Chapitre 4 : Démarrage et arrêt du système

Ce chapitre explique les mécanismes mis en œuvre lors du démarrage et de l'arrêt du système Debian Etc afin de pouvoir le personnaliser et/ou détecter (et corriger) les éventuels dysfonctionnements.

Les termes "démarrage", "initialisation", "boot" (abréviation de "bootstrap") et "amorçage" sont équiva lents. Ils touchent à l'ensemble des processus se déroulant entre l'allumage de la machine et l'obtentio d'un système d'exploitation fonctionnel.

A. Détail du démarrage d'une machine Linux

1. BIOS

Le BIOS (*Basic Input Output System*) est le premier programme exécuté à l'allumage de l'ordinateur. So but est de fournir des routines standard pour les différents types de matériels et de charger le systèm d'exploitation. Ainsi, quelles que soient les technologies des disques durs, ces derniers seront accessibles d la même manière.

Sans le BIOS, la machine ne pourrait pas démarrer car le microprocesseur ne sait utiliser que les instruc tions stockées en mémoire vive ; la RAM (*Random Access Memory*) étant dans un état indéfini à la mis sous tension, ce code est stocké dans une mémoire ROM (*Read Only Memory*) permanente et éventuelle ment modifiable ("flashable") par l'utilisateur.

Le BIOS aura donc pour fonction de :

- Tester les circuits et les composants matériels de la machine.
- Initialiser le système d'affichage en faisant appel au propre BIOS de la carte graphique. Ce poir permettra un affichage à l'écran des opérations suivantes mais obligera la présence d'une carte vidéc même si la machine fait office de serveur et ne dispose pas d'écran.
- Tester la mémoire.
- Détecter les périphériques de stockage (disques durs IDE et SCSI, lecteur de disquettes, lecteur d CD-Rom, périphériques USB...) et déterminer l'unité de disque de démarrage.
- Charger le système d'exploitation présent sur ce périphérique ; pour cela, lancer le gestionnair d'amorçage présent sur le premier secteur du périphérique.

Une séquence de boot peut être définie dans le BIOS. Si un gestionnaire d'amorçage n'est pas trouv sur la disquette, il est recherché sur le CD-Rom puis sur le disque dur, par exemple.

Le disque dur IDE d'amorçage est maître sur le premier contrôleur (`hda`) mais ceci est modifiable pou les BIOS les plus récents.

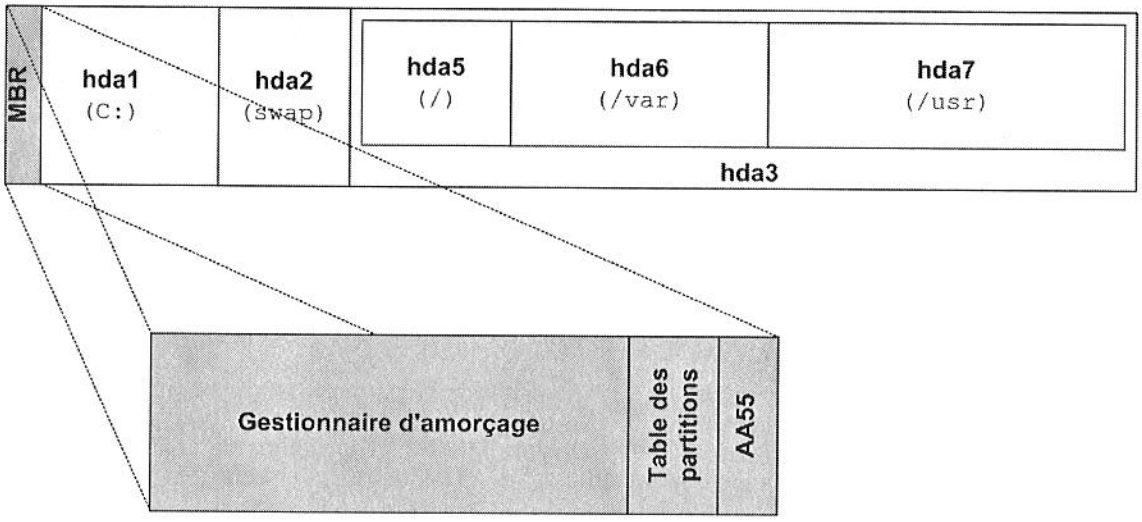
2. Gestionnaire d'amorçage

Le "gestionnaire d'amorçage", "chargeur" ou "bootloader", est le petit programme lancé sur la machin après le BIOS. Sa tâche est de lire et de mettre en mémoire l'image du noyau d'un système d'exploitatior puis de lui passer la main.

Pour être accessible et exécuté par le BIOS, celui-ci se trouve dans le premier bloc de données contenu su le périphérique d'amorçage, plus connu sous le nom de "MBR" (*Master Boot Record*).

Pour être plus exact, le format du premier bloc de données de 512 octets est le suivant :

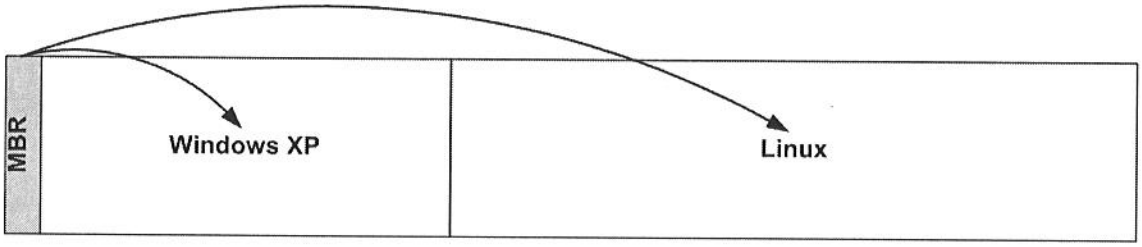
- 446 octets pour le gestionnaire d'amorçage.
- 64 octets pour la table des partitions (16 octets pour définir une partition, soit quatre partitions a maximum sans compter les partitions étendues).
- 2 octets fixés à la valeur hexadécimale AA55 par convention ; sans cela, le MBR sera considéré comm erroné.



Dans le cas le plus simple, le chargeur choisit le système d'exploitation de la partition définie comme active dans la table des partitions ; c'est pour cela qu'un système DOS ne peut être installé ailleurs.

Le bootloader peut aussi se trouver en début de partition dans le "PBR" (*Partition Boot Record*) mais il devra obligatoirement être chargé par un autre gestionnaire d'amorçage présent dans le MBR.

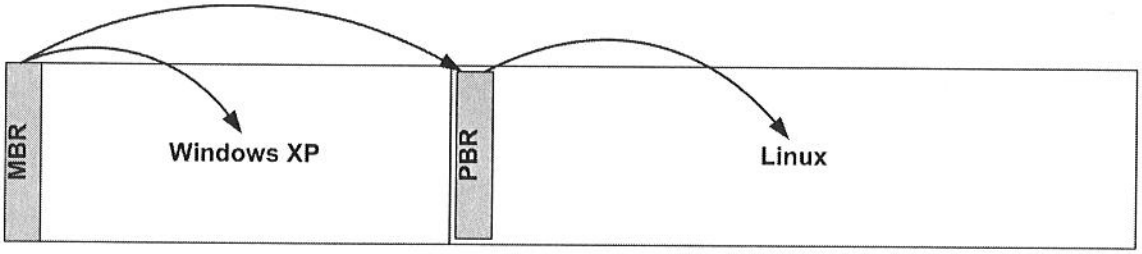
Habituellement, on placera le chargeur Linux dans le MBR, car il saura aussi charger d'autres systèmes d'exploitation cohabitant sur la même machine comme Windows XP par exemple :



Cependant, si chaque système d'exploitation est fourni avec un ou plusieurs gestionnaires d'amorçage, ces derniers ne savent pas forcément charger tous les systèmes présents sur la machine. Lors de l'installation (ou réinstallation) de l'un d'entre eux, il est possible de perdre l'accès aux autres.

Cela arrive lors de l'installation de Windows XP quand Debian Etch est déjà présent sur la machine ; le chargeur fourni ne sait charger que le système Microsoft et éventuellement, un autre chargeur se trouvant dans le PBR d'une autre partition. Il faudra donc installer au préalable le chargeur Linux dans le PBR de la partition Linux. Ceci est réalisable car tous les gestionnaires d'amorçage supportent l'opération de chaînage d'amorçage ("chainloading") qui consiste à déléguer le chargement d'un OS à un autre chargeur.

Ainsi, le chargeur Microsoft sera toujours à même de charger Windows XP mais déléguera le chargement de Linux à l'autre gestionnaire d'amorçage situé dans le PBR :



Il est possible de contourner le problème précédent et de réinscrire le chargeur Linux dans le MBR en utilisant un disque de démarrage (disquette, CD-Rom ou disque USB), contenant un secteur d'amorçage et le noyau Linux, que l'on aura eu soin de créer tant que le système Debian Etch est accessible. Dans ce cas, le disque dur n'étant plus l'unité d'amorçage, il sera possible de démarrer Debian Etch et de restaurer le chargeur Linux original supprimé lors de l'installation de Windows XP.

Il existe plusieurs gestionnaires d'amorçage dont XOSL (*eXtended Operating System Loader*), SysLinux, LoadLin, les chargeurs des systèmes BSD, ceux des différentes versions de Dos/Windows..., ainsi que LILO et GRUB, les deux chargeurs les plus utilisés dans le monde Linux.

a. Création d'un disque de démarrage

Comme indiqué précédemment, un disque de démarrage permettra de s'affranchir de tout gestionnaire d'amorçage présent sur le disque dur, celui-ci contenant son propre chargeur ainsi qu'un noyau Linux paramétré pour utiliser la bonne partition disque comme système de fichiers racine.

