

Dernière mise-à-jour : 2020/01/30 03:27

Gestion du Démarrage et de l'Arrêt du Système

BIOS, EFI et OpenFirmware

Systèmes à base du BIOS

Au démarrage d'un système à base d'un processeur x86 ou x86-64, le premier programme exécuté a été traditionnellement le BIOS. Le BIOS a pour fonction de :

- Tester les composants et les circuits,
- Faire appel au BIOS de la carte graphique pour initialiser le système d'affichage,
- Détecter les périphériques de stockage,
- Lancer le **gestionnaire d'amorçage** du système d'exploitation en utilisant le **bootstrap loader**.

Gestionnaire d'amorçage

La première partie du gestionnaire d'amorçage est en règle générale placé dans le MBR du disque. Le format du MBR est le suivant :

- 446 octets pour le gestionnaire d'amorçage,
- 64 octets pour la table de partitions, soit 16 octets par partition décrite,
- 2 octets ayant une valeur fixe en hexadécimale de **AA55**.



Important : Il est à noter que la première partie du gestionnaire d'amorçage peut également être installé dans un PBR.

Systèmes à base de l'EFI

Depuis 2011, le BIOS est en train d'être remplacé par l'utilisation de l'**UEFI** (**U**nified **E**xtensible **F**irmware **I**nterface ou *Interface micrologicielle extensible unifiée*) issue du développement de l'EFI conçue par Intel pour les processeurs Itanium..

Sous EFI la première partie du gestionnaire de démarrage est un fichier ayant une extension .efi se trouvant dans un sous-répertoire au nom du système d'exploitation à lancer dans une partition appelée **EFI System Partition** ou **ESP**. Cette partition est normalement montée à **/boot/efi** sous Linux.

Pour que EFI fonctionne, le micrologiciel (**firmware**) d'EFI doit avoir connaissance de chaque système d'exploitation à démarrer.

Autres Systèmes

Les systèmes utilisant des processeurs autre qu'un x86 ou x86-64 utilisent un logiciel tel **OpenFirmware**.

Gestionnaires de Démarrage

Des gestionnaires d'amorçage sous Linux, un se distingue comme étant le plus utilisé :

- GRUB (Grand Unified Boot Loader)

Cependant il en existe d'autres :

- LILO (Linux LOader)
 - SysLinux
 - LoadLin
 - ...
-

LILO

LILO (*Linux LOader*) est configuré par le fichier **/etc/lilo.conf**.

La commande LILLO

La commande **lilo** peut prendre une de plusieurs options. Les options les plus importantes sont :

Option	Description
-M	Permet d'écrire sur le MBR
-d	Permet de réduire ou augmenter le temps d'attente avant le lancement du noyau par défaut
-D	Permet de sélectionner un noyau par défaut en indiquant son label
-u	Permet de désinstaller LILLO
-v	Permet d'activer le mode verbose
-m	Permet de modifier le fichier map par défaut (/boot/map)
-i	Permet de spécifier un nouveau fichier à utiliser comme secteur de boot (/boot/boot.b)
-C	Permet de modifier le fichier de configuration par défaut
-q	Permet de créer le fichier /boot/map qui contient l'emplacement des noyaux qui peuvent être booter

Codes Erreur de LILLO

Lors du démarrage, LILLO permet d'identifier les éventuelles erreurs :

Affichage	Erreur
(rien)	Aucun morceau de LILLO n'a été chargé. Soit LILLO n'est pas installé, soit la partition sur laquelle son secteur d'amorce se trouve n'est pas active.
L	Le premier morceau du chargeur d'amorce a été chargé et démarré, mais il ne peut charger le second morceau. Les codes d'erreur à deux chiffres indiquent le type de problème. (Voir également la section "Codes d'erreur disque".) Ce cas indique en général une panne de périphérique ou une incohérence de géométrie (c'est à dire de mauvais paramètres disques).
LI	Le premier morceau du chargeur d'amorce a pu charger le second morceau, mais n'a pas réussi à l'exécuter. Cela peut être causé par une incohérence de géométrie ou par le déplacement de /boot/boot.b sans lancer l'installateur de carte.

Affichage	Erreur
LIL	Le second morceau du chargeur d'amorce a été démarré, mais il ne trouve pas la table de descripteurs dans le fichier carte. C'est en général dû à une panne de périphérique ou une incohérence de géométrie.
LIL?	Le second morceau du chargeur d'amorce a été chargé à un adresse incorrecte. C'est en général causé par une subtile incohérence de géométrie, ou par le déplacement de /boot/boot.b sans lancer l'installateur de carte.
LIL-	La table de descripteurs est corrompue. Cela peut être dû à une incohérence de géométrie ou au déplacement de /boot/map sans lancer l'installeur.
LILO	Tous les éléments de LILO ont été correctement chargés.

