selector `mail.notice` translates to "all mail-related messages having a priority of *notice* or higher," i.e., having a priority of *notice*, *warning*, *err*, *crit*, *alert*, or *emerg*.

You can specify a single priority by prefixing a = to it. The selector `mail.=notice` translates to "all mail-related messages having a priority of *notice*." Priorities may also be negated: `mail.!notice` is equivalent to "all mail messages except those with priority of *notice*or higher," and `mail.!=notice` corresponds to "all mail messages except those with the priority *notice*."

### Actions

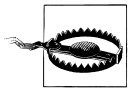
In practice, most log messages are written to files. If you list the full path to a filename as a line's action in *syslog.conf*, messages that match that line will be appended to that file. (If the file doesn't exist, syslog will create it.) In Example 10-1, we instructed syslog to send matched messages to the file */var/log/mail*.

You can send messages other places too. An action can be a file, a named pipe, a device file, a remote host, or a user's screen. Pipes are usually used for debugging. Device files that people use are usually TTYs. Some people also like to send security information to */dev/lp0*—i.e., to a local line printer. Logs that have been printed out can't be erased or altered by an intruder, but they also are subject to mechanical problems (paper jams, ink depletion, etc.) and are harder to parse if you need to find something in a hurry.

Remote logging is one of the most useful features of syslog. If you specify a hostname or IP address preceded by an @ sign as a line's action, messages that match that line will be sent to UDP port 514 on that remote host. For example, the line:

```
*.emerg             @mothership.mydomain.org
```

will send all messages with *emerg* priority to UDP port 514 on the host named *mothership.mydomain.org*. Note that the remote host's (in this example, *mothership's*) *syslogd* process will need to have been started with the `-r` flag for it to accept your log messages. By default, *syslogd* does *not* accept messages from remote systems.



syslog has no access-control mechanism of its own: if you enable the reception of remote messages with the `-r` flag, your host will accept messages on UDP port 514 from any and all remote computers. See the end of this section for some advice on how to mitigate this.

If you run a central log server, which I highly recommend, you'll want to consider some sort of access controls on it for incoming messages. At the very least, you should consider *tcpwrappers*' "hosts access" (source-IP-based) controls or maybe even local firewall rules (*ipchains* or *iptables*).

### More sophisticated selectors

You can list multiple facilities separated by commas in a single *syslog.conf* selector. To extend Example 10-1 to include both mail and uucp messages (still with priority *notice* or higher), you could use this line (Example 10-2).

*Example 10-2. Multiple facilities in a single selector*

```
mail,uucp.notice    /var/log/mail
```

The same is *not* true of priorities. Remember that only one priority or priority wildcard may be specified in a single selector.

## Stealth Logging

Lance Spitzner of the Honeynet Project (*http://www.honeynet.org*) suggests a trick that's useful for honey (decoy) nets and maybe even for production DMZs: "stealth logging." This trick allows a host connected to a hub or other shared medium to send its log files to a non-IP-addressed system that sees and captures the log messages but can't be directly accessed over the network, making it much harder for an intruder on your network to tamper with log files.

The idea is simple: suppose you specify a bogus IP address in a *syslog.conf* action (i.e., an IP address that is legitimate for your host's LAN but isn't actually used by any host running *syslogd*). Since syslog messages are sent using the "connectionless" (one-way) UDP protocol, the sending host doesn't expect any reply when it sends a log message.

Furthermore, assuming your DMZ hosts are connected to a shared medium such as a hub, any syslog messages sent over the network will be broadcast on the local LAN. Therefore, it isn't necessary for a central log server on that LAN to have an IP address: the log server can passively "sniff" the log messages via *snort*, *ethereal*, or some other packet sniffer.

Obviously, since an IP-addressless stealth logger won't be accessible via your usual IP-based remote administration tools, you'll need console access to that host to view your logs. Alternatively, you can add a second network interface to the stealth logger, connecting it to a dedicated management network or directly to your management workstation via crossover cable.

In addition to configuring each DMZ host's *syslog.conf* file to log to the bogus IP, you'll also need a bogus ARP entry added to the network startup script on each sending host. If you don't, each system will try in vain to learn the Ethernet address of the host with that IP, and it won't send any log packets.

For example, if you want a given host to pretend to send packets to the bogus IP 192.168.192.168, then in addition to specifying `@192.168.192.168` as the action on one or more lines in */etc/syslog.conf*, you'll need to enter this command from a shell prompt:

```
arp -s 192.168.192.168 03:03:03:31:33:77
```

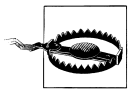
This is not necessary if you send log packets to a "normal" log host (e.g., if 192.168.192.168 is the IP address of a host running *syslogd* with the `-r` flag.)

You may, however, specify multiple selectors separated by semicolons. When a line contains multiple selectors, they're evaluated from left to right: you should list general selectors first, followed by more specific selectors. You can think of selectors as filters: as a message is passed through the line from left to right, it passes first through coarse filters and then through more granular ones.

Continuing our one-line example, suppose we still want important mail and uucp messages to be logged to */var/log/mail*, but we'd like to exclude uucp messages with priority *alert*. Our line then looks like Example 10-3.

*Example 10-3. Multiple selectors in a single line*

```
mail,uucp.notice;uucp.!=alert     /var/log/mail
```



Actually, *syslogd*'s behavior isn't as predictable as this may imply: listing selectors that contradict each other or that go from specific to general rather than vice versa can yield unexpected results. Therefore, it's more accurate to say "for best results, list general selectors to the left and their exceptions (and/or more-specific selectors) to the right."

Wherever possible, keep things simple. You can use the *logger* command to test your *syslog.conf* rules (see "Testing System Logging with logger" later in this chapter).

Note that in the second selector (`uucp.!=alert`), we used the prefix `!=` before the priority to signify "not equal to." If we wanted to exclude uucp messages with priority *alert* and higher (i.e, *alert* and *emerg*), we could omit the `=` (see Example 10-4).

*Example 10-4. Selector list with a less specific exception*

```
mail,uucp.notice;uucp.!alert     /var/log/mail
```

You might wonder what will happen to a uucp message of priority *info*: this matches the second selector, so it should be logged to */var/log/mail*, right? Not based on the previous examples. Since the line's first selector matches only mail and uucp messages of priority *notice* and higher, such a message wouldn't be evaluated against the second selector.

There's nothing to stop you from having a different line for dealing with *info*-level uucp messages, though. You can even have more than one line deal with these if you like. Unlike a firewall rule base, each log message is tested against all lines in */etc/syslog.conf* and acted on as many times as it matches.

Suppose we want emergency messages broadcast to all logged-in users, as well as written to their respective application logs. We could use something like Example 10-5.

*Example 10-5. A sample syslog.conf file*

```
# Sample syslog.conf file that sorts messages by mail, kernel, and "other,"
# and broadcasts emergencies to all logged-in users

# print most sys. events to tty10 and to the xconsole pipe, and emergencies to everyone
kern.warn;*.err;authpriv.none    |/dev/xconsole
*.emerg                           *

# send mail, news (most), & kernel/firewall msgs to their respective logfiles
mail.*                           -/var/log/mail
kern.*                           -/var/log/kernel_n_firewall

# save the rest in one file
*.*;mail.none                    -/var/log/messages
```

Did you notice the - (minus) sign in front of the write-to-file actions? This tells *syslogd* not to synchronize the specified log file after writing a message that matches that line. Skipping synchronization decreases disk utilization and thus improves performance, but it also increases the chances of introducing inconsistencies, such as missing or incomplete log messages, into those files. Use the minus sign, therefore, only in lines that you expect to result in numerous or frequent file writes.

Besides performance optimization, Example 10-5 also contains some useful redundancy. Kernel warnings plus all messages of error-and-higher priority, except *authpriv* messages, are printed to the X-console window. All messages having priority of *emergency* and higher are too, in addition to being written to the screens of all logged-in users.

Furthermore, all mail messages and kernel messages are written to their respective log files. All messages of all priorities (except mail messages of any priority) are written to */var/log/messages*.

Example 10-5 was adapted from the default *syslog.conf* that SuSE 7.1 put on one of my systems. But why shouldn't such a default *syslog.conf* file be fine the way it is? Why change it at all?

Maybe you needn't, but you probably should. In most cases, default *syslog.conf* files either:

- Assign to important messages at least one action that won't effectively bring those messages to your attention (e.g., by sending messages to a TTY console on a system you only access via SSH)
- Handle at least one type of message with too much or too little redundancy to meet your needs

We'll conclude our discussion of *syslog.conf* with Tables 10-1 through 10-4, which summarize *syslog.conf*'s allowed facilities, priorities, and types of actions. Note that numeric codes *should not* be used in *syslog.conf* on Linux systems. They are provided here strictly as a reference, should you need to configure a non-Linux syslog daemon that uses numeric codes (e.g., Cisco IOS), or to send syslog messages to your log server because they're used internally (i.e., in raw syslog packets). You may see them referred to elsewhere.