La commande **mkboot** du paquet *debianutils* crée ce disque. Par défaut, la commande utilise le noyau */vmlinuz* et la partition racine courante. La taille actuelle des noyaux Linux dépassant la capacité d'une disquette, un autre paquet *mkrboot* permet de créer un disque de démarrage sur disquette, sur CD-Rom ou bien de créer une image iso. Ce dernier comprend deux commandes : **mkrboot** et **mkrescue**. **mkrescue** permet de créer une image iso d'un disque de démarrage de secours comme ceci :

```
[root]# mkrescue -initrd /boot/initrd.img-2.4.27-2-386 -kernel /boot/vmlinuz- 2.4.27-2-386
-iso
```

Le noyau Linux est totalement chargé en mémoire vive dès l'amorçage du système, que ce soit à partir du disque dur ou du disque créé avec un des utilitaires Debian. Mis à part un temps de chargement éventuellement plus long dû au support d'amorçage, le système est dans les deux cas entièrement opérationnel.

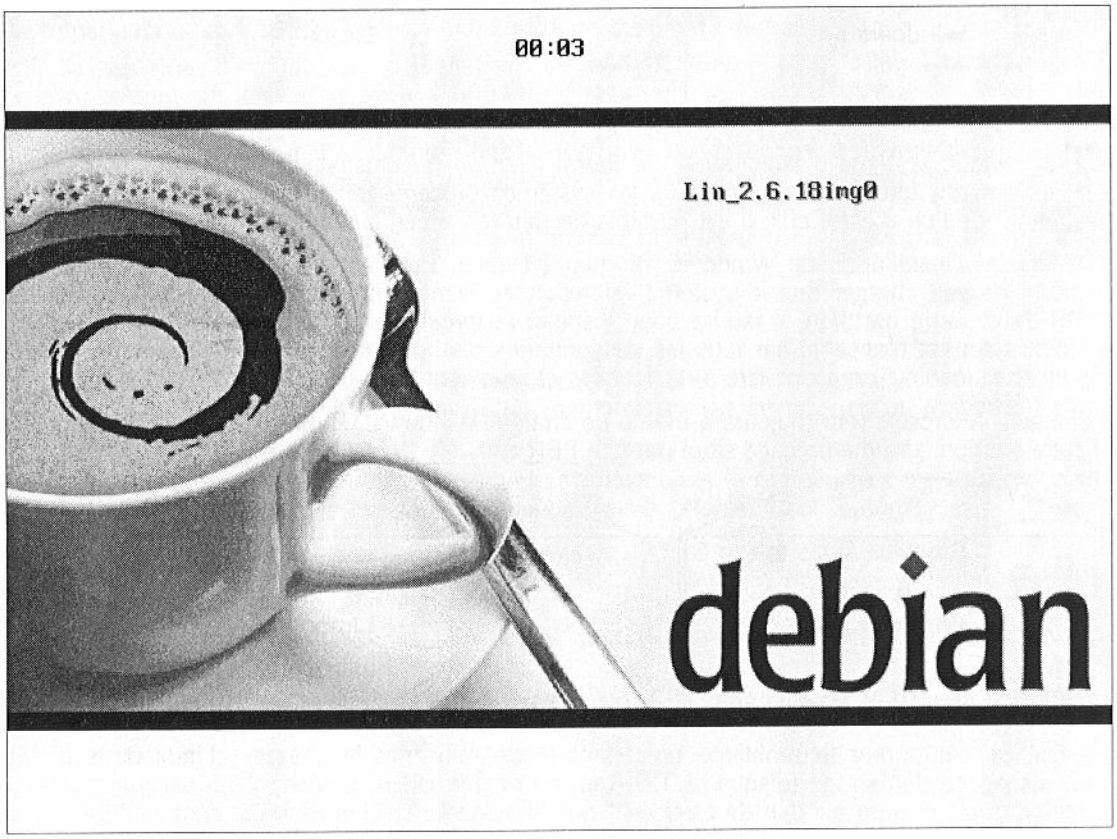
b. LILO

Le programme LILO (*LInux LOader*), écrit par Werner Almesberger, est encore couramment utilisé. Bien qu'il soit disponible sous Debian Etch, le choix par défaut est dorénavant GRUB. L'étude rapide de LILO nous permettra tout de même de mieux appréhender le fonctionnement du chargeur GRUB.

LILO peut démarrer différents systèmes dont DOS, OS/2, FreeBSD, et bien sûr Linux. Il permet en outre de passer des paramètres au noyau Linux lors de son chargement.

Utilisation

L'interface fournie par LILO est soit une invite texte au démarrage qui affiche **LILO:** ou **boot:**, soit un menu graphique dans lequel on peut naviguer avec les flèches de direction :



Une attente de quelques secondes est généralement configurée pour donner le temps à l'utilisateur de choisir le système d'exploitation à démarrer. Dans le cas de l'interface en ligne de commandes, il faudra saisir le libellé du système à charger ; la liste des systèmes disponibles étant affichée lors de l'appui sur la touche **[Tab]** du clavier (touche **[?]** avec les claviers US). Sans cela, LILO lancera le système d'exploitation par défaut.

Pour passer des paramètres à un noyau Linux lors de son chargement, il suffira de saisir ceux-ci à la suite du libellé (appuyer sur les touches **[Tab]** pour basculer du graphique au texte). Par exemple, après avoir listé les systèmes disponibles, pour changer le système de fichiers principal que doit utiliser le noyau, on saisira :

```
boot:
Linux       Old       DOS       Floppy
boot: linux root=/dev/hdb3
```

Pour intervenir sur un système en panne de démarrage, vous pouvez utiliser le CD-Rom d'installation et saisir à l'invite :

```
boot: expert
```

Le démarrage en mode expert vous permet, entre autres, de réinstaller votre chargeur de démarrage, que ce soit LILO ou bien GRUB.

Configuration

La modification du MBR (ou du PBR) est réalisée par la commande **`lilo`** qui suit les instructions laissées dans le fichier */etc/lilo.conf*. Voici un exemple de fichier de configuration (un caractère **#** en début de ligne annonce un commentaire) :

```
# This allows booting from any partition on disks with more than 1024
# cylinders.
lba32

# Specifies the boot device
boot=/dev/hda

# Specifies the device that should be mounted as root.
root=/dev/hdb1

# Bitmap configuration for /boot/debianlilo.bmp
bitmap=/boot/coffee.bmp
bmp-colors=1,,0;9,,0
bmp-table=106p,144p,2,9,144p
bmp-timer=514p,144p,6,8,0

# Install the specified file as the new boot sector.
# LILO supports built in boot sectory, you only need
# to specify the type, choose one from 'text', 'menu' or 'bitmap'.
# new: install=bmp      old: install=/boot/boot-bmp.b
# new: install=text     old: install=/boot/boot-text.b
# new: install=menu     old: install=/boot/boot-menu.b or boot.b
# default: 'menu' is default, unless you have a bitmap= line
# Note: install=bmp must be used to see the bitmap menu.
# install=menu
install=bmp

# Prompt to use certaing image. If prompt is specified without timeout,
# boot will not take place unless you hit RETURN
prompt
timeout=50

# Specifies the location of the map file. If MAP is
# omitted, a file /boot/map is used.
map=/boot/map

# Specifies the VGA text mode that should be selected when
# booting. The following values are recognized (case is ignored) :
#   NORMAL  select normal 80x25 text mode.
                                                                    .../...
```

```
.../...
#   EXTENDED  select 80x50 text mode. The word EXTENDED can be
#     abbreviated to EXT.
#   ASK  stop and ask for user input (at boot time).
#   <number>  use the corresponding text mode. A list of available modes
#     can be obtained by booting with  vga=ask  and pressing [Enter].
vga=normal

image=/boot/vmlinuz-2.6.18-4-686
   label="Linux"
   initrd=/boot/initrd.img-2.6.18-4-686
   read-only

image=/boot/vmlinuz-2.6-686
   label="Old"
   initrd=/boot/initrd.img-2.6-686
   read-only

# Alternate OS: Dos
other=/dev/hda1
   label="DOS"

# Alternate boot on floppy
other=/dev/fd0
   label="Floppy"
   # deactivate controls
   unsafe
```

Il existe encore un très grand nombre de directives pouvant être inscrites dans ce fichier, notamment po définir la géométrie des disques, changer l'interface ou encore imposer la saisie d'un mot de passe po toutes ou certaines images. Il y a par exemple :

- **linear** : crée des adresses de secteurs linéaires plutôt que des adresses 3D (secteur/tête/cylindre). Ce permet de résoudre des problèmes dus à la géométrie de certains disques.
- **lba32** : supprime la limite à 1 024 cylindres pour l'installation de */boot* si le matériel supporte cet directive.