Si le BIOS signale une erreur lorsque LILO essaye de charger une image d'amorce, le code d'erreur correspondant est affiché. Ces codes vont de 0x00 à 0xbb. Reportez-vous au Guide Utilisateur de LILO pour leur explication.

Grub 2

Ubuntu utilise le gestionnaire d'amorçage **Grub2**. Grub2 est le successeur de Grub mais ne garde que le nom de ce dernier. En effet, Grub2 est une réécriture complète de Grub.

Dans le cas où le gestionnaire d'amorçage **Grub2** n'est pas installé, il convient de saisir la commande suivante :

```
# grub-install /dev/périphérique [Entrée]
```

où **périphérique** est le nom du périphérique où grub doit s'installer dans le MBR.



Dans le cas où le gestionnaire d'amorçage est **Grub** il est possible de passer à Grub2 en utilisant la commande suivante :

```
# upgrade-from-grub-legacy [Entrée]
```

Les fichiers de modules de Grub2, lui permettant de gérer le démarrage sur un nombre important de système de fichiers, se trouvent dans le répertoire **/boot/grub/i386-pc** :

```

root@ubuntu:~# ls /boot/grub
fonts  gfxblacklist.txt  grub.cfg  grubenv  i386-pc  locale
root@ubuntu:~# ls /boot/grub/i386-pc/
915resolution.mod  datetime.mod  gdb.mod  mdraid1x.mod  pbkdf2.mod
tftp.mod
acpi.mod            diskfilter.mod  geli.mod  memdisk.mod  pbkdf2_test.mod
tga.mod
adler32.mod         disk.mod        gettext.mod  memrw.mod  pcidump.mod
time.mod
affs.mod            div_test.mod    gfxmenu.mod  minicmd.mod  pci.mod
trig.mod
afs.mod             dm_nv.mod       gfxterm_background.mod  minix2_be.mod  plan9.mod
tr.mod
ahci.mod            drivemap.mod    gfxterm_menu.mod  minix2.mod    play.mod
truecrypt.mod
all_video.mod       echo.mod        gfxterm.mod      minix3_be.mod  png.mod
true.mod
aout.mod            efiemu32.o      gptsync.mod      minix3.mod     priority_queue.mod
udf.mod
archelp.mod         efiemu64.o      gzio.mod          minix_be.mod   probe.mod
ufs1_be.mod
ata.mod             efiemu.mod      halt.mod          minix.mod       procfs.mod
ufs1.mod
at_keyboard.mod     ehci.mod        hashsum.mod       mmap.mod        progress.mod
ufs2.mod
backtrace.mod       elf.mod         hdparm.mod        moddep.lst      pxechain.mod
uhci.mod
bfs.mod             eval.mod        hello.mod          modinfo.sh      pxe.mod
usb_keyboard.mod    exfat.mod       help.mod          morse.mod       raid5rec.mod
biosdisk.mod        exfctest.mod    hexdump.mod       mpi.mod         raid6rec.mod
usb.mod
bitmap.mod          exfat.mod       hfs.mod           msdospart.mod   read.mod
usbms.mod
bitmap_scale.mod    ext2.mod