*Table 10-1. syslog.conf's allowed facilities*

| Facilities | Facility codes |
|---|---|
| *auth* | 4 |
| *auth-priv* | 10 |
| *cron* | 9 |
| *daemon* | 3 |
| *kern* | 0 |

*Table 10-1. syslog.conf's allowed facilities (continued)*

| Facilities | Facility codes |
|---|---|
| *lpr* | 6 |
| *mail* | 2 |
| *mark* | N/A |
| *news* | 7 |
| *syslog* | 5 |
| *user* | 1 |
| *uucp* | 8 |
| *local{0-7}* | 16-23 |
| * ("any facility") | N/A |

*Table 10-2. syslog.conf's priorities*

| Priorities (in increasing order) | Priority codes |
|---|---|
| none | N/A |
| debug | 7 |
| info | 6 |
| notice | 5 |
| warning | 4 |
| err | 3 |
| crit | 2 |
| alert | 1 |
| emerg | 0 |
| * ("any priority") | N/A |

*Table 10-3. Use of "!" and "=" as prefixes with priorities*

| Prefix | Description |
|---|---|
| *.notice (no prefix) | any event with priority of 'notice' or higher |
| *.!notice | no event with priority of 'notice' or higher |
| *.=notice | only events with priority 'notice' |
| *.!=notice | no events with priority of 'notice' |

*Table 10-4. Types of actions in syslog.conf*

| Action | Description |
|---|---|
| /some/file | Log to specified file |
| -/some/file | Log to specified file but don't sync afterwards |
| /some/pipe | Log to specified pipe |
| /dev/some/tty_or_console | Log to specified console |
| @remote.hostname.or.IP | Log to specified remote host |

*Table 10-4. Types of actions in syslog.conf (continued)*

| Action | Description |
|---|---|
| username1, username2, etc. | Log to these users' screens |
| * | Log to all users' screens |

### Running syslogd

Just as the default *syslog.conf* may or may not meet your needs, the default startup mode of *syslogd* may need tweaking as well. Table 10-5 and subsequent paragraphs touch on some *syslogd* startup flags that are particularly relevant to security. For a complete list, you should refer to the manpage *sysklogd (8)*.

In addition, note that when you're changing and testing *syslog*'s configuration and startup options, it usually makes sense to start and stop *syslogd* and *klogd* in tandem (see the "What About klogd?" sidebar at the beginning of this chapter if you don't know what *klogd* is). Since it also makes sense to start and stop these the same way your system does, I recommend that you use your system's syslog/klogd startup script.

On most Linux systems, both facilities are controlled by the same startup script, named either */etc/init.d/syslog* or */etc/init.d/sysklog* ("sysklog" is shorthand for "syslog and klogd"). See Table 10-5 for a list of some of *syslogd*'s flags.

*Table 10-5. Some useful syslogd flags*

| Flag | Description |
|---|---|
| `-m` *`minutes_btwn_marks`* | Minutes between "mark" messages (timestamp-only messages that, depending on your viewpoint, either clarify or clutter logs. A value of 0 signifies "no marks"). |
| `-a` *`/additional/socket`* | Used to specify additional sockets, besides */dev/log*, on which *syslogd* should listen for messages. |
| `-f` *`/path/to/syslog.conf`* | Used to provide the path/name of *syslog.conf*, if different than */etc/syslog.conf*. |
| `-r` | Listens for *syslog* messages from remote hosts. |

The first *syslogd* flag we'll discuss is the only one used by default in Red Hat 7.x in its */etc/init.d/syslog* script. This flag is *-m 0*, which disables *mark* messages. *mark* messages contain only a timestamp and the string `--MARK--`, which some people find useful for navigating lengthy log files. Others find them distracting and redundant, given that each message has its own timestamp anyhow.

To turn *mark* messages on, specify a positive nonzero value after *-m* that tells *syslogd* how many minutes should pass before it sends itself a mark message. Remember that *mark* has its own facility (called, predictably, "mark") and that you must specify at least one selector that matches mark messages (such as *mark.**, which matches all messages sent to the *mark* facility, or **.**, which matches all messages in all facilities).

For example, to make *syslogd* generate *mark* messages every 30 minutes and record them in */var/log/messages*, you would first add a line to */etc/syslog.conf* similar to Example 10-6.

*Example 10-6. syslog.conf selector for mark-messages*

```
mark.*                          -/var/log/messages
```

You would then need to start *syslogd*, as shown in Example 10-7.

*Example 10-7. Invoking syslogd with 30-minute marks*

```
mylinuxbox:/etc/init.d# ./syslogd -m 30
```

Another useful *syslogd* flag is *-a [socket]*. This allows you to specify one or more sockets (in addition to */dev/log* for *syslogd*) from which to accept messages.

In Chapter 6, we used this flag to allow a chrooted *named* process to bounce its messages off of a */dev/log* socket (device-file) in the chroot jail to the nonchrooted *syslogd* process. In that example, BIND was running in a "padded cell" (subset of the full filesystem) and had its own log socket, */var/named/dev/log*. We therefore changed a line in */etc/init.d/syslog* that read as shown in Example 10-8.

*Example 10-8. init.d/syslog line invoking syslogd to read messages from a chroot jail*

```
daemon syslogd -m 0 -a /var/named/dev/log
```

(Note that the "daemon" function at the beginning of this line is unique to Red Hat's init script functions; the important part here is *syslogd -m 0 -a /var/named/dev/log*.)

More than one *-a* flag may be specified (Example 10-9).

*Example 10-9. Invoking syslogd with multiple "additional log device" directives*

```
syslogd -a /var/named/dev/log -a /var/otherchroot/dev/log -a /additional/dev/log
```

Continuing down the list of flags in Table 10-5, suppose you need to test a new syslog configuration file named *syslog.conf.test*, but you prefer not to overwrite */etc/syslog.conf*, which is where syslogd looks for its configuration file by default. Use the *-f* flag to tell syslogd to use your new configuration file (Example 10-10).

*Example 10-10. Specifying the path to syslogd's configuration file*

```
mylinuxbox:/etc/init.d# ./syslogd -f ./syslog.conf.test
```

We've already covered use of the *-r* flag, which tells syslogd to accept log messages from remote hosts, but we haven't talked about the security ramifications of this. On the one hand, security is clearly enhanced when you use a centralized log server or do anything else that makes it easier for you to manage and monitor your logs.

On the other hand, you must take different threat models into account. Are your logs sensitive? If log messages traverse untrusted networks and if the inner workings of the servers that send those messages are best kept secret, then the risks may outweigh the benefit (at least, the specific benefit of syslogd's unauthenticated clear-text remote logging mechanism).

If this is the case for you, skip to this chapter's section on Syslog-ng. Syslog-ng can send remote messages via the TCP protocol and can therefore be used in conjunction with *stunnel*, *ssh*, and other tools that greatly enhance its security. Since syslog uses only the connectionless UDP protocol for remote logging and therefore can't "tunnel" its messages though *stunnel* or *ssh*, syslog is inherently less securable than Syslog-ng.

If your log messages aren't sensitive (at least the ones you send to a remote logger), then there's still the problem of Denial of Service and message forgery attacks. If you invoke *syslogd* with the *-r* flag, it will accept *all* remote messages without performing *any checks whatsoever* on the validity of the messages themselves or on their senders. Again, this risk is most effectively mitigated by using Syslog-ng.

But one tool you *can* use with syslog to partially mitigate the risk of invalid remote messages is TCPwrappers. Specifically, TCPwrappers' "hosts access" authentication mechanism provides a simple means of defining which hosts may connect and via which protocols they may connect to your log server. Hosts-access authentication is easily tricked by source-IP-spoofing (especially since syslog transactions are strictly one way), but it's better than nothing, and it's probably sufficient to prevent mischievous but lazy attackers from interfering with syslog.

If you're willing to bet that it is, obtain and install TCPwrappers and refer to its *hosts_access(5)* manpage for details. Note that despite its name, TCPwrappers' hosts access can be used to control UDP-based applications.

## Syslog-ng

As useful and ubiquitous as syslog is, it's beginning to show its age. Modern Unix and Unix-like systems are considerably more complex than they were when syslog was invented, and they have outgrown both syslog's limited facilities and its primitive network-forwarding functionality.

Syslog-ng ("syslog new generation") is an attempt to increase syslog's flexibility by adding better message filtering, better forwarding, and eventually (though not quite yet), message integrity and encryption. In addition, Syslog-ng supports remote logging over both the TCP and UDP protocols. Syslog-ng is the brainchild of and is primarily developed and maintained by Balazs ("Bazsi") Scheidler.

Lest you think Syslog-ng is untested or untrusted, it's already been incorporated into Debian GNU/Linux 2.2 "Potato" as a binary package (in the "admin" section). Syslog-ng is in fact both stable and popular. Furthermore, even though its advanced security features are still works in progress, Syslog-ng can be used in conjunction with TCP "tunneling" tools such as *stunnel* and *ssh* to authenticate or encrypt log messages sent to remote hosts.

## Compiling and Installing Syslog-ng from Source Code

The non-Debian users among you may not wish to wait for your distribution of choice to follow suit with its own binary package of Syslog-ng. Let's start, then, with a brief description of how to compile and install Syslog-ng from source.

First, you need to obtain the latest Syslog-ng source code. As of this writing, there are two concurrent branches of Syslog-ng development. Syslog-ng Version 1.4 is the stable branch, so I recommend you use the latest release of Syslog-ng 1.4.

Version 1.5 is the experimental branch, and although it's officially disclaimed as unstable, some people use it on production systems due to its new "field expansion" feature, which allows you to write messages in your own custom formats. If you decide this functionality is worth the risk of running experimental code, be sure to subscribe to the Syslog-ng mailing list (see *http://lists.balabit.hu/mailman/listinfo/syslog-ng* to subscribe).

Speaking of which, it probably behooves you to browse the archives of this mailing list periodically even if you stick to the stable branch of Syslog-ng. Bazsi Scheidler tends to prioritize bug fixes over documentation, so Syslog-ng documentation tends to be incomplete and even out of date.

But Bazsi not only maintains the mailing list, he also very actively participates in it, as do other very knowledgeable and helpful Syslog-ng users and contributors. Thus the mailing list is an excellent source of Syslog-ng assistance. Before posting a question, you may wish to see if anyone else has asked it first. See the Syslog-ng mailing list archives at *http://lists.balabit.hu/pipermail/syslog-ng/*.

Syslog-ng can be downloaded either directly from Bazsi Scheidler's web site at *http://www.balabit.hu* or from its Freshmeat project site at *http://freshmeat.net/projects/syslog-ng/*. In addition to Syslog-ng itself, you'll need the source code for libol, Syslog-ng's support library.

Unzip and untar both archives. Compile and install libol first, then Syslog-ng. For both packages the procedure is the same:

1. Change the working directory to the source's root:

   ```
   cd packagename
   ```

2. Run the source's configure script:

   ```
   ./configure
   ```

3. Build the package:

```
./make
```

4. Install the package:

```
./make install
```

This will install everything in the default locations, which for both libol and Sylog-ng are subdirectories of */usr/local* (e.g., */usr/local/lib*, */usr/local/sbin*, etc.). If you wish to install either package somewhere else—e.g., your home directory (which is not a bad place to test new software)—then in Step 2, pass that directory to *configure* with the *--prefix=* flag as in Example 10-11.