Une fois ce fichier modifié, il ne faut pas oublier de lancer la commande **lilo** pour écrire le chargeur dai le MBR avec les directives définies précédemment. L'option **-t** permettra de tester le fichier de configur tion sans écrire réellement le chargeur. On peut aussi désigner le niveau de verbosité en ajoutant une plusieurs options **-v** à cette commande. Ainsi :

```
[root]# lilo -v -t
LILO version 22.6.1 (test mode), Copyright (C) 1992-1998 Werner Almesberger
Development beyond version 21 Copyright (C) 1999-2004 John Coffman
Released 17-Nov-2004, and compiled at 12:32:32 on May 25 2005
Debian GNU/Linux

Reading boot sector from /dev/hda
Warning: Kernel & BIOS return differing head/sector geometries for device 0x80
    Kernel: 3968 cylinders, 16 heads, 63 sectors
      BIOS: 991 cylinders, 64 heads, 63 sectors
Warning: Kernel & BIOS return differing head/sector geometries for device 0x81
    Kernel: 5952 cylinders, 16 heads, 63 sectors
      BIOS: 743 cylinders, 128 heads, 63 sectors
Using BITMAP secondary loader
Calling map_insert_data
Warning: The boot sector and map file are on different disks.
Mapping bitmap file /boot/debianlilo.bmp
Calling map_insert_file

Boot image: /boot/vmlinuz-2.6.18-4-686
Mapping RAM disk /boot/initrd.img-2.6.18-4-686
Added Linux *
                                                                    .../..
```

```
.../...
Boot image: /boot/vmlinuz-2.6-686
Mapping RAM disk /boot/initrd.img-2.6-686
Added Old

Boot other: /dev/hda1, on /dev/hda, loader CHAIN
Added DOS

Boot other: /dev/fd0, loader CHAIN
Pseudo partition start: 0
Added Floppy

The boot sector and the map file have *NOT* been altered.

[root]# lilo
Added Linux *
Added Old
Added DOS
Added Floppy
```

Une erreur fréquente consiste à oublier de lancer cette dernière commande. Aucune modification n'est alors visible au démarrage puisque le MBR est resté inchangé.

En mode non expert, le chargeur GRUB est installé par défaut. L'installation du paquetage *lilo* par la suite vous invite fortement à utiliser la commande **liloconfig** juste après. En effet, cette commande permet de créer simplement le fichier */etc/lilo.conf* en vous posant quelques questions. Pour la plupart des configurations machines, il n'est pas nécessaire d'y revenir. De plus, il lance dans la foulée la commande **lilo**.

Désinstallation

Enfin, pour désinstaller LILO et replacer le MBR dans son état initial, plusieurs méthodes existent :

- Sous Linux :

```
[root]# lilo -U
```

- Sous DOS/Windows 9x avec la disquette de boot :

```
A:\> fdisk /mbr
```

- Sous Windows XP, sous la console de récupération avec la commande **fixmbr <nom du périphérique>**.

Une documentation très fournie sur la configuration de **lilo** et le fonctionnement de ce gestionnaire d'amorçage accompagne LILO ; elle est installée dans le répertoire */usr/share/doc/lilo* lorsque ce chargeur est installé sur Debian Etch.

c. GRUB

GRUB (*GRand Unified Bootloader*), créé par Erich Boleyn en 1995, est le chargeur du système GNU/Hurd. Il est capable de démarrer à peu près tout, de Linux à Windows NT en passant par BeOS, FreeBSD, NetBSD, OpenBSD, Windows 9x, OS/2... ; Debian Etch configure ce gestionnaire d'amorçage par défaut lors de l'installation basique (hors mode expert où le choix est laissé à l'utilisateur).

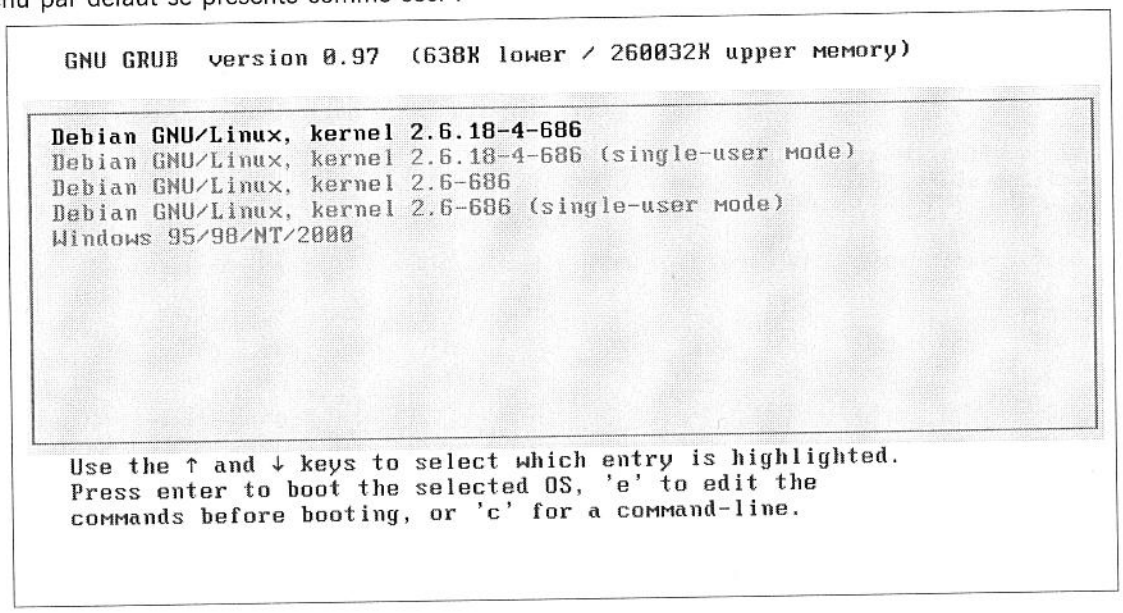
Plus souple et plus puissant que LILO, il reconnaît entre autres les systèmes de fichiers ext2, ext3, reiserfs, FAT32, JFS et XFS. Cela lui permet de stocker le chemin sur le système de fichiers et non l'adresse du fichier contenant le noyau à démarrer. Ainsi, même si le fichier noyau est déplacé physiquement sur la partition, à cause d'une défragmentation par exemple, il ne sera pas nécessaire dans son cas de reconstruire le MBR. Il est même envisageable de démarrer le système via le réseau.

De plus, sachant charger nativement les noyaux Linux, Hurd et *BSD, l'utilisation du "chainloading" ne sera obligatoire que pour les systèmes DOS/ Windows ; cela allège considérablement la gestion des chargeurs sur une machine abritant un grand nombre de systèmes d'exploitation.

Une autre caractéristique intéressante de GRUB est la mise à disposition d'un mini shell au démarrage permettant d'exécuter toutes les commandes internes. Ainsi, il sera possible de démarrer n'importe quelle machine à partir d'une disquette contenant GRUB, du moment que le noyau a été identifié.

Utilisation

Le menu par défaut se présente comme ceci :



Shell Grub

Pour obtenir une émulation du shell GRUB sous Linux, il suffit de saisir la commande **grub**. Le mini sh
obtenu nous permettra de lancer les mêmes commandes GRUB qu'au démarrage de la machine ainsi q
les instructions d'installation de GRUB dans le MBR.

```
[root]# grub
Probing devices to guess BIOS drives. This may take a long time.
...

    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ 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.]

grub>
```

Il est possible d'afficher la totalité des commandes supportées en utilisant la touche **[Tab]** (qui fait aussi
complétion des noms de fichiers) et d'en afficher la description avec la commande **help** ; la comman
quit permettant de sortir de l'interface :

```
grub>
Possible commands are: blocklist boot cat chainloader clear cmp color configfile
de bug device displayapm displaymem dump embed find fstest geometry halt help
hide imps probe initrd install ioprobe kernel lock makeactive map md5crypt
module modulenounzip pager partnew parttype password pause quit read reboot
rootrootnoverify savedefault serial setkey setup terminal terminfo
testload testvbe unhide uppermem vbeprobe

grub> help root
root: root [DEVICE [HDBIAS]]
    Set the current "root device" to the device DEVICE, then attempt
    to mount it to get the partition size (for passing the partition
    descriptor in `ES:ESI', used by some chain-loaded bootloaders),
    the BSD drive-type (for booting BSD kernels using their native
    boot format), and correctly determine  the PC partition where a
    BSD sub-partition is located. The optional HDBIAS parameter is a
    number to tell a BSD kernel how many BIOS drive numbers are on
    controllers before the current one. For example, if there is an
    IDE disk and a SCSI disk, and your FreeBSD root partition is on
    the SCSI disk, then use a '1' for HDBIAS.
                                                                .../.
```

```
.../...
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
    GRUB can read, but setting the correct root device is still
    desired. Note that the items mentioned in `root' which derived
    from attempting the mount will NOT work correctly.

grub> quit
```

Les noms des fichiers sont de la forme *(hd1,0)/boot/vmlinuz-2.6.18-4-686* où :

- *hd1* : unité de disque (*hd* pour tous les disques durs, *fd* pour les lecteurs de disquettes et *nd* pour les périphériques réseau) suivie de son ordre d'apparition (*0* étant le premier comme en langage C) ; cela représente donc ici le deuxième disque dur de la machine, soit */dev/hdb* si elle est dotée de disques IDE.
- *0* : numéro de la partition sur le disque ; comme pour le numéro de disque, on commence à partir de *0* ; soit ici la partition */dev/hdb1*.
- */boot/vmlinuz-2.6.18-4-686* : enfin le nom du fichier sur la partition puisque le système de fichiers est supporté. Attention, ce nom ne tient pas compte d'une éventuelle table de montage. Si la partition représente le répertoire */boot* sur votre système, il faut considérer le fichier */vmlinuz** qui se trouve effectivement à la racine de ce système de fichiers, et non */boot/vmlinuz** lorsqu'il est adressé sous Linux suivant la table de montage du système.

La commande GRUB **root** permettra de fixer les deux premières valeurs pour la suite des commandes :

```
grub> root (hd1,0)
 Filesystem type is ext2fs, partition type 0x83
```

Commandes de base et fichier de configuration

Pour charger un noyau Linux, il suffira de charger le fichier correspondant à l'aide de la commande **kernel** (suivie des paramètres nécessaires qui ont la même syntaxe que pour LILO), puis de lui passer la main avec la commande **boot** :

```
grub> kernel (hd1,0)/boot/vmlinuz-2.6.18-4-686 root=/dev/hdb1
   [Linux-bzImage, setup=0x1e00, size=0x131e9d]

grub> boot
```

Tous les paramètres pouvant être passés à Linux au démarrage sont décrits dans le fichier */usr/src/linux-source-<version>/Documentation/kernel-parameters.txt* lorsque les sources du noyau sont installées sur le système.