```

usbserial_common.mod				
blocklist.mod	extcmd.mod	hfspluscomp.mod	multiboot2.mod	reboot.mod
usbserial_ftdi.mod				
boot.img	fat.mod	hfsplus.mod	multiboot.mod	regex.mod
usbserial_pl2303.mod				
boot.mod	file.mod	http.mod	natedisk.mod	reiserfs.mod
usbserial_usbdebug.mod				
bsd.mod	font.mod	hwmatch.mod	net.mod	relocator.mod
usbtest.mod				
btrfs.mod	freedos.mod	iorw.mod	newc.mod	romfs.mod
vbe.mod				
bufio.mod	fshelp.mod	iso9660.mod	nilfs2.mod	scsi.mod
verify.mod				
cat.mod	fs.lst	jfs.mod	normal.mod	search_fs_file.mod
vga.mod				
cbfs.mod	functional_test.mod	jpeg.mod	ntfscomp.mod	search_fs_uuid.mod
vga_text.mod				
cbfs.mod	gcry_arcfour.mod	keylayouts.mod	ntfs.mod	search_label.mod
video_bochs.mod				
cbmemc.mod	gcry_blowfish.mod	keystatus.mod	ntldr.mod	search.mod
video_cirrus.mod				
cbtable.mod	gcry_camellia.mod	ldm.mod	odc.mod	sendkey.mod
video_colors.mod				
cbtime.mod	gcry_cast5.mod	legacycfg.mod	offsetio.mod	serial.mod
video_fb.mod				
chain.mod	gcry_crc.mod	legacy_password_test.mod	ohci.mod	setjmp.mod
videoinfo.mod				
cmdline_cat_test.mod	gcry_des.mod	linux16.mod	part_acorn.mod	setjmp_test.mod
video.lst				
cmosdump.mod	gcry_dsa.mod	linux.mod	part_amiga.mod	setpci.mod
video.mod				
cmostest.mod	gcry_idea.mod	loadenv.mod	part_apple.mod	sfs.mod
videotest_checksum.mod				
cmp.mod	gcry_md4.mod	loopback.mod	part_bsd.mod	signature_test.mod

videotest.mod				
command.lst	gcry_md5.mod	lsacpi.mod	part_dfly.mod	sleep.mod
xfs.mod				
configfile.mod	gcry_rfc2268.mod	lsapm.mod	part_dvh.mod	sleep_test.mod
xnu.mod				
core.img	gcry_rijndael.mod	lsmmap.mod	part_gpt.mod	spkmodem.mod
xnu_uuid.mod				
cpio_be.mod	gcry_rmd160.mod	ls.mod	partmap.lst	squash4.mod
xnu_uuid_test.mod				
cpio.mod	gcry_rsa.mod	lspci.mod	part_msdos.mod	syslinuxcfg.mod
xzio.mod				
cpuid.mod	gcry_seed.mod	luks.mod	part_plan.mod	tar.mod
zfsencrypt.mod				
crc64.mod	gcry_serpent.mod	lvm.mod	part_sun.mod	terminal.lst
zfsinfo.mod				
cryptodisk.mod	gcry_sha1.mod	lzopio.mod	part_sunpc.mod	terminal.mod
zfs.mod				
crypto.lst	gcry_sha256.mod	macbless.mod	parttool.lst	terminfo.mod
crypto.mod	gcry_sha512.mod	macho.mod	parttool.mod	test_blockarg.mod
cs5536.mod	gcry_tiger.mod	mda_text.mod	password.mod	testload.mod
datehook.mod	gcry_twofish.mod	mdraid09_be.mod	password_pbkdf2.mod	test.mod
date.mod	gcry_whirlpool.mod	mdraid09.mod	pata.mod	testspeed.mod