*Example 10-11. Telling configure where to install the package*

```
mylinuxbox:/usr/src/libol-0.2.23# ./configure --prefix=/your/dir/here
```

After both libol and Syslog-ng have been compiled and installed, you need to set up a few things in Syslog-ng's operating environment. First, create the directory */etc/syslog-ng*. Next, copy one or more of the example *syslog-ng.conf* files into this directory from the source-distribution's *contrib/* and *doc/* directories (unless you intend to create your *syslog-ng.conf* completely from scratch).

Finally, you need to create a startup script for *syslog-ng* in */etc/init.d* and symbolic links to it in the appropriate runlevel directories (for most Linux distributions, */etc/rc2.d*, */etc/rc3.d*, and */etc/rc5.d*). Sample *syslog-ng* init scripts for several Linux distributions are provided in the Syslog-ng source distribution's *contrib/* directory. If you don't find one there that works for you, it's a simple matter to make a copy of your old *syslog* or *sysklogd* init-script and hack it to start *syslog-ng* rather than *syslogd*.

## Running syslog-ng

It's premature to start *syslog-ng* before you've created a configuration file. However, since *syslog-ng* has so few startup flags, I'll mention them in brief and spend the remainder of this section on *syslog-ng.conf* use.

The only flags supported by the *syslog-ng* daemon are listed in Table 10-6.

*Table 10-6. syslog-ng startup flags*

| Flag | Description |
|---|---|
| `-d` | Print debugging messages |
| `-v` | Print even more debugging messages |
| `-f filename` | Use *filename* as the configuration file (default=*/etc/syslog-ng/syslog-ng.conf*) |
| `-V` | Print version number |
| `-p pidfilename` | Name process-ID-file *pidfilename* (default=*/var/run/syslog-ng.pid*) |

In normal use, set these flags in the startup script you installed or created when you installed Syslog-ng, and use that script not only automatically at startup time, but also manually if you need to restart or stop Syslog-ng afterwards.

## Configuring Syslog-ng

There's quite a bit more involved in configuring Syslog-ng than with syslog, but that's a symptom of its flexibility. Once you understand how *syslog-ng.conf* works, writing your own configurations is simple, and adapting sample configurations for your own purposes is even simpler. Its main drawback is its sketchy documentation; hopefully, what follows here will mitigate that drawback for you.

By default, Syslog-ng's configuration file is named *syslog-ng.conf* and resides in */etc/syslog-ng/*. Let's dissect a simple example of one in Example 10-12.

*Example 10-12. A simple syslog-ng.conf file*

```
# Simple syslog-ng.conf file.

options {
    use_fqdn(no);
    sync(0);
    };

source s_sys { unix-stream("/dev/log"); internal(); };
source s_net { udp(); };

destination d_security { file("/var/log/security"); };
destination d_messages { file("/var/log/messages"); };
destination d_console { usertty("root"); };

filter f_authpriv { facility(auth, authpriv); };
filter f_messages { level(info .. emerg)
        and not facility(auth, authpriv); };
filter f_emergency { level(emerg); };

log { source(s_sys); filter(f_authpriv); destination(d_security); };
log { source(s_sys); filter(f_messages); destination(d_messages); };
log { source(s_sys); filter(f_emergency); destination(d_console); };
```

As you can see, a *syslog-ng.conf* file consists of *options{}*, *source{}*, *destination{}*, *filter{}*, and *log{}* statements. Each of these statements may contain additional settings, usually delimited by semicolons.

Syntactically, *syslog-ng.conf* is very similar to C and other structured programming languages. Statements are terminated by semicolons; whitespace is ignored and may therefore be used to enhance readability (e.g., by breaking up and indenting lengthy statements across several lines).

After defining global options, message sources, message destinations, and message filters, combine them to create logging rules.

### Global options

Global options are set in *syslog-ng.conf*'s *options{}* section. Some options may be used in the *options{}* section and in one or more other sections. Predictably, options set within *source{}*, *destination{}*, *filter{}*, and *log{}* sections overrule those set in *options{}*. Table 10-7 lists some of the most useful of Syslog-ng's options.

*Table 10-7. Syslog-ng options*

| Option | Description |
|---|---|
| `chain_hostnames( yes \| no )` | After printing the hostname provided by tcp/udp message's sender, show names of all hosts by which a tcp or udp message has been handled (default=yes). |
| `skeep_hostname( yes \| no )` | Trust hostname provided by tcp/udp message's sender (default=no). |
| `suse_fqdn( yes \| no )` | Record full name of tcp/udp message-sender (default=no). |
| `suse_dns( yes \| no )` | Resolve IP address of tcp/udp message-sender (default=yes). |
| `suse_time_recvd( yes \| no )` | Set message's timestamp equal to time message was received, not time contained in message (default=no). |
| `stime_reopen( NUMBER )` | Number of seconds after a tcp connection dies before reconnecting (default=60). |
| `stime_reap( NUMBER )` | Number of seconds to wait before closing an inactive file (i.e., an open log file to which no messages have been written for the specified length of time) (default=60). |
| `slog_fifo_size( NUMBER )`[a] | Number of messages to queue in memory before processing if *syslog-ng* is busy; note that when queue is full, new messages will be dropped, but the larger the fifo size, the greater *syslog-ng*'s RAM footprint (default=100). |
| `ssync( NUMBER )`[a] | Number of lines (messages) written to a log file before file is synchronized (default=0). |
| `sowner( string )`[a] | Owner of log files *syslog-ng* creates (default=root). |
| `sgroup( string )`[a] | Group for log files *syslog-ng* creates (default=root). |
| `sperm( NUMBER )`[a] | File-permissions for log files *syslog-ng* creates (default=0600). |
| `screate_dirs( yes \| no )`[a] | Whether to create directories specified in destination-file paths if they don't exist (default=no). |
| `sdir_owner( string )`[a] | Owner of directories *syslog-ng* creates (default=root). |
| `sdir_group( string )`[a] | Group for directories *syslog-ng* creates (default=root). |
| `sdir_perm( NUMBER )`[a] | Directory permissions for directories *syslog-ng* creates (default=0700). |

[a] These options may also be used in *file()* declarations within *destination{}* statements.

Options that deal with hostnames and their resolution (*chain_hostnames()*, *keep_hostname()*, *use_fqdn()*, and *use_dns*) deal specifically with the hostnames of remote log clients and not with hostnames/IPs referenced in the body of the message.

In other words, if *syslog-ng.conf* on a central log server contains this statement:

```
options { use_dns(yes); };
```

and the remote host *joe-bob*, whose IP address is 10.9.8.7, sends this message:

```
Sep 13 19:56:56 s_sys@10.9.8.7 sshd[13037]: Accepted publickey for ROOT from
10.9.8.254 port 1355 ssh2
```

then the log server will log:

```
Sep 13 19:56:56 s_sys@joebob sshd[13037]: Accepted publickey for ROOT from
10.9.8.254 port 1355 ssh2
```

As you can see, *10.9.8.7* was resolved to *joebob*, but *10.9.8.254* wasn't looked up. (For now you can disregard the `s_sys@` in front of the hostname; I'll explain that shortly.) The *use_dns(yes)* statement applies only to the hostname at the beginning of the message indicating which host sent it; it doesn't apply to other IP addresses that may occur later in the message.

Note also that options related to files and directories may be specified both in the global *options{}* statement and as modifiers to *file()* definitions within *destination{}* statements. *file()* options, when different from their global counterparts, override them. This allows you to create a "rule of thumb" with specific exceptions.

The *chain_hostname()* and *keep_hostname()* options are also worth mentioning. By default, *keep_hostname()* is set to `no`, meaning that *syslog-ng* will not take the hostname supplied by a remote log server at face value; *syslog-ng* will instead resolve the source IPs of packets from that host to determine for itself what that host's name is. This is in contrast to syslog, which takes remote hosts' names at face value.

*chain_hostname()* determines whether *syslog-ng* should list all hosts through which each message has been relayed. By default, this option is set to `yes`.

Example 10-13 illustrates the effects of *keep_hostname(no)* and *chain_hostname(yes)* (i.e., syslog-ng's default behavior). It shows a log message (in this case, a *syslog-ng* startup notification) being generated locally and then relayed twice. *host1*, who gives its hostname as "linux," generates the message and then sends it to *host2*. *host2* records both "linux" *and* "host1," having double checked that hostname itself via DNS. Finally, the message is relayed to *host3*.

*Example 10-13. A log message relayed from one host to two others*

**Original log entry on host1:**

```
Sep 19 22:57:16 s_loc@linux syslog-ng[1656]: syslog-ng version 1.4.13 starting
```

**Entry as sent to and recorded by host2:**

```
Sep 19 22:57:16 s_loc@linux/host1 syslog-ng[1656]: syslog-ng version 1.4.13 starting
```

*Example 10-13. A log message relayed from one host to two others (continued)*

```
Same log entry as relayed from host2 to host3:
Sep 19 22:57:16 s_loc@linux/host1/host2 syslog-ng[1656]: syslog-ng version 1.4.13 starting
```

There are several interesting things to note in this example. First, you can see that in the second entry (the one logged by *host2*), Syslog-ng does *not* clearly indicate that "linux" is actually *host1*—it simply adds the "real" hostname after the "fake" one in the slash-delimited hostname chain.

Second, the timestamp is *identical* in all three log entries. It's unlikely that three hosts would be in sync to the millisecond *and* be able to relay log messages amongst themselves virtually instantaneously. In fact, the timestamp given to the message by the originating host (*host1* here) is preserved on each host to which the message is relayed, unless a host has its own *use_time_recd()* option set to "yes" (which causes *syslog-ng* to replace message-provided timestamps with the time at which the message was received locally).