Dans le cas d'un système DOS/Windows à charger, il faudra utiliser le mécanisme de chaînage d'amorçage :

```
grub> chainloader (hd0,0)+1

grub> boot
```

Le fichier de configuration de GRUB est */boot/grub/menu.lst* ; en voici un exemple commenté :

```
# menu.lst - See: grub(8), info grub, update-grub(8)
#            grub-install(8), grub-floppy(8),
#            grub-md5-crypt, /usr/share/doc/grub
#            and /usr/share/doc/grub-doc/.

## default num
# Set the default entry to the entry number NUM. Numbering starts from 0, and
# the entry number 0 is the default if the command is not used.
#
# You can specify 'saved' instead of a number. In this case, the default entry
# is the entry saved with the command 'savedefault'.
default         0
                                                                    .../...
```

```
.../...
## timeout sec
# Set a timeout, in SEC seconds, before automatically booting the default entry
# (normally the first entry defined).
timeout     5

# Pretty colours
color cyan/blue white/blue

#
# Put static boot stanzas before and/or after AUTOMAGIC KERNEL LIST

### BEGIN AUTOMAGIC KERNELS LIST
## lines between the AUTOMAGIC KERNELS LIST markers will be modified
## by the debian update-grub script except for<+>the default options below

## DO NOT UNCOMMENT THEM, Just edit them to your needs

## ## Start Default Options ##

# plusieurs lignes de configuration par défaut mises en commentaires

## ## End Default Options ##

title          Debian GNU/Linux, kernel 2.6.18-4-686
root           (hd1,0)
kernel         /boot/vmlinuz-2.6.18-4-686 root=/dev/hdb1 ro
initrd         /boot/initrd.img-2.6.18-4-686
savedefault
boot
title          Debian GNU/Linux, kernel 2.6.18-4-686 (recovery mode)
root           (hd1,0)
kernel         /boot/vmlinuz-2.6.18-4-686 root=/dev/hdb1 ro single
initrd         /boot/initrd.img-2.6.18-4-686
savedefault
boot
title          Debian GNU/Linux, kernel 2.6-686
root           (hd1,0)
kernel         /boot/vmlinuz-2.6-686 root=/dev/hdb1 ro
initrd         /boot/initrd.img-2.6-686
savedefault
boot
title          Debian GNU/Linux, kernel 2.6-686 (recovery mode)
root           (hd1,0)
kernel         /boot/vmlinuz-2.6-686 root=/dev/hdb1 ro single
initrd         /boot/initrd.img-2.6-686
savedefault
boot

### END DEBIAN AUTOMAGIC KERNELS LIST

# This is a divider, added to separate the menu items below from the Debian
# ones.
title      Other operating systems:
root


# This entry automatically added by the Debian installer for a non-linux OS
# on /dev/hda1
                                                                    .../...
```

```
.../...
title          Microsoft Windows 2000 Professionnel
root           (hd0,0)
savedefault
makeactive
chainloader    +1
```

Si GRUB n'a pas été retenu lors de l'installation de la distribution, il sera nécessaire d'installer les fichiers de base ; cela est facilement réalisable avec la commande **grub-install**, suivie du nom du périphérique où le MBR doit être inscrit :

```
root# grub-install /dev/hda
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(hd0)    /dev/hda
(hd1)    /dev/hdb
```

Comme GRUB sait accéder aux fichiers sur les systèmes de fichiers, il est inutile de reconstruire le MBR ; il peut lire dès le démarrage de la machine son fichier de configuration et ainsi, tenir compte des modifications.

Désinstallation

Pour supprimer GRUB du MBR, il suffit d'installer un autre chargeur Linux ou d'utiliser les commandes DOS/Windows adéquates afin de restaurer le bootloader Microsoft.

3. Chargement du noyau

Une fois chargé par le gestionnaire de démarrage, le noyau Linux effectue plusieurs opérations :

- Détection du matériel. Cette opération a déjà été réalisée par le BIOS mais les systèmes d'exploitation récents préfèrent se passer des routines fournies par celui-ci pour des raisons d'optimisation et de support des dernières technologies. C'est ici que Linux va parcourir les différents bus (PCI, AGP...) de communication du système et reconnaître les cartes filles présentes dans la machine. Les protocoles réseau de bas niveau sont définis à ce stade. Les nombreux messages affichés à l'écran sont temporairement conservés dans le fichier */proc/kmsg* et visualisables en employant la commande **dmesg**.
- Montage du système de fichiers principal contenant la racine / pour accéder à certains fichiers comme le binaire du premier programme du système. Cette partition est alors montée en lecture seule, ce qui permettra de vérifier le système de fichiers ; elle sera remontée plus tard en écriture. Ainsi on pourra travailler dessus.
- Lancement du premier processus du système : Init.

4. Disque initrd

Il peut manquer dans le code du noyau Linux une partie des pilotes et fonctionnalités nécessaires au démarrage du système ; ceux-ci ayant été compilés en modules (cf. chapitre 14 - Compilation du noyau Linux).

Par exemple, si le support de l'adaptateur SCSI contrôlant le disque sur lequel se trouve la partition racine est en module, le noyau ne pourra pas charger ce dernier tant qu'il n'aura pas monté la partition principale (elle ne peut être montée puisque le module se trouve sur celle-ci).

La solution consiste à fabriquer l'image d'un disque initial qui contiendra tous les modules nécessaires au démarrage du système. Cette image, sous forme de fichier, sera chargée en mémoire à l'initialisation de la machine par GRUB au même titre que le noyau Linux ; on la nomme généralement "initrd" pour "*initial RAM Disk*" et on place son image dans */boot* comme celle du noyau.

B. Processus Init et niveaux d'exécution

Nous avons vu que le système GNU/Linux était un Unix empruntant à la fois des concepts de la famille "System V" et de la famille "BSD".

Sa procédure d'initialisation suit celle des Unix "System V" et utilise le concept de niveaux d'exécution ("runlevel") pour définir les services devant être lancés au démarrage.

Une fois son chargement terminé, le noyau Linux lance le processus Init (*/sbin/init*) qui prend en charge la fin de l'initialisation du système ; étant le premier processus lancé, il possède le PID 1. L'affichage correspondant au début de cette phase lors du démarrage est reconnaissable sous Debian Etch par le message "INIT: version 2.86 booting".

Le processus Init a pour mission d'initialiser l'environnement logiciel du système et de lancer un certain nombre d'autres processus qui seront autant de services ou sous-systèmes.

Les tâches d'initialisation comprennent notamment :

- le montage des systèmes de fichiers */proc* et */sys* ;
- l'initialisation de certains paramètres du noyau ;
- la mise à l'heure du système par rapport à l'heure matérielle ;
- la définition des consoles texte ;
- la définition du nom de la machine ;
- la détection des périphériques USB ;
- le chargement du support RAID et LVM ;
- l'activation des quotas ;
- la vérification des systèmes de fichiers et le montage de ceux-ci (ainsi que le remontage du système de fichiers racine qui était monté en lecture seule jusqu'à présent) ;
- l'activation du swap ;
- le lancement du démon Syslog de journalisation des messages ;
- le chargement des modules du noyau ;
- le nettoyage de certains fichiers temporaires ;
- l'affectation des valeurs de base à certaines variables d'environnement comme **PATH** ou **RUNLEVEL**.

Les services démarrés dépendront, quant à eux, du niveau d'exécution spécifié au processus Init.

Il existe huit runlevels définis sous GNU/linux, dont quatre réservés :

- **0** : arrêt de la machine ("halt") ;
- **1** : mode mono-utilisateur ou maintenance ;
- **6** : redémarrage de la machine ("reboot") ;
- **S** ou **s** : mode mono-utilisateur dans lequel seule la partition racine est montée ; utile pour vérifier e réparer les autres systèmes de fichiers.

Le mode mono-utilisateur est interprété différemment suivant les distributions. Pour le System V, l niveau **S** était prévu pour la vérification et la réparation des systèmes de fichiers où seul le système d fichiers racine était monté ; aucun service n'était lancé. Tandis que le niveau **1** était utilisé pour les tâche d'administration système comme l'ajout et la suppression de paquetages ou la modification de la configura tion ; certains services étaient alors lancés même si le réseau était désactivé. Sous Debian Etch, les niveau **1** et **S** sont distincts. Des services sont configurés pour être lancés au niveau **S** tandis qu'au niveau **1** aucun service n'est lancé.

Les autres niveaux d'exécution sont, quant à eux, définissables à souhait et permettent de distingue différents modes d'utilisation du système. Pour Debian Etch :

- **2** : mode multiutilisateur et connexion graphique si présent (Niveau de démarrage par défaut sou Debian) ;
- **3** : libre ;
- **4** : libre ;
- **5** : libre.

De plus, l'administrateur pourra modifier cela à loisir en sélectionnant les services associés à chaqu runlevel.

Si un gestionnaire de connexion est installé, le démarrage se fait en mode graphique au niveau 2 par défaut.

1. Fichier de configuration /etc/inittab

Pour définir le comportement du système et les services à démarrer en fonction du niveau d'exécution, Init se réfère au fichier de configuration */etc/inittab*.

Voici un exemple de ce fichier sur Debian Etch :

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:2:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin

# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit/etc/
inittab to let this work."