Grub2 lit ses entrées de menus à partir du fichier **/boot/grub/grub.cfg**. Pour visualiser ce fichier, il convient de saisir la commande suivante :

```
root@ubuntu:~# cat /boot/grub/grub.cfg
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/00_header ###
```

```
if [ -s $prefix/grubenv ]; then
    set have_grubenv=true
    load_env
fi
if [ "${next_entry}" ] ; then
    set default="${next_entry}"
    set next_entry=
    save_env next_entry
    set boot_once=true
else
    set default="0"
fi

if [ x"${feature_menuentry_id}" = xy ]; then
    menuentry_id_option="--id"
else
    menuentry_id_option=""
fi

export menuentry_id_option

if [ "${prev_saved_entry}" ]; then
    set saved_entry="${prev_saved_entry}"
    save_env saved_entry
    set prev_saved_entry=
    save_env prev_saved_entry
    set boot_once=true
fi

function savedefault {
    if [ -z "${boot_once}" ]; then
        saved_entry="${chosen}"
        save_env saved_entry
    fi
}
```

```
}
function recordfail {
    set recordfail=1
    if [ -n "${have_grubenv}" ]; then if [ -z "${boot_once}" ]; then save_env recordfail; fi; fi
}
function load_video {
    if [ x$feature_all_video_module = xy ]; then
        insmod all_video
    else
        insmod efi_gop
        insmod efi_uga
        insmod ieee1275_fb
        insmod vbe
        insmod vga
        insmod video_bochs
        insmod video_cirrus
    fi
}

if [ x$feature_default_font_path = xy ] ; then
    font=unicode
else
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    font="/usr/share/grub/unicode.pf2"
fi
```

```
if loadfont $font ; then
    set gfxmode=auto
    load_video
    insmod gfxterm
    set locale_dir=$prefix/locale
    set lang=fr_FR
    insmod gettext
fi
terminal_output gfxterm
if [ "${recordfail}" = 1 ] ; then
    set timeout=-1
else
    if [ x$feature_timeout_style = xy ] ; then
        set timeout_style=hidden
        set timeout=0
        # Fallback hidden-timeout code in case the timeout_style feature is
        # unavailable.
        elif sleep --interruptible 0 ; then
            set timeout=0
        fi
    fi
fi
### END /etc/grub.d/00_header ###

### BEGIN /etc/grub.d/05_debian_theme ###
set menu_color_normal=white/black
set menu_color_highlight=black/light-gray
if background_color 44,0,30; then
    clear
fi
### END /etc/grub.d/05_debian_theme ###

### BEGIN /etc/grub.d/10_linux ###
function gfxmode {
    set gfxpayload="${1}"
```

```
    if [ "${1}" = "keep" ]; then
        set vt_handoff=vt.handoff=7
    else
        set vt_handoff=
    fi
}
if [ "${recordfail}" != 1 ]; then
    if [ -e ${prefix}/gfxblacklist.txt ]; then
        if hwmatch ${prefix}/gfxblacklist.txt 3; then
            if [ ${match} = 0 ]; then
                set linux_gfx_mode=keep
            else
                set linux_gfx_mode=text
            fi
        else
            set linux_gfx_mode=text
        fi
    else
        set linux_gfx_mode=keep
    fi
else
    set linux_gfx_mode=text
fi
export linux_gfx_mode
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-
simple-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
    recordfail
    load_video
    gfxmode $linux_gfx_mode
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
```

```
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    linux /boot/vmlinuz-3.13.0-32-generic root=UUID=70eb8bc5-1759-433d-9797-9342a7b82cb2 ro quiet splash
$vt_handoff
    initrd /boot/initrd.img-3.13.0-32-generic
}
submenu 'Options avancées pour Ubuntu' $menuentry_id_option 'gnulinux-
advanced-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
    menuentry 'Ubuntu, avec Linux 3.13.0-32-generic' --class ubuntu --class gnu-linux --class gnu --class os
$menuentry_id_option 'gnulinux-3.13.0-32-generic-advanced-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
            search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
        else
            search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
        fi
        echo 'Chargement de Linux 3.13.0-32-generic...'
        linux /boot/vmlinuz-3.13.0-32-generic root=UUID=70eb8bc5-1759-433d-9797-9342a7b82cb2 ro quiet splash
$vt_handoff
        echo 'Chargement du disque mémoire initial...'
        initrd /boot/initrd.img-3.13.0-32-generic
    }
    menuentry 'Ubuntu, with Linux 3.13.0-32-generic (recovery mode)' --class ubuntu --class gnu-linux --class gnu
--class os $menuentry_id_option 'gnulinux-3.13.0-32-generic-recovery-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
```

```
    recordfail
    load_video
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    echo      'Chargement de Linux 3.13.0-32-generic...'
    linux     /boot/vmlinuz-3.13.0-32-generic root=UUID=70eb8bc5-1759-433d-9797-9342a7b82cb2 ro recovery
nomodeset
    echo      'Chargement du disque mémoire initial...'
    initrd    /boot/initrd.img-3.13.0-32-generic
}
}

### END /etc/grub.d/10_linux ###

### BEGIN /etc/grub.d/20_linux_xen ###

### END /etc/grub.d/20_linux_xen ###

### BEGIN /etc/grub.d/20_memtest86+ ###
menuentry 'Memory test (memtest86+)' {
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
```

```
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    knetbsd      /boot/memtest86+.elf
}
menuentry 'Memory test (memtest86+, serial console 115200)' {
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    linux16      /boot/memtest86+.bin console=ttyS0,115200n8
}
### END /etc/grub.d/20_memtest86+ ###

### BEGIN /etc/grub.d/30_os-prober ###
### END /etc/grub.d/30_os-prober ###

### BEGIN /etc/grub.d/30_uefi-firmware ###
### END /etc/grub.d/30_uefi-firmware ###

### BEGIN /etc/grub.d/40_custom ###
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
### END /etc/grub.d/40_custom ###

### BEGIN /etc/grub.d/41_custom ###
if [ -f ${config_directory}/custom.cfg ]; then
    source ${config_directory}/custom.cfg
```

```
elif [ -z "${config_directory}" -a -f $prefix/custom.cfg ]; then
    source $prefix/custom.cfg;
fi
### END /etc/grub.d/41_custom ###
```



Notez que ce fichier ne doit pas être modifié manuellement. En effet, il est généré par la commande **update-grub**.