Finally, Example 10-13 also shows that when *host1* created the message, it (actually its local *syslog-ng* process) appended *s_loc*, to the message—this is the label of the *source{}* on *host1* from which the local *syslog-ng* process received the message. Example 10-14 lists *host1*'s *syslog-ng.conf* file, the one responsible for the first entry shown in Example 10-13.

*Example 10-14. host1's syslog-ng.conf file*

```
options { };
source s_loc { unix-stream("/dev/log"); internal(); };
destination d_host2 { udp("host2" port(514)); };
destination d_local { file("/var/log/messages"); };
log { source(s_loc); source(s_net); destination(d_host2); destination(d_local); };
```

Which brings us to the next topic: Syslog-ng message sources.

### Sources

The *syslog-ng.conf* file listed in Example 10-14 contains one *source{}* definition, which itself contains two source "drivers" (message-inputs). *syslog-ng.conf* may contain many *source{}* definitions, each of which may, in turn, contain multiple drivers. In other words, the syntax of source definitions is as follows:

```
source sourcelabel { driver1( [options] ); driver2( [options] ); etc. };
```

where *sourcelabel* is an arbitrary string used to identify this group of inputs, and where *driver1()*, *driver2()*, etc. are one or more source drivers that you wish to treat as a single group.

Let's take a closer look at the source definition in Example 10-14:

```
source s_loc { unix-stream("/dev/log"); internal(); };
```

This line creates a source called *s_loc* that refers to messages obtained from */dev/log* (i.e., the local system-log socket) and from the local *syslog-ng* process.

Syslog-ng is quite flexible in the variety of source drivers from which it can accept messages. In addition to Unix sockets (e.g., */dev/log*), *syslog-ng* itself, and UDP streams from remote hosts, Syslog-ng can accept messages from named pipes, TCP connections from remote hosts, and special files (e.g., */proc* files). Table 10-8 lists Syslog-ng's supported source drivers.

*Table 10-8. Source drivers for Syslog-ng*

| Source | Description |
|---|---|
| `internal()` | Messages from the *syslog-ng* daemon itself. |
| `file("filename" [options])` | Messages read from a special file such as */proc/kmsg*. |
| `pipe("filename" )` | Messages received from a named pipe. |
| `unix_stream("filename" [options])` | Messages received from Unix sockets that can be read from in the connection-oriented stream mode—e.g., */dev/log* under kernels prior to 2.4; the maximum allowed number of concurrent stream connections may be specified (default=100). |
| `unix_dgram("filename" [options])` | Messages received from Unix sockets that can be read from in the connectionless datagram mode—e.g. *klogd* messages from */dev/log* under kernel 2.4.x. |
| `tcp([ip(address)] [port(#)] [max-connections(#)] )` | Messages received from remote hosts via the tcp protocol on the specified TCP port (default=514) on the specified local network interface (default=all); the maximum number of concurrent TCP connections may be specified (default=10). |
| `udp([ip(address)] [port(#)] )` | Messages received from remote hosts via the udp protocol on the specified UDP port (default=514) on the specified local network interface (default=all). |

As we just saw in Example 10-14, *internal()* is *syslog-ng* itself: *syslog-ng* sends itself startup messages, errors, and other messages via this source. Therefore, you should include *internal()* in at least one *source{}* definition. *file()* is used to specify special files from which *syslog-ng* should retrieve messages. The special file you'd most likely want *syslog-ng* to read messages from is */proc/kmsg*.

Note, however, that *file()* is *not* intended for use on regular text files. If you wish *syslog-ng* to "tail" dynamic log files written by other applications (e.g., *httpd*), you'll need to write a script that pipes the output from a *tail -f [filename]* command to *logger*. (For instructions on using *logger*, see the section "Testing System Logging with logger" later in this chapter.)

*unix_stream()* and *unix_dgram()* are important drivers: these read messages from connection-oriented and connectionless Unix sockets, respectively. As noted at the end of "Compiling and Installing Syslog-ng from Source Code," Linux kernels Versions 2.4.1 and higher use Unix datagram sockets; if you specify */dev/log* as a *unix_stream()* source, kernel messages won't be captured. Therefore, use *unix_dgram()* when defining your local-system log source, e.g.:

```
source s_loc { unix-dgram("/dev/log"); internal(); };
```

If your kernel is pre-2.4.0, you should instead use *unix_stream()* for */dev/log*.

*tcp()* and *udp()* read messages from remote hosts via the connection-oriented TCP protocol and the connectionless UDP protocol, respectively. In both *tcp()* and *udp()*, a listening address and a port number may be specified. By default, *syslog-ng* listens on 0.0.0.0:514—that is, "all interfaces, port 514." (Specifically, the default for *tcp()* is 0.0.0.0:TCP514, and for *udp(),* that is 0.0.0.0:UDP514.)

Example 10-15 shows source statements for *tcp()* and *udp()*, with IP and port options defined.

*Example 10-15. tcp() and udp() sources*

```
source s_tcpmessages { tcp( ip(192.168.190.190) port(10514) ); };
source s_udpmessages { udp(); };
```

In Example 10-15, we're defining the source *s_tcpmessages* as all messages received on TCP port 10514, but only on the local network interface whose IP address is 192.168.190.190. The source *s_udpmessages*, however, accepts all UDP messages received on UDP port 514 on all local network interfaces.

Besides *ip()* and *port()*, there's one more source option I'd like to cover. *max_connections()*, which can only be used in *tcp()* and *unix_stream()* sources, restricts the number of simultaneous connections from a given source that *syslog-ng* will accept. This is a tradeoff between security and performance: if this number is high, then few messages will be dropped when the server is under load, but at the expense of resources. If this number is low, the chance that logging activity will bog down the server is minimized, but whenever the number of maximum connections is reached, messages will be dropped until a connection is freed up.

The correct syntax for *max-connections()* is simple: specify a positive integer between the parentheses. For example, let's adapt the *tcp()* source from Example 10-15 to accept a maximum of 100 concurrent TCP connections from remote hosts:

```
source s_tcpmessages { tcp( ip(192.168.190.190) port(10514) max-connections(100) );
};
```

By default, *max-connections()* is set to 100 for *unix-stream()* sources and 10 for *tcp()* sources.

By the way, TCP port 514 is the default listening port not only for *syslog-ng*, but also for *rshd*. This isn't a big deal, for the simple reason that *rshd* has no business running on an ostensibly secure Internet-accessible system. If, for example, you wish to use both *syslog-ng* and *rshd* on an intranet server (even then I recommend *sshd* instead), then you should specify a different (unused) port for *syslog-ng* to accept TCP connections on.

### Destinations

*syslog-ng* can be configured to send messages to the same places syslog can: ASCII files, named pipes, remote hosts via UDP, and TTYs. In addition, *syslog-ng* can send

messages to Unix sockets, remote hosts via TCP, and to the standard inputs of programs. Table 10-9 lists the allowed destination types (called "drivers") in Syslog-ng.

*Table 10-9. Supported destination drivers in syslog-ng.conf*

| Driver | Description |
|---|---|
| `file("filename[$MACROS]" )` | Write messages to standard ASCII-text log file. If file doesn't exist, *syslog-ng* will create it. Macros may be used within or in lieu of a filename; these allow dynamic naming of files (see Table 10-10). |
| `tcp("address" [port(#);] )` | Transmit messages via TCP to the specified TCP port (default=514) on the specified IP address or hostname. (You must specify an address or name.) |
| `udp("address" [port(#);] )` | Transmit messages via UDP to the specified UDP port (default=514) on the specified IP address or hostname. (You must specify an address or name.) |
| `pipe("pipename")` | Send messages to a named pipe such as */dev/xconsole*. |
| `unix_stream("filename" [options])` | Send messages in connection-oriented stream mode to a Unix socket such as */dev/log*. |
| `unix_dgram("filename" [options])` | Send messages in connectionless datagram mode to a Unix socket such as */dev/log*. |
| `usertty( username )` | Send messages to specified user's console. |
| `program("/path/to/program")` | Send messages to standard input of specified program with specified options. |

As with ordinary syslog, the most important type of destination is *file()*. Unlike with syslog, Syslog-ng supports filename-expansion macros and a number of options that give one much more granular control over how log files are handled.

When you specify the name of a file for *syslog-ng* to write messages to, you may use macros to create all or part of the filename. For example, to tell *syslog-ng* to write messages to a file whose name includes the current day, you could define a destination like this:

```
destination d_dailylog { file("/var/log/messages.$WEEKDAY"); };
```

When Syslog-ng writes to this particular destination, it will use the filename */var/log/messages.Tues*, */var/log/messages.Wed*, etc., depending on what day it is. See Table 10-10 for a complete list of supported filename macros.

*Table 10-10. Macros supported in file() destinations*

| Macro | Expands to |
|---|---|
| `PROGRAM` | The name of the program that sent the message |
| `HOST` | The name of the host that originated the message |
| `FACILITY` | The facility to which the message was logged |

*Table 10-10. Macros supported in file() destinations (continued)*

| Macro | Expands to |
|---|---|
| PRIORITY *or* LEVEL *(synonyms)* | The designated priority level |
| YEAR | The current year[a] |
| MONTH | The current month[a] |
| DAY | The current day[a] |
| WEEKDAY | The current day's name (Monday, etc.)[a] |
| HOUR | The current hour[a] |
| MIN | The current minute[a] |
| SEC | The current second[a] |

[a] If the global option *use_time_recvd()* is set to yes, then this macro's value will be taken from the local system time when the message was received; otherwise, for messages from remote hosts, the timestamp contained in the message will be used.

As with syslog, if a file specified in a *file()* destination doesn't exist, *syslog-ng* will create it. Unlike syslog, Syslog-ng has a number of options that can be implemented both globally and on a per-log-file basis. (Global settings are overridden by per-log-file settings, allowing you to create "general rules" with exceptions.)

For example, whether and how *syslog-ng* creates new directories for its log files is controlled via the options *create_dirs()*, *dir_owner()*, *dir_group()*, and *dir_perm()*. Example 10-16 illustrates the use of these options within a *destination{}* statement.