# What to do when the power fails/returns.
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
Format:
#  <id>:<runlevels>:<action>:<process>
#
                                                                .../...
```

```
.../...
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6

# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100

# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Les lignes commençant par "**#**" sont des commentaires, les autres sont composées de quatre champs sép rés par "**:**" :

- Identifiant : ce premier champ doit être unique et comprendre de 1 à 4 caractères alphanumériques.
- Liste des niveaux d'exécution concernés : ce champ contient le numéro (ou lettre) de chaque runlev concerné par la ligne. Si aucun niveau n'est spécifié, la ligne s'appliquera à tous les runlevels.
- Action : méthode ou manière d'exécuter la commande spécifiée dans le champ suivant.
- Commande ou processus : chemin de la commande à lancer avec ses paramètres pour les nivea d'exécution définis précédemment.

Les différentes directives pour le troisième champ "action" sont :

- **respawn** : le processus sera relancé s'il se termine. Utilisé principalement avec **mingetty** pour assu la gestion d'un certain nombre de terminaux texte.
- **once** : le processus n'est exécuté qu'une fois.
- **wait** : identique à la directive précédente mais ici, Init attend que le processus soit terminé avant passer à la ligne suivante.
- **boot** : le processus est exécuté au démarrage du système, le champ "runlevels" est ignoré.
- **bootwait** : identique à la directive précédente mais ici, Init attend que le processus soit terminé ava de passer à la ligne suivante.
- **off** : ne fait rien ; cela revient à commenter la ligne.
- **ondemand** : identique à **respawn** mais cette directive utilise les pseudos runlevels **a**, **b** et **c**. Ce permet de demander à Init d'entreprendre une action sans changer de niveau d'exécution.
- **initdefault** : définit le niveau d'exécution par défaut au démarrage du système. Si cette directive e absente, Init le demandera sur la console. Le champ commande est ignoré.
- **sysinit** : la commande sera exécutée au démarrage, avant même les lignes contenant les directiv **boot** ou **bootwait**. Le deuxième champ sera ignoré de la même manière.
- **powerfail** : la commande est exécutée lorsque Init reçoit le signal SIGPWR signifiant q l'alimentation est sur le point d'être interrompue. Ce signal est généralement envoyé par un matériel gestion d'énergie tel qu'un UPS.
- **powerwait** : identique à la directive précédente mais ici, Init attend que le processus soit terminé ava de passer à la ligne suivante.
- **powerokwait** : le processus est exécuté si Init est informé du rétablissement de l'alimentation.
- **powerfailnow** : le processus est exécuté si Init est informé que l'accumulateur de l'UPS externe e presque vide.
- **ctrlaltdel** : la commande est lancée quand Init reçoit le signal SIGINT. Ceci se produit lorsqu' appuie sur les touches **[Ctrl]** **[Alt]** [**Suppr]** simultanément à partir de la console. Le processus e généralement **shutdown** pour passer en mode monoutilisateur ou redémarrer le système.
- **kbrequest** : permet de lancer le processus suivant certaines séquences de touches saisies au clavier.

L'analyse du fichier de configuration précédent nous apprend donc que :

- Le niveau d'exécution par défaut est le numéro **2**.
- La commande **/etc/init.d/rcS** est lancée en premier lieu.
- La commande **/etc/init.d/rc** est lancée pour chaque niveau d'exécution avec son numéro en argument.
- La commande **/sbin/shutdown** est exécutée avec différents arguments lors de l'appui sur les touches **[Ctrl] [Alt] [Suppr]** et lors d'un événement concernant l'alimentation.
- Six terminaux virtuels "tty" sont initialisés avec la commande **/sbin/getty** pour les niveaux d'exécution **2** et **3** et un seul pour les niveaux **4** et **5**. De plus, ceux-ci sont relancés s'ils se terminent.

2. Contrôle du processus Init

Ce mécanisme permet de changer de niveau d'exécution à tout moment. L'administrateur pourra donc modifier l'état du système en contrôlant le processus Init.

Niveau d'exécution courant

Avant toute chose, on pourra connaître le niveau d'exécution actuel grâce à la commande **runlevel**. Celle-ci indique le niveau d'exécution actuel ainsi que le précédent (le **N** en première position indiquant que le système n'a pas changé de niveau depuis le démarrage de la machine) :

```
[root]# runlevel
N 2
```

Changement de niveau d'exécution

Pour contrôler le processus Init et changer de runlevel, il suffit d'appeler la commande **init** ou **telinit** avec le niveau d'exécution (**0**, **1**, **2**, **3**, **4**, **5**, **6**, **S** ou **s**) ou le pseudo- niveau d'exécution (**a**, **b** ou **c**) en argument :

```
root# runlevel
N 2
root# telinit 1
INIT: Switching to runlevel: 1
INIT: Sending processes the TERM signal
INIT: Sending processes the KILL signal
Stopping GNOME Display Manager: gdm not running.
Stopping periodic command scheduler: cron.
Stopping Hardware abstraction layer: hald.
Stopping system message bus: dbus-1.
Stopping mouse interface server: gpm.
Stopping internet superserver: inetd.
Stopping printer spooler: lpd [not running]
Stopping mail transport agent: Postfix.
Stopping OpenBSD Secure Shell server: sshd.
Stopping file alteration monitor: FAM.
Shutting down ALSA...done (not loaded).
Stopping NFS common utilities: statd.
Stopping portmap daemon: portmap.
Stopping deferred execution scheduler: atd.
Stopping kernel log daemon: klogd.
Stopping system log daemon: syslogd.
Sending all processes the TERM signal...done.
Sending all processes the KILL signal...done.
Entering single-user mode...
INIT: Going single user
INIT: Sending processes the TERM signal
Give root password for maintenance
(or type Control-D to continue):
root# runlevel
1 S
```

Il peut paraître étonnant que l'indication donnée par la commande runlevel soit "1 S" et non pas "2 1"! Même si le niveau 1 est demandé, il y a basculement automatique vers le mode single comme les traces d'exécution le montrent.

Cette commande accepte en plus l'argument **Q** (ou **q**) pour spécifier à Init de relire le fichier */etc/inittab* et de prendre en compte les modifications apportées à celui-ci immédiatement.

Le délai par défaut que donne Init aux processus pour s'arrêter lors du changement de runlevel est de ci secondes ; c'est le temps entre l'envoi du signal **SIGTERM** et **SIGKILL**. Ce temps peut être modifié ajoutant l'option **-t** sur la ligne de commande de **telinit** ; pour le passer à dix secondes :

```
[root]# telinit -t 10 3
INIT: Switching to runlevel: 3
...
```

Démarrage en mode mono-utilisateur

De plus, il est possible de forcer le niveau d'exécution du système au démarrage de la machine en spé fiant son numéro au noyau à charger (ou le mot "single" pour le runlevel **S**). Ainsi avec LILO, il faudra sa au démarrage :

```
boot:
Linux       Old       DOS       Floppy
boot: linux single
```

Sous Grub, il faudra éditer les arguments de l'une des entrées en appuyant sur la touche **[e]**, puis vali après l'ajout de **single** en fin de ligne :

```
   GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

      [ 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. ]

grub> kernel /boot/vmlinuz-2.6.18-4-686 root=/dev/hdb1 ro single_
```

Le mode mono-utilisateur permet de se connecter à des fins de maintenance du système. Il ne v dispense pas d'entrer le mot de passe du compte administrateur. Ce comportement par défaut est confi dans */etc/inittab* où il est indiqué que la commande à lancer est **/sbin/sulogin** lorsque le niv S est atteint. Il est recommandé de garder les choses en l'état pour des questions de sécurité.

3. Scripts de démarrage

Un certain nombre de commandes exécutées par **init** se trouvent dans les répertoires */etc/init.* */etc/rc?.d* (? peut être remplacé par 0, 1, 2, 3, 4, 5, 6, S).

Ces répertoires contiennent:

```
root# ls -l /etc/init.d
total 154
-rwxr-xr-x  1 root root 11923 2005-02-24 20:03 alsa
-rwxr-xr-x  1 root root  1074 2002-01-18 09:13 atd
-rw-r--r--  1 root root  2593 2004-09-10 17:00 bootclean.sh
-rwxr-xr-x  1 root root  1529 2005-01-04 23:43 bootlogd
-rwxr-xr-x  1 root root  1371 2004-09-10 17:00 bootmisc.sh
-rwxr-xr-x  1 root root   935 2004-09-10 17:00 checkfs.sh
-rwxr-xr-x  1 root root  7160 2004-09-10 17:00 checkroot.sh
-rwxr-xr-x  1 root root  5366 2005-02-05 10:17 console-screen.sh
-rwxr-xr-x  1 root root  1096 2004-07-28 16:12 cron
-rwxr-xr-x  1 root root  1878 2005-04-05 23:11 dbus-1
-rwxr-xr-x  1 root root  7823 2005-01-09 01:16 discover
...
-rwxr-xr-x  1 root root  2235 2004-09-10 17:00 rc
-rwxr-xr-x  1 root root  1190 2004-09-10 17:00 rcS
...
```

- */etc/init.d/rcS* : nous avons vu que cette commande est exécutée en premier lieu par Init ; er c'est un script shell qui contient toutes les commandes préalables d'initialisation du système. Sa le est très enrichissante. Ce script lance tous les scripts de */etc/rcS.d/*.