Lors de l'exécution de la commande **update-grub**, plusieurs fichiers sont lus :

Le fichier `/etc/default/grub`

Ce fichier contient la configuration par défaut des paramètres de Grub2 :

```
root@ubuntu:~# cat /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
```

```
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
[3]+  Fini                nice -n 19 sleep 5678
```



Notez que toute modification de ce fichier nécessite l'exécution de la commande **update-grub** pour que les modifications soient prises en compte.

Dans ce fichier les directives sont :

Directive	Description
GRUB_DEFAULT	Entrée du menu sélectionner par défaut
GRUB_TIMEOUT	Durée de l'affichage du menu avant le démarrage en utilisant la valeur de GRUB_DEFAULT
GRUB_DISTRIBUTOR	Ligne de commande qui génère le texte de l'entrée
GRUB_CMDLINE_LINUX_DEFAULT	Paramètres passés au noyau lors d'un démarrage normal (Hors donc le mode secours)

Directive	Description
GRUB_CMDLINE_LINUX	Paramètres passés au noyau peu importe le type de démarrage
GRUB_TERMINAL	Si non commentée, cette directive désactive le démarrage graphique
GRUB_GFXMODE	Indique la résolution utilisée lors d'un démarrage graphique
GRUB_DISABLE_LINUX_UUID	Si true , cette directive empêche l'utilisation de l'UUID de la partition
GRUB_DISABLE_LINUX_RECOVERY	Si true , cette directive empêche la génération des entrées en mode recovery
GRUB_INIT_TUNE	Permet d'obtenir un beep au démarrage de Grub2
GRUB_BADRAM	Permet de spécifier de la mémoire défectueuse

Les fichiers du répertoire /etc/grub.d

Les fichiers de ce répertoire sont exécutés dans l'ordre alphanumérique et servent à construire les menus de Grub2 :

```
root@ubuntu:~# ls -l /etc/grub.d
total 76
-rwxr-xr-x 1 root root 9424 mai 15 21:01 00_header
-rwxr-xr-x 1 root root 6058 mai 8 14:08 05_debian_theme
-rwxr-xr-x 1 root root 11608 mai 15 21:01 10_linux
-rwxr-xr-x 1 root root 10412 mai 15 21:01 20_linux_xen
-rwxr-xr-x 1 root root 1992 mars 12 2014 20_memtest86+
-rwxr-xr-x 1 root root 11692 mai 15 21:01 30_os-prober
-rwxr-xr-x 1 root root 1416 mai 15 21:01 30_uefi-firmware
-rwxr-xr-x 1 root root 214 mai 15 21:01 40_custom
-rwxr-xr-x 1 root root 216 mai 15 21:01 41_custom
-rw-r--r-- 1 root root 483 mai 15 21:01 README
```