*Example 10-16. Controlling a file() destination's directory-creating behavior*

```
destination d_mylog { file("/var/log/ngfiles/mylog" create_dirs(yes) dir_owner(root) \
dir_group(root) dir_perm(0700)); };
```

Example 10-16 also happens to show the default values of the *dir_owner*, *dir_group()*, and *dir_perm()* options. While this may seem unrealistic (why would anyone go to the trouble of setting an option to its default?), it's necessary if nondefaults are specified in a global *options{}* statement and you want the default values used for a specific file—remember, options set in a *destination{}* statement override those set in an *options{}* statement.

Other global/file-specific options can be used to set characteristics of the log file itself: *owner()*, *group()*, and *perm()*, which by default are set to *root*, *root*, and *0600*, respectively. In case you're wondering, there is no *create_file()* option—*syslog-ng* has the irrevocable ability to create files (unless that file's path includes a nonexistent directory and *create_dirs()* is set to no). Example 10-17 shows a destination definition that includes these options.

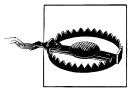
*Example 10-17. Options that affect file properties*

```
destination d_micklog { file("/var/log/micklog" owner(mick) group(wheel) perm(0640)); };
```

The other *file()* option we'll cover here is *sync()*, which can be used to limit the frequency with which log files are synchronized. This is analogous to syslog's "-" prefix, but much more granular: whereas the "-" merely turns off synchronization, *file()* accepts a numeric value that delays synchronization to as many or as few messages as you like.

The higher the value, the more messages are cached prior to filesystem synchronization and, therefore, the fewer "open for read" actions on the filesystem. The lower the number, the lower the chances of data loss and the lower the delay between a message being processed and written to disk.

By default, *sync()* is set to zero, meaning "synchronize after each message." In general, the default or a low *sync()* value is preferable for low-volume scenarios, but numbers in the 100s or even 1,000s may be necessary in high-volume situations. A good rule of thumb is to set this value to the approximate number of log-message lines per second your system must handle at peak loads.



If you use a log monitor such as Swatch (described later in this chapter) to be alerted of attacks in progress, don't set *sync()* too high. If an intruder deletes a log file, all of Syslog-ng's cached messages will be lost without having been parsed by the log monitor. (Log monitors parse messages as they are written, not beforehand.)

### Filters

And now we come to some of the serious magic in Syslog-ng: message filters. Filters, while strictly optional, allow you to route messages based not only on priority/level and facility (which syslog can do), but also on the name of the program that sent the message, the name of the host that forwarded it over the network, a regular expression evaluated against the message itself, or even the name of another filter.

A *filter{}* statement consists of a label (the filter's name) and one or more criteria connected by operators (*and*, *or*, and *not* are supported). Table 10-11 lists the different types of criteria that a *filter{}* statement may contain.

*Table 10-11. filter{} functions*

| Function (criterion) | Description |
|---|---|
| `facility( facility-name )` | Facility to which the message was logged (see Table 10-1 for facility names). |
| `priority( priority-name )`<br>`priority( priority-name1, priority-name2, etc. )`<br>`priority( priority-name1 .. priority-name2 )` | Priority assigned to the message (see Table 10-2 for priority-names); a list of priorities separated by commas may be specified, or a range of priorities expressed as two priorities (upper and lower limits) separated by two periods. |
| `level( priority-name )` | Same as *priority()*. |
| `program( program-name )` | Program that created the message. |
| `host( hostname )` | Host from which message was received. |

*Table 10-11. filter{} functions (continued)*

| Function (criterion) | Description |
|---|---|
| `match( regular-expression )` | Regular expression to evaluate against the message's body. |
| `filter( filter-name )` | Other filter to evaluate. |

Example 10-18 shows several *filter{}* statements taken from the default *syslog-ng.conf* file included in Debian 2.2's syslog-ng package.

*Example 10-18. Filters*

```
filter f_mail { facility(mail); };
filter f_debug { not facility(auth, authpriv, news, mail); };
filter f_messages { level(info .. warn) and not facility(auth, authpriv, cron, daemon,
mail, news); };
filter f_cother { level(debug, info, notice, warn) or facility(daemon, mail); };
```

The first line in Example 10-17, `filter f_mail`, matches all messages logged to the *mail* facility. The second filter, *f_debug*, matches all messages not logged to the *auth*, *authpriv*, *news*, and *mail* facilities.

The third filter, *f_messages*, matches messages of priority levels *info* through *warn*, except those logged to the *auth*, *authpriv*, *cron*, *daemon*, *mail*, and *news* facilities. The last filter, called *f_cother*, matches all messages of priority levels *debug*, *info*, *notice*, and *warn*, and also all messages logged to the *daemon* and *mail* facilities.

When you create your own filters, be sure to test them using the *logger* command. See the section entitled "Testing System Logging with logger" later in this chapter.

### Log statements

Now we combine the elements we've just defined (sources, filters, and destinations) into *log{}* statements. Arguably, these are the simplest statements in *syslog-ng.conf*: each consists only of a semicolon-delimited list of *source()*, *destination()*, and, optionally, *filter()* references. (Filters are optional because a *log{}* statement containing only *source()* and *destination()* references will send all messages from the specified sources to all specified destinations.)

Elements from several previous examples are combined in Example 10-19, which culminates in several *log{}* statements.

*Example 10-19. Another sample syslog-ng.conf file*

```
source s_loc { unix-stream("/dev/log"); internal(); };
source s_tcpmessages { tcp( ip(192.168.190.190); port(10514);); };

destination d_dailylog { file("/var/log/messages.$WEEKDAY"); };
destination d_micklog { file("/var/log/micklog" owner(mick) perm(0600)); };

filter f_mail { facility(mail); };
```

*Example 10-19. Another sample syslog-ng.conf file (continued)*

```
filter f_messages { level(info .. warn) and not facility(auth, authpriv, cron, daemon,
mail, news); };

log { source(s_tcpmessages); destination(d_micklog); };
log { source(s_loc); filter(f_mail); destination(d_micklog); };
log { source(s_loc); filter(f_messages); destination(d_dailylog); };
```

As you can see in this example, all messages from the host 192.168.190.190 are written to the log file */var/log/micklog*, as are all local mail messages. Messages that match the *f_messages()* filter are written to the log file */var/log/messages.$WEEKDAY*, e.g., */var/log/Sun*, */var/log/Mon*, etc.

Example 10-19 isn't very realistic, though: no nonmail messages with priority-level higher than *warn* are dealt with. This begs the question, "Can I get *syslog-ng* to filter on 'none of the above?'" The answer is yes: to match all messages that haven't yet matched filters in previous *log{}* statements, you can use the built-in filter *DEFAULT*. The following line, if added to the bottom of Example 10-18, will cause all messages not processed by any of the prior three *log{}* statements to be written to the daily log file:

```
log { source(s_loc); filter(DEFAULT); destination(d_dailylog); };
```

## Advanced Configurations

As you're hopefully convinced of by this point, Syslog-ng is extremely flexible, so much so that it isn't feasible to illustrate all possible Syslog-ng configurations. I would be remiss, however, if I didn't list at least one advanced *syslog-ng.conf* file.

Example 10-20 shows a setup that causes *syslog-ng* to watch out for login failures and access denials by matching messages against a regular expression and then sending the messages to a shell script (listed in Example 10-21).

*Example 10-20. Using syslog-ng as its own log watcher*

```
# WARNING: while this syslog-ng.conf file is syntactically correct and complete, it is
# intended for illustrative purposes only -- entire categories of message
# are ignored!

source s_local { unix_stream("dev/log"); internal(); };
filter f_denials { match("[Dd]enied|[Ff]ail"); };
destination d_mailtomick { program("/usr/local/sbin/mailtomick.sh"); };
log { source(s_local); filter(f_denials); destination(d_mailtomick); };
```

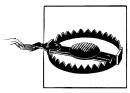
*Example 10-21. Script for emailing log messages*

```
#!/bin/bash
# mailtomick.sh
# Script which listens for standard input and emails each line to mick
#
```

*Example 10-21. Script for emailing log messages (continued)*

```
while read line;
do
echo $line | mail -s "Weirdness on that Linux box" mick@pinheads-on-ice.net
done
```

The most important lines in Example 10-20 are the filter *f_denials* and the destination *d_mailtomick*. The filter uses a *match()* directive containing a regular expression that matches the strings "denied," "Denied," "Fail," and "fail."* The destination *d_mailtomick* sends messages via a *program()* declaration to the standard input of a script I wrote called */usr/local/sbin/mailtomick.sh*.



Before we go further in the analysis, here's an important caveat: *program()* opens the specified program once and leaves it open until syslog-ng is stopped or restarted. Keep this in mind when deciding whether to use *pipe()* or *program()* (i.e., *pipe()* doesn't do this), and in choosing what sort of applications you invoke with *program()*.

In some cases, keeping a script open (actually a *bash* process) is a waste of resources and even a security risk (if you run *syslog-ng* as *root*). Furthermore, the particular use of email in Examples 10-19 and 10-20 introduces the possibility of Denial of Service attacks (e.g., filling up the system administrator's mailbox). But under the right circumstances, such as on a non-Internet-accessible host that has a few CPU cycles to spare, this is a legitimate use of Syslog-ng.

The script itself, */usr/local/sbin/mailtomick.sh*, simply reads lines from the standard input and emails each line to *mick@pinheads-on-ice.net*. Since *syslog-ng* needs to keep this script open, the *read* command is contained in an endless loop. This script will run until the *syslog-ng* process that invoked it is restarted or killed.

In the interest of focusing on the most typical uses of Syslog-ng, I've listed some *syslog-ng.conf* options without giving examples of their usage and omitted a couple of other options altogether. Suffice it to say that the global/file option *log_fifo_size()* and the global options *time_reap()*, *time_reopen()*, *gc_idle_threshold()*, and *gc_busy_threshold()* are useful for tuning *syslog-ng*'s performance to fit your particular environment.