- toujours dans */etc/init.d* : les scripts shell permettant de lancer tous les services sur le système. Les scripts présents ici doivent obligatoirement supporter l'argument **start** pour lancer le service et l'argument **stop** pour l'arrêter. Une grande majorité de ces scripts supporte aussi les arguments **restart**, **reload** et **status**.

```
root# /etc/init.d/cron
Usage: /etc/init.d/cron start|stop|restart|reload|force-reload
```

- */etc/rc?.d* : ces répertoires contiennent des liens vers les scripts du répertoire *init.d* à lancer lorsque Init entre dans le niveau d'exécution correspondant au numéro du répertoire. Le nom des liens se trouvant ici est important car il détermine la façon dont ils seront appelés. Le script **rc** lancera les scripts contenus dans le répertoire correspondant au niveau d'exécution passé en argument. Par exemple, le répertoire *rc3.d* contient :

```
root# ls -l /etc/rc3.d
total 0
lrwxrwxrwx  1 root root 15 2006-03-28 13:10 K20exim4 -> ../init.d/exim4
lrwxrwxrwx  1 root root 18 2006-03-28 13:10 K20festival -> ../init.d/festival
lrwxrwxrwx  1 root root 13 2006-03-27 13:35 K20gpm -> ../init.d/gpm
lrwxrwxrwx  1 root root 13 2006-03-28 13:11 K20lpd -> ../init.d/lpd
lrwxrwxrwx  1 root root 14 2006-03-28 13:11 K40alsa -> ../init.d/alsa
lrwxrwxrwx  1 root root 20 2006-03-28 13:11 K79nfs-common -> ../init.d/nfs-common
lrwxrwxrwx  1 root root 13 2006-03-27 13:35 K86ppp -> ../init.d/ppp
lrwxrwxrwx  1<+>root root 18 2006-03-27 11:05 S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx  1 root root 15 2006-03-27 11:05 S11klogd -> ../init.d/klogd
lrwxrwxrwx  1 root root 17 2006-03-27 12:20 S18portmap -> ../init.d/portmap
lrwxrwxrwx  1 root root 16 2006-03-28 00:17 S20dbus-1 -> ../init.d/dbus-1
lrwxrwxrwx  1 root root 15 2006-03-27 11:03 S20inetd -> ../init.d/inetd
lrwxrwxrwx  1 root root 17 2006-03-27 10:58 S20makedev -> ../init.d/makedev
lrwxrwxrwx  1 root root 17 2006-03-27 21:20 S20postfix -> ../init.d/postfix
lrwxrwxrwx  1 root root 13 2006-03-27 12:22 S20ssh -> ../init.d/ssh
lrwxrwxrwx  1 root root 13 2006-03-28 00:18 S21fam -> ../init.d/fam
lrwxrwxrwx  1 root root 13 2006-03-27 11:05 S89atd -> ../init.d/atd
lrwxrwxrwx  1 root root 14 2006-03-27 11:03 S89cron -> ../init.d/cron
lrwxrwxrwx  1 root root 13 2006-03-28 00:37 S99gdm -> ../init.d/gdm
lrwxrwxrwx  1 root root 19 2006-03-27 10:59 S99rmnologin -> ../init.d/rmnologin
lrwxrwxrwx  1 root root 23 2006-03-27 10:59 S99stop-bootlogd -> ../init.d/
stop-bootlogd
```

- *rc* : cette commande parcourt le répertoire correspondant au runlevel passé en argument et exécute chaque script qui y figure en lançant d'abord ceux commençant par la lettre **K** (*kill*) avec l'argument **stop**, puis ceux débutant par la lettre **S** (*start*) avec l'argument **start** ; le numéro suivant la lettre indiquant l'ordre d'exécution. Si deux fichiers ont le même numéro, l'ordre alphabétique primera.

Cette gestion des scripts de démarrage peut paraître complexe au premier abord, mais elle permet de centraliser dans un même script le contrôle complet (**start** et **stop**) d'un service, tout en autorisant plusieurs niveaux d'exécution du système ; ceci est fondamental pour la maintenance de ces scripts.

La structure générale d'un de ces scripts shell est la suivante :

```
#!/bin/sh
# Start/stop the service


case "$1" in
start)
        # commandes de lancement de service
   ;;
stop)
        # commandes d'arrêt du service
       ;;
restart)
        # commandes de redémarrage du service
       ;;
                                                                    .../...
```

```
.../...
reload|force-reload)
        # commandes de rechargement de la configuration du service
        ;;

*)   echo "Usage: $0 start|stop|restart|reload|force-reload"
        exit 1
        ;;
esac
exit 0
```

De plus, tous ces scripts peuvent être lancés manuellement par l'administrateur sur la ligne de comma des ; il n'est donc pas nécessaire de redémarrer un système Debian Etch après l'ajout d'un service ou modification de sa configuration.

Par exemple, pour relancer le serveur de messagerie directement à partir de la ligne de commandes :

```
root# /etc/init.d/postfix restart
Stopping mail transport agent: Postfix.
Starting mail transport agent: Postfix.
```

4. Ajout et suppression de services au démarrage

Le programme **/etc/init.d/rc** exécutant tous les scripts présents dans le répertoire du runlevel, suffira d'ajouter un lien dans ce répertoire vers le script adéquat pour lancer (ou arrêter) un service lorsq le système entre dans ce niveau d'exécution.

Par exemple, pour que le serveur Web soit lancé automatiquement au démarrage (nous supposons ici q **initdefault** est positionné à **3** dans */etc/inittab*), sachant que le script contrôlant ce service *apache2* est */etc/init.d/apache2*, il faudra créer un lien vers ce script en respectant certains points :

- Ce lien doit se trouver dans */etc/rc3.d* puisque c'est le répertoire du runlevel par défaut.
- Le nom de ce lien doit commencer par la lettre *S* puisque l'on veut démarrer le service.
- Le numéro suivant la première lettre doit tenir compte des autres services devant être lancés préalable.
- La fin de ce nom de fichier, bien que peu importante, devrait être significative : *apache2* est un b choix ici.

En plus de tout cela, il ne faudra pas oublier de supprimer le lien *K91apache2* déjà présent ; ainsi :

```
[root]# cd /etc/rc3.d
[root]# ls -l *apache2*
lrwxrwxrwx  1 root root 17 Feb 19 21:07 K91apache2 -> ../init.d/apache2
[root]# mv K91apache2 S91apache2
[root]# ls -l *apache2*
lrwxrwxrwx  1 root root 17 Feb 19 21:07 S91apache2 -> ../init.d/apache2
```

La suppression du lien *K91apache2* et la création du lien *S91apache2* pointant sur le même fich */etc/init.d/apache2*, sont réalisées via la commande **mv** (déplacement/changement de nom).

Pour réaliser cette opération, il existe des outils en ligne de commandes ou avec une interface graphic permettant d'en simplifier la tâche.

Parmi les commandes en ligne, **update-rc.d** fournie en standard, permet de manipuler ces liens. outil est d'ailleurs utilisé lors de l'installation ou de la désinstallation d'un paquet Debian comportant service, dans les scripts **postinst** et **postrm** dudit paquet.

La constitution détaillée d'un paquet Debian dépasse le cadre de cet ouvrage.

L'exemple qui suit nous montre une séquence de commandes **update-rc.d**, qui effectuent pour le service Web une suppression de tous les liens et une création par défaut des liens.

```
[root]# update-rc.d -f apache2 remove
update-rc.d: /etc/init.d/apache2 exists during rc.d purge (continuing)
 Removing any system startup links for /etc/init.d/apache2 ...
   /etc/rc0.d/K91apache2
   /etc/rc1.d/K91apache2
   /etc/rc2.d/S91apache2
   /etc/rc3.d/K91apache2
   /etc/rc4.d/S91apache2
   /etc/rc5.d/S91apache2
   /etc/rc6.d/K91apache2
[root]# update-rc.d apache2 defaults 91
 Adding system startup for /etc/init.d/apache2 ...
   /etc/rc0.d/K91apache2 -> ../init.d/apache2
   /etc/rc1.d/K91apache2 -> ../init.d/apache2
   /etc/rc6.d/K91apache2 -> ../init.d/apache2
   /etc/rc2.d/S91apache2 -> ../init.d/apache2
   /etc/rc3.d/S91apache2 -> ../init.d/apache2
   /etc/rc4.d/S91apache2 -> ../init.d/apache2
   /etc/rc5.d/S91apache2 -> ../init.d/apache2
```

À noter que l'option **-f** est utilisée pour forcer une action, et que dans la mise en place des liens par défaut, le service Web est activé aux niveaux **2**, **3**, **4** ,**5** et désactivé aux niveaux **0**, **1** et **6**.

Il est possible d'utiliser cette commande avec un plus grand contrôle, la commande précédente peut être réalisée de cette façon :

```
[root]# update-rc.d apache2 stop 91 0 1 6 . start 91 2 3 4 5 .
Adding system startup for /etc/init.d/apache2 ...
   /etc/rc0.d/K91apache2 -> ../init.d/apache2
   /etc/rc1.d/K91apache2 -> ../init.d/apache2
   /etc/rc6.d/K91apache2 -> ../init.d/apache2
   /etc/rc2.d/S91apache2 -> ../init.d/apache2
   /etc/rc3.d/S91apache2 -> ../init.d/apache2
   /etc/rc4.d/S91apache2 -> ../init.d/apache2
   /etc/rc5.d/S91apache2 -> ../init.d/apache2
```

Un autre outil en ligne vient à bon escient seconder **update-rc.d**, il s'agit de **sysv-rc-conf**, qui via une interface *ncurses*, permet de configurer très facilement tous les services installés ainsi que leurs niveaux de démarrage et d'arrêt :

```
SysV Runlevel Config   -: stop service  +/*: start service  h: help  q: quit

service      1     2     3     4     5     0     6     S
-------------------------------------------------------------------------------
alsa        [_]   [ ]   [ ]   [X]   [X]   [ ]   [ ]   [ ]
atd         [ ]   [X]   [X]   [X]   [X]   [ ]   [ ]   [ ]
bootlogd    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [X]
cron        [ ]   [X]   [X]   [X]   [X]   [ ]   [ ]   [ ]
dbus-1      [ ]   [X]   [X]   [X]   [X]   [ ]   [ ]   [ ]
discover    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [X]
dns-clean   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [X]
exim4       [ ]   [ ]   [ ]   [X]   [X]   [ ]   [ ]   [ ]
fam         [ ]   [X]   [X]   [X]   [X]   [ ]   [ ]   [ ]
festival    [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]
gdm         [ ]   [ ]   [X]   [X]   [X]   [ ]   [ ]   [ ]
gpm         [ ]   [X]   [ ]   [X]   [X]   [ ]   [ ]   [ ]
halt        [ ]   [ ]   [ ]   [ ]   [ ]   [X]   [ ]   [ ]
hotplug     [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [ ]   [X]

Use the arrow keys or mouse to move around.      ^n: next pg      ^p: prev pg
                    space: toggle service on / off
```

5. Répertoire /etc/rcS.d

Le premier script appelé par **init** est */etc/init.d/rcS* et ce dernier appelle tous les scripts trouvant dans */etc/rcS.d*. Les scripts qui s'y trouvent permettent de définir les paramètres du systèr d'effectuer des initialisations comme le chargement des modules, le démarrage des services réseau, réglage de l'horloge.