The official (maintained) documentation for Syslog-ng is the *Syslog-ng Reference Manual*. PostScript, SGML, HTML, and ASCII text versions of this document are included in the */doc* directory of Syslog-ng's source-code distribution.

For advanced or otherwise unaddressed issues, the best source of Syslog-ng information is the Syslog-ng mailing list and its archives. See *http://lists.balabit.hu/mailman/listinfo/syslog-ng* for subscription information and archives.

* If you're completely new to regular expressions, I highly recommend *Mastering Regular Expressions* by Jeffrey E. F. Friedl (O'Reilly).

## Testing System Logging with logger

Before we leave the topic of system-logger configuration and use, we should cover a tool that can be used to test your new configurations, regardless of whether you use syslog or Syslog-ng: *logger*. *logger* is a command-line application that sends messages to the system logger. In addition to being a good diagnostic tool, *logger* is especially useful for adding logging functionality to shell scripts.

The usage we're interested in here, of course, is diagnostics. It's easiest to explain how to use *logger* with an example.

Suppose you've just reconfigured syslog to send all daemon messages with priority "warn" to */var/log/warnings*. To test the new *syslog.conf* file, you'd first restart *syslogd* and *klogd* and then you'd enter a command like the one in Example 10-22.

*Example 10-22. Sending a test message with logger*

```
mylinuxbox:~# logger -p daemon.warn "This is only a test."
```

As you can see, logger's syntax is simple. The *-p* parameter allows you to specify a *facility.priority* selector. Everything after this selector (and any other parameters or flags) is taken to be the message.

Because I'm a fast typist, I often use *while…do…done* statements in interactive *bash* sessions to run impromptu scripts (actually, just complex command lines). Example 10-23's sequence of commands works interactively or as a script.

*Example 10-23. Generating test messages from a bash prompt*

```
mylinuxbox:~# for i in {debug,info,notice,warning,err,crit,alert,emerg}
> do
> logger -p daemon.$i "Test daemon message, level $I"
> done
```

This sends tests messages to the daemon facility for each of all eight priorities.

Example 10-24, presented in the form of an actual script, generates messages for *all* facilities at each priority level.

*Example 10-24. Generating even more test messages with a bash script*

```
#!/bin/bash
for i in {auth,auth-priv,cron,daemon,kern,lpr,mail,mark,news,syslog,user,uucp,local0,
local1,local2,local3,local4,local5,local6,local7}       # (this is all one line!)

do
for k in {debug,info,notice,warning,err,crit,alert,emerg}
do
logger -p $i.$k "Test daemon message, facility $i priority $k"
done
done
```

Logger works with both syslog and Syslog-ng.

# Managing System-Log Files

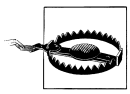
Configuring and fine-tuning your system-logging facilities is extremely important for system security and general diagnostics. But if your logs grow too large and fill up their filesystem, all that work may come to naught.

As with syslog itself, most Linux distributions come with a preconfigured log-rotation scheme. As with syslog, while this scheme tends to work adequately for many users, it's too important a mechanism to take for granted: it behooves you to understand, periodically evaluate, and, if necessary, customize your log-management setup.

## Log Management in Red Hat 7 and Debian 2.2: /sbin/logrotate

Both Red Hat 7 and Debian 2.2 use a binary program called *logrotate* to handle system-log growth. In fact, they use very similar implementations of *logrotate*: global options and low-level (system) log files are addressed in */etc/logrotate.conf*, and application-specific configuration scripts are kept in */etc/logrotate.d/*.

When *logrotate* is run, all scripts in */etc/logrotate.d* are included into *logrotate.conf* and parsed as one big script. This makes *logrotate*'s configuration very modular: when you install an RPM or DEB package (of software that creates logs), your package manager automatically installs a script in */etc/logrotate.d*, which will be removed later if you uninstall the package.



Actually, the *include* directive in *logrotate.conf* may be used to specify additional or different directories and files to include. In no event, however, should you remove the statement that includes */etc/logrotate.d* if you use Red Hat or Debian, both of whose package managers depend on this directory for package-specific log-rotation scripts.

### Syntax of logrotate.conf and its included scripts

There are really only two types of elements in *logrotate.conf* and its included scripts: directives (i.e., options) and log-file specifications. A directive is simply a parameter or a variable declaration; a log-file specification is a group of directives that apply to a specific log file or group of log files.

In Example 10-25, we see a simple */etc/logrotate.conf* file.

*Example 10-25. Simple logrotate.conf file*

```
# Very simple logrotate.conf file

# Global options: rotate logs monthly, saving four old copies and sending
# error-messages to root. After "rotating out" a file, touch a new one

monthly
rotate 4
errors root
create

# Keep an eye on /var/log/messages
/var/log/messages {
   size 200k
   create
   postrotate
      /bin/kill -HUP `cat /var/run/syslog-ng.pid 2> /dev/null` 2> /dev/null || true
   endscript
}
```

In Example 10-25, the global options at the top may be thought of as the default log-file specification. Any directive for a specific log file takes precedence over the global options. Accordingly, we see in this example that although by default logs are rotated once a month and that four archives will be kept, the file */var/log/messages* will be rotated not on the basis of time, but on size.

However, the other global directives will still apply: four old copies will be kept; immediately after a log is renamed (which is how they're "rotated"), a newly empty current log file will be created ("touched"); and error messages will be emailed to *root*.

*logrotate* supports a large number of different directives, but in practice, you'll probably spend more time tweaking the subscripts placed in *logrotate.d* than you will writing scripts from scratch. With that in mind, Table 10-12 lists some commonly encountered *logrotate* directives. A complete list is provided in the manpage *logrotate(8)*.

*Table 10-12. Common logrotate directives*

| Directive | Description |
|---|---|
| */path/to/logfile* {<br>*directive1*<br>*directive2*<br>*etc.*<br>} | Log file specification header/footer (i.e., "apply these directives to the file */path/to/logfile*"). Whitespace is ignored.<br>Applicable global directives are also applied to the log file, but when a given directive is specified both globally and locally (within a log file specification), the local setting overrules the global one. |
| rotate *number* | Tells *logrotate* to retain *number* old versions of the specified log file. Setting this to zero amounts to telling *logrotate* to overwrite the old log file. |

*Table 10-12. Common logrotate directives (continued)*

| Directive | Description |
|---|---|
| daily \| weekly \| monthly \| size=*number_bytes* | The criterion for rotating the specified file: either because one day or week or month has passed since the last rotation, or because the file's size has reached or exceeded *number_bytes* since the last time `logrotate` was run.<br>Note that if *number_bytes* is a number, bytes are assumed; if expressed as a number followed by a lowercase "k," Kilobytes are assumed; if expressed as a number followed by a capital "M," Megabytes are assumed. |
| mail *[username\|mail@address]* | Email old files to the specified local user or email address rather than deleting them. |
| errors *[username\|email@address]* | Email `logrotate` error messages to the specified local user or email address. |
| `compress` | Use *gzip* to compress old versions of log files. |
| `copytruncate` | Instead of renaming the current log file and creating a new (empty) one, move most of its data out into an archive file. Accommodates programs that can't interrupt logging (i.e., that need to keep the log file open for writing continuously). |
| `create [octalmode owner group]` | Recreate the (now empty) log file immediately after rotation. If specified, set any or all of these properties: *octalmode* (file-mode in octal notation—e.g., 0700), *owner*, and *group* properties. |
| `ifempty \| notifempty` | By default, *logrotate* will rotate a file even if it's empty. *notifempty* cancels this behavior; *ifempty* restores it (e.g., overriding a global *notifempty* setting). |
| `include file_or_directory` | When parsing *logrotate.conf*, include the specified file or the files in the specified directory. |
| `missingok \| nomissingok` | By default, *logrotate* will return a message if a log file doesn't exist. *nomissingok* cancels this behavior (i.e., tells *logrotate* to skip that log file quietly); *missingok* restores the default behavior (e.g., overriding a global *nomissingok* setting). |
| `olddir dir \| noolddir` | Tells *logrotate* to keep old versions of a log file in *dir*, whereas *noolddir* tells *logrotate* to keep old versions in the same directory as the current version (*noolddir* is the default behavior). |
| `postrotate`<br>` line1`<br>` line2`<br>` etc.`<br>`endscript` | Execute specified *lines* after rotating the log file. Can't be declared globally. Typically used to send a SIGHUP to the application that uses the log file. |
| `prerotate`<br>` line1`<br>` line2`<br>` etc.`<br>`endscript` | Execute specified *lines* before rotating the log file. Can't be declared globally. |

### Just What Do We Mean By "Rotate?"

All log-management mechanisms involve periodically moving/renaming a log file to an archive copy and creating a new (empty) log file. Rotation is necessary when multiple archive copies are maintained.

In the most common log-rotation scheme, a set of static filenames is maintained. For example, *messages, messages.1, messages.2, messages.3* is a typical three-archive filename set—*messages* being the "current" log file and *messages.3* being the oldest archive.

In this scheme, rotation is achieved by coping the second-to-oldest file over the oldest file (e.g., `mv messages.2 messages.3`). The third-oldest file's name is then changed to that of the second-oldest file's, and so forth, until the current file is renamed and a new (empty) "current" log file is created (e.g., `mv messages messages.1; touch messages`). This is how *logrotate* behaves when its *rotate* parameter is set to a nonzero value.

In the second common mechanism, archive filenames are unique (e.g., *messages*, *messages.20010807*, *messages.20010708*, etc.). In this case, rotation is a simple matter of changing the current file's name and then creating a new (empty) "current" log file (e.g., `mv messages messages.20010928; touch messages`). The final step is to compare the age of the oldest log archive file to a "maximum age" setting and to delete it if it's reached that age.

This second scheme is used by SuSE's *aaa_base_rotate_logs* script (covered later in this chapter).

#### Running logrotate

In both Red Hat 7 and Debian 2.2, *logrotate* is invoked by the script */etc/cron.daily/logrotate*, which consists of a single command:

```
/usr/sbin/logrotate /etc/logrotate.conf
```

This doesn't necessarily mean that logs are rotated daily; it means that *logrotate* checks each log file daily against its configuration script and rotates or doesn't rotate the log file accordingly.

If you want *logrotate* to be run less frequently, you can move this script to */etc/cron.weekly* or even */etc/cron.monthly* (though the latter is emphatically *not* recommended unless *logrotate* is, for some strange reason, configured to rotate each and every file monthly).

## Log Management in SuSE 7

Log rotation in SuSE, as with so much else, is configured at a gross level in */etc/rc.config* (the configuration file for *suseconfig*, which is the primary backend engine of *yast*). This file contains a variable called *MAX_DAYS_FOR_LOG_FILES*, which you

can use to set the maximum number of days system logs are kept (by default, 365). In addition, the log-rotation tools themselves come preconfigured and preactivated.

Chances are, however, that you'll need to tweak SuSE's log-management setup more granularly than *MAX_DAYS_FOR_LOG_FILES*, especially if you install Syslog-ng and disable syslog. As it happens, SuSE's log-rotation scheme is less powerful but also much simpler than Red Hat's and Debian's *logrotate*.

SuSE uses a script called */etc/cron.daily/aaa_base_rotate_logs* for day-to-day log rotation. This script shouldn't be manually edited; its behavior is controlled by the file */etc/logfiles*, which is simply a list of the files you wish to rotate along with the maximum sizes you want them to reach, the permissions and ownerships they should have, and the startup script (if any) that should be restarted after rotation is done.

Example 10-26 is an excerpt from the default */etc/logfiles* from SuSE 7.1.

*Example 10-26. Excerpts from /etc/logfiles*

```
# /etc/logfiles - This file tells cron.daily, which log files have to be watched
#
# File                        max size   mode     ownership    service
#                                                             (reload if changed)
/var/log/mgetty.*               +1024k   644      root.root
/var/log/messages               +4096k   640      root.root
/var/log/httpd/access_log       +4096k   644      root.root     apache
/var/squid/logs/access.log      +4096k   640      squid.root
```

In the first noncomment line, all log files whose name begins */var/log/mgetty* will be rotated after exceeding 1,024 kilobytes, after which they'll be rotated to new files whose permissions are `-rw-r--r--` and that are owned by user *root* and group *root*.

The third line states that the file */var/log/httpd/access_log* should be rotated after exceeding 4,096 kilobytes, should be recreated with permissions `-rw-r--r--`, owned by user *root* and group *root*, and after rotation is done, the startup script */etc/init.d/apache* should be restarted.

Since the maximum age of all log files is set globally in */etc/rc.config*, take care not to set the maximum size of a frequently written-to file (such as */var/log/messages*) too high. If this happens and if the maximum age is high enough, your logs may fill their volume.

Speaking of which, I highly recommend the use of a dedicated */var* partition on any machine that acts as a server; a full */var* partition is much less likely to cause disruptive system behavior (e.g., crashing) than a full root partition.

## Using Swatch for Automated Log Monitoring

Okay, you've painstakingly configured, tested, and fine-tuned your system logger to sort system messages by type and importance and then log them both to their respective files and to a central log server. You've also configured a log-rotation scheme that keeps as much old log data around as you think you'll need.

But who's got the time to actually *read* all those log messages?

*swatch* (the "Simple WATCHer") does. *swatch*, a free log-monitoring utility written 100% in Perl, monitors logs as they're being written and takes action when it finds something you've told it to look out for. Swatch does for logs what tripwire does for system-file integrity.

## Installing Swatch

There are two ways to install *swatch*. First, of course, is via whatever binary package of *swatch* your Linux distribution of choice provides. (I use the term loosely here; "executable package" is more precise.) The current version of Mandrake has an RPM package of *swatch*, but none of the other most popular distributions (i.e., Red Hat, SuSE, Slackware, or Debian) appear to.

This is just as well, though, since the second way to install *swatch* is quite interesting. *swatch*'s source distribution, available from *http://www.stanford.edu/~atkins/swatch*, includes a sophisticated script called *Makefile.PL* that automatically checks for all necessary Perl modules (see "Should We Let Perl Download and Install Its Own Modules?" later in this chapter) and uses Perl 5's CPAN functionality to download and install any modules you need; it then generates a *Makefile* that can be used to build swatch.

After you've installed the required modules, either automatically from *swatch*'s *Makefile.PL* script or manually (and then running perl Makefile.PL), *Makefile.PL* should return the contents of Example 10-27.

*Example 10-27. Successful Makefile.PL run*

```
 [root@barrelofun swatch-3.0.1]# perl Makefile.PL
Checking for Time::HiRes 1.12 ... ok
Checking for Date::Calc ... ok
Checking for Date::Format ... ok
Checking for File::Tail ... ok
Checking if your kit is complete...
Looks good
Writing Makefile for swatch
[root@barrelofun swatch-3.0.1]#
```

Once *Makefile.PL* has successfully created a *Makefile* for *swatch*, you can execute the following commands to build and install it:

```
make
make test
make install
make realclean
```

The *make test* command is optional but useful: it ensures that *swatch* can properly use the Perl modules we just went to the trouble of installing.

## Should We Let Perl Download and Install Its Own Modules?

The Comprehensive Perl Archive Network (CPAN) is a network of Perl software archives from around the world. Perl Version 5.6.x includes modules (CPAN and CPAN::FirstTime, among others) that allow it to fetch, verify the checksums of, and even use gcc to compile Perl modules from CPAN sites on the Internet. In-depth descriptions of CPAN and Perl's CPAN functionality are beyond this chapter's scope, but I have one hint and one warning to offer.

First, the hint. To install the module Example::Module (not a real Perl module), you enter the command:

```
perl -MCPAN -e "install Example::Module"
```

If it's the first time you've used the -MCPAN flag, the module CPAN::FirstTime will be triggered and you'll be asked to choose from various options as to how Perl should fetch and install modules from CPAN. These are well-phrased questions with reasonable defaults. But do pay attention to the output while this command executes: the module you're installing may depend on other modules and may require you to go back and execute, e.g.:

```
perl -MCPAN -e "install Example::PreRequisite"
```

before making a second attempt at installing the first module.

Now for the warning: using CPAN is neither more nor less secure than downloading and installing other software from any other Internet source. On the one hand, before being installed, each downloaded module is automatically checked against a checksum that incorporates a cryptographically strong MD5 hash. On the other hand, this hash is intended to prevent corrupt downloads from going unnoticed, not to provide security per se.

Furthermore, even assuming that a given package's checksum probably won't be replaced along with a tampered-with module (a big assumption), all this protects against is the unauthorized alteration of software after it's been uploaded to CPAN by its author. There's nothing to stop an evil registered CPAN developer (anybody may register as one) from uploading hostile code along with a valid checksum. But of course, there's nothing to stop that evil developer from posting bad stuff to SourceForge or FreshMeat, either.

Thus, if you really want to be thorough, the most secure way to install a given Perl module is to:

1. Identify/locate the module on *http://search.cpan.org*.
2. Follow the link to CPAN's page for the module.
3. Download the module *not* from CPAN, but from its developer's official web site (listed under "Author Information" in the web page referred to earlier in Step 2).
4. If available, also download any checksum or hash provided by the developer for the tarball you just downloaded.

*—continued—*

5. Use *gpg*, *md5*, etc. to verify that the tarball matches the hash.
6. Unzip and expand the tarball, e.g., `tar -xzvf groovyperlmod.tar.gz`.
7. If you're a Righteously Paranoid Kung-Fu Master or aspire to becoming one, review the source code for sloppiness and shenanigans, report your findings to the developer or the world at large, and bask in the open source community's awe and gratitude. (I'm being flippant, but open source code is truly open only when people bother to examine it!)

Follow the module's building and installing directions, usually contained in a file called *INSTALL* and generally amounting to something like:

```
perl ./Makefile.PL
make
make test
make install
```

Note that if the modules you need are being brought to your attention by swatch's `Makefile.PL` script, then to use the paranoid installation method, you'll want to write down the needed module names and kill that script (via plain old CONTROL-c) before installing the modules and rerunning swatch's *Makefile.PL*.

Before I forget, there's actually a third way to install missing Perl modules: from your Linux distribution's FTP site or CDROM. While none approach CPAN's selection, most Linux distributions have packaged versions of the most popular Perl modules. Following are the modules you need for swatch and the packages that contain them in Red Hat 7 and Debian 2.2:

- Perl ModuleRed Hat 7 RPMDebian "deb" package
- Date::Calcperl-Date-Calclibdate-calc-perl
- Time::HiResperl-Time-HiReslibtime-hires-perl
- Date::Formatperl-TimeDatelibtimedate-perl
- File::Tailperl-File-Taillibfile-tail-perl

None of this may seem terribly specific to swatch, and indeed it isn't, but it *is* important—more and more useful utilities are being released either as Perl modules or as Perl scripts that depend on Perl modules, so the chances are that swatch will not be the last Makefile.PL-based utility you install. Understanding some ramifications of all this module madness is worth the liter of ink I just spent on it, trust me.

## swatch Configuration in Brief

Since the whole point of *swatch* is to simplify our lives, configuring *swatch* itself is, well, simple. *swatch* is controlled by a single file, *$HOME/.swatchrc* by default. This file contains text patterns, in the form of regular expressions, that you want *swatch* to watch for. Each regular expression is followed by the action(s) you wish to *swatch* to take whenever it encounters that text.

For example, suppose you've got an Apache-based web server and you want to be alerted any time someone attempts a buffer-overflow attack by requesting an extremely

long filename (URL). By trying this yourself against the web server while tailing its */var/apache/error.log*, you know that Apache will log an entry that includes the string "File name too long." Suppose further that you want to be emailed every time this happens. Example 10-28 shows what you'd need to have in your *.swatchrc* file.