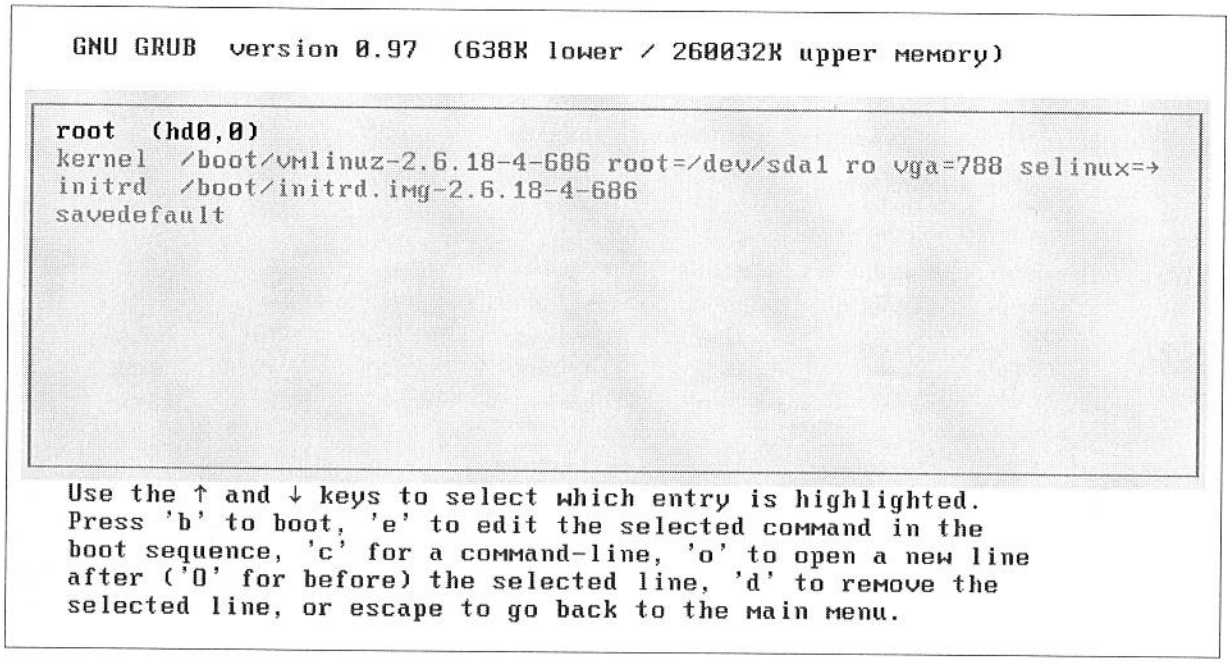
Ce répertoire contient un grand nombre de liens vers des scripts représentant les différents aspects de configuration système :

```
root# ls -l /etc/rcS.d
total 1
-rw-r--r--  1 root root 701 2004-09-10 17:00 README
lrwxrwxrwx  1 root root  21 2006-03-27 10:59 S02mountvirtfs -> ../init.d/
mountvirtfs
lrwxrwxrwx  1 root root  14 2006-03-28 00:22 S04udev -> ../init.d/udev
lrwxrwxrwx  1 root root  18 2006-03-27 10:59 S05bootlogd -> ../init.d/bootlogd
lrwxrwxrwx  1 root root  25 2006-03-27 11:06 S05initrd-tools.sh -> ../init.d/
initrd-tools.sh
lrwxrwxrwx  1 root root  19 2006-03-27 11:05 S05keymap.sh -> ../init.d/keymap.
shlrwxrwxrwx  1 root root  22 2006-03-27 10:59 S10checkroot.sh -> ../init.d/
checkroot.sh
lrwxrwxrwx  1 root root  25 2006-03-27 10:59 S18hwclockfirst.sh -> ../init.d/
hwclockfirst.sh
lrwxrwxrwx  1 root root  24 2006-03-27 11:04 S18ifupdown-clean -> ../init.d/
ifupdown-clean
lrwxrwxrwx  1 root root  27 2006-03-27 12:19 S20module-init-tools -> ../init.d/
module-init-tools
lrwxrwxrwx  1 root root  18 2006-03-27 10:59 S20modutils -> ../init.d/modutils
lrwxrwxrwx  1 root root  26 2006-03-27 12:47 S25libdevmapper1.01 -> ../init.d/
libdevmapper1.01
lrwxrwxrwx  1 root root  20 2006-03-27 10:59 S30checkfs.sh -> ../init.d/
checkfs.sh
lrwxrwxrwx  1 root root  19 2006-03-27 10:59 S30procps.sh -> ../init.d/procps.shl
rwxrwxrwx  1 root root  21 2006-03-27 10:59 S35mountall.sh -> ../init.d/
mountall.sh
lrwxrwxrwx  1 root root  18 2006-03-27 11:09 S36discover -> ../init.d/discover
lrwxrwxrwx  1 root root  21 2006-03-27 10:59 S36mountvirtfs -> ../init.d/
mountvirtfs
lrwxrwxrwx  1 root root  19 2006-03-28 00:22 S36udev-mtab -> ../init.d/
udev-mtablrwxrwxrwx  1 root root  18 2006-03-27 11:05 S38pppd-dns -> ../init.d/
pppd-dns
lrwxrwxrwx  1 root root  19 2006-03-27 11:05 S39dns-clean -> ../init.d/
dns-cleanlrwxrwxrwx  1 root root  18 2006-03-27 11:04 S39ifupdown -> ../init.d/
ifupdown
lrwxrwxrwx  1 root root  21 2006-03-27 10:59 S40hostname.sh -> ../init.d/
hostname.sh
lrwxrwxrwx  1 root root  17 2006-03-27 11:09 S40hotplug -> ../init.d/hotplug
lrwxrwxrwx  1 root root  20 2006-03-27 11:04 S40networking -> ../init.d/
networking
lrwxrwxrwx  1 root root  21 2006-03-27 11:09 S41hotplug-net -> ../init.d/
hotplug-net
lrwxrwxrwx  1 root root  17 2006-03-27 12:20 S43portmap -> ../init.d/portmap
lrwxrwxrwx  1 root root  21 2006-03-27 10:59 S45mountnfs.sh -> ../init.d/
mountnfs.sh
lrwxrwxrwx  1 root root  27 2006-03-27 11:05 S48console-screen.sh -> ../init.d/
console-screen.sh
lrwxrwxrwx  1 root root  20 2006-03-27 10:59 S50hwclock.sh -> ../init.d/
hwclock.sh
lrwxrwxrwx  1 root root  21 2006-03-27 10:59 S55bootmisc.sh -> ../init.d/
bootmisc.sh
rwxrwxrwx  1 root root  17 2006-03-27 10:59  S55urandom -> ../init.d/urandom
lrwxrwxrwx  1 root root  17 2006-03-27 11:03 S70nviboot -> ../init.d/nviboot
lrwxrwxrwx  1 root root  24 2006-03-27 21:15 S70xfree86-common -> ../init.d/
xfree86-common
lrwxrwxrwx  1 root root  14 2006-03-28 00:18 S75sudo -> ../init.d/sudo
```

6. Remplacement du processus Init

Il est possible de remplacer le processus **Init** par un autre programme. À titre d'exemple, la manipulation qui suit remplace le processus **Init** par un simple **shell**. Ceci permet, sans nécessiter un CD-Rom de secours, d'accéder à un environnement administrateur sans mot de passe afin d'intervenir sur le système.

Au démarrage du système, avec GRUB comme gestionnaire de démarrage, il suffit de sélectionner un noyau Linux et de l'éditer en ajoutant `init=/bin/sh` à la ligne des paramètres du noyau (kernel) pour arriver à l'écran ci-dessous :



Un appui sur la touche **[b]** démarre le système et lance **/bin/sh** en lieu et place de **init**. La partition racine "/" est montée en lecture seule, il faut la remonter ainsi que celles souhaitées pour une intervention en écriture (cf. chapitre 6).

En cas de perte du mot de passe du compte root, sa réinitialisation ou modification se déroulerait comme suit :

```
sh-2.05b# mount -n -o remount,rw /
sh-2.05b# vi /etc/passwd
sh-2.05b# vi /etc/shadow
```

L'interprétation et la modification des fichiers */etc/passwd* et */etc/shadow* sont vues au chapitre 5.