*Example 10-28. Simple entry in .swatchrc*

```
watchfor /File name too long/
        mail addresses=mick\@visi.com,subject=BufferOverflow_attempt
```

As you can see, the entry begins with a *watchfor* statement, followed by a regular expression. If you aren't yet proficient in the use of regular expressions, don't worry: this can be as simple as a snippet of the text you want *swatch* to look for, spelled out verbatim between two slashes.

Swatch will perform your choice of a number of actions when it matches your regular expression. In this example, we've told swatch to send email to `mick\@visi.com`, with a subject of `BufferOverflow_attempt`. Note the backslash before the `@` sign—without it, Perl will interpret the `@` sign as a special character. Note also that if you want spaces in your subject-line, each space needs to be escaped with a backslash—e.g., `subject=Buffer\ Overflow\ attempt`.

Actions besides sending email include the ones in Table 10-13.

*Table 10-13. Some actions swatch can take*

| Action (keyword) | Description |
|---|---|
| `echo=`*`normal, underscore, blue, inverse, etc.`* | Print matched line to console, with or without special text mode (default mode is "normal"). |
| `bell` *`N`* | Echo the line to console, with "beep" sounded *N* times (default = 1). |
| `exec` *`command`* | Execute the command or script *command*. |
| `pipe` *`command`* | Pipe the line to the command *command*. |
| `throttle` *`HH:MM:SS`* | Wait for *HH:MM:SS* (period of time) after a line triggers a match, before performing actions on another match of the same expression. Helps prevent Denial of Service attacks via swatch (e.g., deliberately triggering huge numbers of swatch events in a short period). |

For more details on configuring these and the other actions that swatch supports, see the *swatch(1)* manpage.



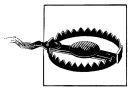
If you use Syslog-ng, you may be able to use some combination of *match()* filters, *program()* destinations, and *pipe()* destinations to achieve most of what *swatch* does.

However, *swatch*'s *throttle* parameter is an important advantage: whereas Syslog-ng acts on every message that matches a given filter, *throttle* gives *swatch* the intelligence to ignore repeated occurrences of a given event, potentially preventing minor events from becoming major annoyances.

Let's take that example a step further. Suppose in addition to being emailed about buffer-overflow attempts, you want to know whenever someone hits a certain web page, but only if you're logged on to a console at the time. In the same *.swatchrc* file, you'd add something like Example 10-29.

*Example 10-29. An event that beeps and prints to console*

```
watchfor /wuzza.html/
    echo=red
    bell 2
```



You will only see these messages and hear these beeps if you are logged on to the console in the same shell session from which you launched swatch. If you log out to go get a sandwich, when you return and log back in, you will no longer see messages generated by the swatch processes launched in your old session, even though those processes will still be running.

When in doubt, add either a "mail" action or some other non console-specific action (e.g., an "exec" action that triggers a script that pages you, etc.), unless, that is, the pattern in question isn't critical.

Alert readers have no doubt noticed that the scenario in the previous example will work only for Apache installations in which both errors and access messages are logged to the same file. We haven't associated different expressions with different watched files, nor can we. But what if you want swatch to watch more than one log file?

This is no problem. Although each *.swatchrc* file may describe only one watched file, there's nothing to stop you from running multiple instances of swatch, each with its own *.swatchrc* file. In other words, *.swatchrc* is the default, but not the required name for swatch configurations.

To split our two examples into two files, you'd put the lines in Example 10-27 into a file called, for example, *.swatchrc.hterror* and the lines in Example 10-28 into a file called *.swatchrc.htaccess*.

## Advanced swatch Configuration

So far we've only considered actions we want triggered every time a given pattern is matched. There are several ways we can control swatch's behavior with greater granularity, however.

The first and most obvious is that search patterns take the form of regular expressions. Regular expressions, which really constitute a text-formatting language of their own, are incredibly powerful and responsible for a good deal of the magic of Perl, sed, vi, and many other Unix utilities.

It behooves you to know at least a couple "regex" tricks. Trick number one is called alternation, and it adds a "logical or" to your regular expression in the form of a "|" sign. Consider this regular expression:

```
/reject|failed/
```

This expression will match any line containing either the word "reject" or the word "failed." Use alternation when you want swatch to take the same action for more than one pattern.

Trick number two is the Perl-specific regular-expression modifier "case-insensitive," also known as "slash-i" since it always follows a regular expression's trailing slash. The regular expression:

```
/reject/i
```

matches any line containing the word "reject" whether it's spelled "Reject," "REJECT," "rEjEcT," etc. Granted, this isn't nearly as useful as alternation, and in the interest of full disclosure, I'm compelled to mention that slash-i is one of the more CPU-intensive Perl modifiers. However, if despite your best efforts at log tailing, self attacking, etc., you aren't 100% sure how a worrisome attack might look in a log file, slash-i helps you make a reasonable guess.

Another way to control swatch more precisely is to specify what time of day a given action may be performed. You can do this by sticking a *when=* option after any action. For example, in Example 10-30, I have a *.swatchrc* entry for a medium-importance event, which I want to know about via console messages during weekdays, but which I'll need email messages to know about during the weekend.

*Example 10-30. Actions with when option specified*

```
/file system full/
    echo=red
    mail addresses=mick\@visi.com,subject=Volume_Full,when=7-1:1-24
```

The syntax of the *when=* option is `when=`*`range_of_days:range_of_hours`*. Thus, in Example 10-30, we see that any time the message "file system full" is logged, swatch will echo the log entry to the console in red ink. It will also send email, but only if it's Saturday ("7") or Sunday ("1").

## Running swatch

Swatch expects *.swatchrc* to live in the home directory of the user who invokes swatch. Swatch also keeps its temporary files there by default. (Each time it's invoked, it creates and runs a script called a "watcher process," whose name ends with a dot followed by the PID of the swatch process that created it).

The `-c` *`path/to/configfile`* and `--script-dir=`*`/path/to/scripts`* flags let you specify alternate locations for swatch's configuration and script files, respectively. Never

keep either in a world-writable directory, however. In fact, only these files' owners should be able to read them.

For example, to invoke swatch so that it reads my custom configuration file in */var/log* and also uses that directory for its watcher process script, I'd use the command listed in Example 10-31.

*Example 10-31. Specifying nondefault paths*

```
mylinuxbox:~# swatch -c /var/log/.swatchrc.access --script-dir=/var/log &
```

I also need to tell swatch which file to tail, and for that I need the *-t filename* flag. If I wanted to use the previous command to have swatch monitor */var/log/apache/access_log*, it would look like this:

```
mylinuxbox:~# swatch -c /var/log/.swatchrc.access --script-dir=/var/log
\  -t /var/log/apache/access_log &
```



swatch generally doesn't clean up after itself very well; it tends to leave watcher-process scripts behind. Keep an eye out and periodically delete these in your home directory or in the script directories you tend to specify with *--script-dir*.

Again, if you want swatch to monitor multiple files, you'll need to run swatch multiple times, with at least a different tailing target (*-t* value) specified each time and probably a different configuration file for each as well.

## Fine-Tuning swatch

Once swatch is configured and running, we must turn our attention to the Goldilocks Goal: we want swatch to be running neither "too hot" (alerting us about routine or trivial events) nor "too cold" (never alerting us about anything). But what constitutes "just right?" There are as many answers to this question as there are uses for Unix.

Anyhow, you don't need me to tell you what constitutes nuisance-level reporting: if it happens, you'll know it. You may even experience a scare or two in responding to events that set off alarms appropriately but turn out to be harmless nonetheless. Read the manual, tweak *.swatchrc*, and stay the course.

The other scenario, in which too little is watched for, is much harder to address, especially for the beginning system administrator. By definition, anomalous events don't happen very frequently, so how do you anticipate how they'll manifest themselves in the logs? My first bit of advice is to get in the habit of browsing your system logs often enough to get a feel for what the routine operation of your systems looks like.

Better still, "tail" the logs in real time. If you enter the command *tail -f /var/log/messages*, the last 50 lines of the system log will be printed, plus *all subsequent lines*,

*as they're generated*, until you kill tail with a *Control-c*. This works for any file, even a log file that changes very rapidly.

Another good thing you can do is to "beat up on" (probe/attack) your system in one virtual console or xterm while tailing various log files in another. nmap and Nessus, which are covered in Chapter 3 (Hardening Linux), are perfect for this.

By now you may be saying, "Hey, I thought the whole reason I installed swatch was so I wouldn't have to watch log files manually!" Wrong. Swatch *minimizes*, but does not eliminate, the need for us to parse log files.

Were you able to quit using your arithmetic skills after you got your first pocket calculator? No. For that matter, can you use a calculator in the first place unless you already know how to add, multiply, etc.? Definitely not. The same goes for log file parsing: you can't tell swatch to look for things you can't identify yourself, no more than you can ask for directions to a town whose name you've forgotten.

### Why You Shouldn't Configure swatch Once and Forget About It

In the same vein, I urge you to not be complacent about swatch silence. If swatch's actions don't fire very often, it could be that your system isn't getting probed or misused very much, but it's at least as likely that swatch isn't casting its net wide enough. Continue to periodically scan through your logs manually to see if you're missing anything, and continue to tweak *.swatchrc*.

Don't forget to periodically reconsider the auditing/logging configurations of the daemons that generate log messages in the first place. Swatch won't catch events that aren't logged at all. Refer to the *syslogd(8)* manpage for general instructions on managing your *syslogd* daemon, and the manpages of the various things that log to syslog for specific instructions on changing the way they log events.

## Resources

*http://www.stanford.edu/~atkins/swatch*. swatch home page. (Has links to the latest version, online manpages, etc.)

*http://www.stanford.edu/~atkins/swatch/lisa93.html*. Hansen, Stephen and Todd Atkins, creators of swatch. "Centralized System Monitoring with Swatch." (Old, but still useful.)

*http://www.enteract.com/~lspitz/swatch.html*. Spitzner, Lance. "Watching Your Logs." (A brief introduction to swatch.)

Friedl, Jeffrey E. F. *Mastering Regular Expressions*. Sebastopol, CA: O'Reilly & Associates, Inc. 1998.