C. Arrêt d'une machine Linux

L'arrêt d'un système GNU/Linux ne doit pas se faire en mettant hors tension la machine. Pour éviter toute mauvaise surprise comme la perte de données, il faut exécuter un certain nombre de tâches avant de pouvoir couper le courant, notamment :

- Prévenir les utilisateurs connectés au système de l'arrêt imminent de la machine pour qu'ils puissent sauvegarder leur travail.
- Arrêter tous les services : cela se résume à placer dans */etc/rc[016].d* un lien commençant par la lettre *K* vers tous les scripts se trouvant dans */etc/init.d*. Ces liens ont déjà été créés lors de l'installation. Cela permet à Init de terminer normalement les services tournant sur le système.
- Inscrire toutes les données contenues dans les tampons ("buffers") en mémoire sur le disque et démonter les systèmes de fichiers. Sans cela, les données se trouvant dans le cache disque et non encore écrites sur les unités de stockage seront perdues.

1. Commandes

a. shutdown

La commande **shutdown** permet d'arrêter, de redémarrer et de passer le système en mode maintenance Elle offre la possibilité, en outre, de programmer cette opération à une date précise et d'en informer le utilisateurs. Si l'arrêt du système est prévu dans moins de cinq minutes, la commande **shutdow** empêchera tout utilisateur, autre que **root**, de se connecter.

Son rôle principal est donc de prévenir les utilisateurs connectés à la machine et de changer le nivea d'exécution actuel pour le runlevel **0**, **1** ou **6**, comme avec la commande **telinit**.

Sa syntaxe est :

```
/sbin/shutdown [-t sec] [-arkhncfFHP] heure [message]
```

L'heure peut être spécifiée de plusieurs manières :

- **hh:mm** : heure à laquelle l'opération est programmée ;
- **[+]m**: nombre de minutes avant que l'opération soit effectuée ;
- **now** : seconde façon d'exprimer que l'opération doit être immédiate (alias de **+0**).

Les options à retenir de cette commande sont :

- **-h** : arrêter le système ;
- **-r** : redémarrer le système ;
- **-c** : annuler l'opération d'arrêt ou de redémarrage programmée ;
- **-f** : effectuer un redémarrage rapide sans vérification des systèmes de fichiers ;
- **-F** : forcer la vérification des systèmes de fichiers au prochain démarrage ;
- **-H** : après l'arrêt du système, la machine est arrêtée ou placée dans le gestionnaire de démarrage, pour les systèmes qui le permettent.
- **-P** : arrêter le système met la machine hors tension.

Pour arrêter le système dans dix minutes :

```
[root]# shutdown -h +10

Broadcast message from root (pts/1) (Wed Jun 15 19:26:17 2005):

The system is going DOWN for system halt in 10 minutes!
```

Pour annuler l'opération précédente et redémarrer le système (à partir d'un autre terminal) :

```
[root]# shutdown -c
[root]# shutdown -r now
```

L'emploi de cette commande peut être limité à certains utilisateurs et un contrôle d'accès est disponible v le fichier */etc/shutdown.allow*.

b. halt, reboot et poweroff

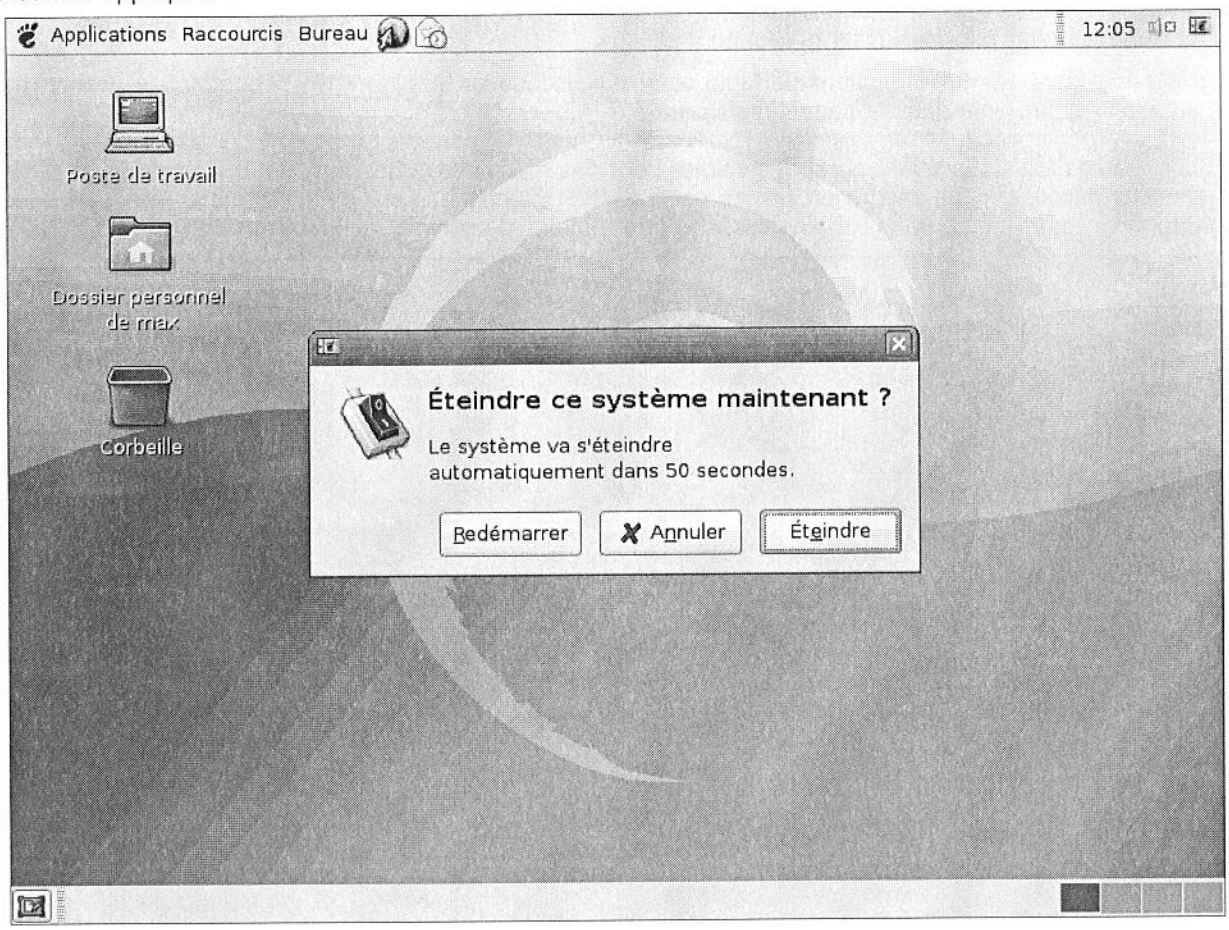
Les commandes d'arrêt **halt**, **reboot** et **poweroff** sont utilisées dans le dernier script lancé des niveau d'exécution **0** et **6** :

- **halt** : synchronise les disques avec les tampons, met à jour le fichier */var/log/wtmp* (qui contie un historique des utilisateurs connectés, visible à l'aide de la commande **last**) et arrête le système.
- **reboot** : identique à **halt** sauf pour le dernier point où la commande redémarre le système au lieu d l'arrêter.
- **poweroff** : identique à **halt** mais tente en plus d'éteindre l'alimentation électrique de la machine si carte mère de celle-ci est dotée de cette fonctionnalité.

Ces deux dernières commandes sont en fait un lien vers la première. La commande **halt** ne devrait p être lancée directement en ligne de commandes par l'administrateur puisqu'elle ne se charge pas d'arrêt les services ; si tel était le cas, elle ne ferait qu'invoquer la commande **shutdown** avec l'option **-h** (po **halt** et **poweroff**) ou **-r** (pour **reboot**).

c. Autres commandes

Il est aussi possible de procéder à l'arrêt ou au redémarrage du système à partir d'une console virtuelle texte en appuyant sur **[Ctrl]** **[Alt]** **[Suppr]** (redémarrage) et à partir de la console graphique, en cliquant sur le bouton approprié.



D. Gestion d'énergie

1. Onduleurs

Nous avons vu que le fichier */etc/inittab* était doté de quatre directives permettant la gestion d'énergie : **powerfail**, **powerwait**, **powerokwait** et **powerfailnow**.

C'est au programme gérant l'onduleur ou UPS (*Uninterruptible Power System*) d'envoyer le signal **SIGPWR** à Init et de renseigner le fichier */etc/powerstatus* sur l'état de l'alimentation électrique.

Lorsque Init reçoit un signal **SIGPWR**, il regarde dans */etc/powerstatus*.
Si celui-ci contient **FAIL**, il exécute les entrées **powerfail** et **powerwait** du fichier */etc/inittab*. S'il contient **OK**, il exécute l'entrée **powerokwait**. Enfin, si la chaîne de caractères est **LOW**, il exécute l'entrée **powerfailnow**.

Même si les onduleurs les plus sophistiqués peuvent avoir un protocole de communication spécifique avec l'ordinateur, ils sont généralement supportés sous Linux car ils fonctionnent aussi avec un protocole de base commun à tous les onduleurs.

Le paquetage logiciel habituellement utilisé est *apcupsd* mais beaucoup d'autres existent.

2. APM

La plupart des ordinateurs actuels intègrent une gestion d'énergie avancée ou APM *(Advanced Power Mar gement)*.

Il est possible d'utiliser les informations du BIOS sur l'état des accumulateurs ou sur l'activité des périphe ques et du processeur pour optimiser la gestion d'énergie.

L'intérêt est évident sur un ordinateur portable et une collection de programmes permettant la gestion l'alimentation est présente dans le paquetage `apmd`.

Bien que ces fonctions soient aussi disponibles pour des machines autres que des portables, il est gé ralement déconseillé de mettre en œuvre cette gestion d'énergie. Cela peut engendrer des pertes performances, et même dans certains cas, des interruptions de service.