Le fichier /etc/grub.d/10_Linux

Le fichier **10_Linux** contient des boucles pour rechercher des noyaux Linux :

```
root@ubuntu:~# cat /etc/grub.d/10_linux
```

```
#!/bin/sh
set -e

# grub-mkconfig helper script.
# Copyright (C) 2006,2007,2008,2009,2010 Free Software Foundation, Inc.
#
# GRUB is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# GRUB is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GRUB. If not, see <http://www.gnu.org/licenses/>.

prefix="/usr"
exec_prefix="/usr"
datarootdir="/usr/share"
ubuntu_recovery="1"
quiet_boot="1"
quick_boot="1"
gfxpayload_dynamic="1"
vt_handoff="1"

. "${datarootdir}/grub/grub-mkconfig_lib"

export TEXTDOMAIN=grub
export TEXTDOMAINDIR="${datarootdir}/locale"

CLASS="--class gnu-linux --class gnu --class os"
```

```
if [ "x${GRUB_DISTRIBUTOR}" = "x" ] ; then
    OS=GNU/Linux
else
    case ${GRUB_DISTRIBUTOR} in
        Ubuntu|Kubuntu)
            OS="${GRUB_DISTRIBUTOR}"
            ;;
        *)
            OS="${GRUB_DISTRIBUTOR} GNU/Linux"
            ;;
    esac
    CLASS="--class $(echo ${GRUB_DISTRIBUTOR} | tr 'A-Z' 'a-z' | cut -d' ' -f1|LC_ALL=C sed 's,[^[:alnum:]]_,_,g')
${CLASS}"
fi

# loop-AES arranges things so that /dev/loop/X can be our root device, but
# the initrds that Linux uses don't like that.
case ${GRUB_DEVICE} in
    /dev/loop/*|/dev/loop[0-9])
        GRUB_DEVICE=`losetup ${GRUB_DEVICE} | sed -e "s/^[^()]*([^(])\+\\.*/\\1/"`
        # We can't cope with devices loop-mounted from files here.
        case ${GRUB_DEVICE} in
            /dev/*) ;;
            *) exit 0 ;;
        esac
    ;;
esac

if [ "x${GRUB_DEVICE_UUID}" = "x" ] || [ "x${GRUB_DISABLE_LINUX_UUID}" = "xtrue" ] \
    || ! test -e "/dev/disk/by-uuid/${GRUB_DEVICE_UUID}" \
    || uses_abstraction "${GRUB_DEVICE}" lvm; then
    LINUX_ROOT_DEVICE=${GRUB_DEVICE}
else
    LINUX_ROOT_DEVICE=UUID=${GRUB_DEVICE_UUID}
```

```
fi

case x"$GRUB_FS" in
  xtrfs)
    rootsubvol="`make_system_path_relative_to_its_root /\`"
    rootsubvol="${rootsubvol#/}"
    if [ "x${rootsubvol}" != x ]; then
      GRUB_CMDLINE_LINUX="rootflags=subvol=${rootsubvol} ${GRUB_CMDLINE_LINUX}"
    fi;;
  xzfs)
    rpool=`${grub_probe} --device ${GRUB_DEVICE} --target=fs_label 2>/dev/null || true`
    bootfs="`make_system_path_relative_to_its_root / | sed -e "s,@$,,\"`"
    LINUX_ROOT_DEVICE="ZFS=${rpool}${bootfs}"
    ;;
esac

title_correction_code=

if [ -x /lib/recovery-mode/recovery-menu ]; then
  GRUB_CMDLINE_LINUX_RECOVERY=recovery
else
  GRUB_CMDLINE_LINUX_RECOVERY=single
fi
if [ "$ubuntu_recovery" = 1 ]; then
  GRUB_CMDLINE_LINUX_RECOVERY="$GRUB_CMDLINE_LINUX_RECOVERY nomodeset"
fi

if [ "$vt_handoff" = 1 ]; then
  for word in $GRUB_CMDLINE_LINUX_DEFAULT; do
    if [ "$word" = splash ]; then
      GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT \$vt_handoff"
    fi
  done
fi
```

```
linux_entry ()
{
    os="$1"
    version="$2"
    type="$3"
    args="$4"

    if [ -z "$boot_device_id" ]; then
        boot_device_id="$(grub_get_device_id "${GRUB_DEVICE}")"
    fi
    if [ x$type != xsimple ] ; then
        case $type in
            recovery)
                title="$(gettext_printf "%s, with Linux %s (%s)" "${os}" "${version}" "$(gettext
"${GRUB_RECOVERY_TITLE}")")" ;;
            *)
                title="$(gettext_printf "%s, with Linux %s" "${os}" "${version}")" ;;
        esac
        if [ x"$title" = x"$GRUB_ACTUAL_DEFAULT" ] || [ x"Previous Linux versions>$title" = x"$GRUB_ACTUAL_DEFAULT"
]; then
            replacement_title="$(echo "Advanced options for ${OS}" | sed 's,>,>>,g')>$(echo "$title" | sed 's,>,>>,g')
            quoted="$(echo "$GRUB_ACTUAL_DEFAULT" | grub_quote)"
            title_correction_code="${title_correction_code}if [ \"x\\$default\" = '$quoted' ]; then default='$(echo
\"$replacement_title\" | grub_quote)'; fi;"
            grub_warn "$(gettext_printf "Please don't use old title \"\`%s\" for GRUB_DEFAULT, use \"\`%s\" (for versions
before 2.00) or \"\`%s\" (for 2.00 or later)" "$GRUB_ACTUAL_DEFAULT" "$replacement_title" "gnulinux-advanced-
$boot_device_id>gnulinux-$version-$type-$boot_device_id")"
        fi
        echo "menuentry '$(echo "$title" | grub_quote)' ${CLASS} \${menuentry_id_option 'gnulinux-$version-$type-
$boot_device_id' {' | sed 's/^/$submenu_indentation/'
    else
        echo "menuentry '$(echo "$os" | grub_quote)' ${CLASS} \${menuentry_id_option 'gnulinux-simple-
$boot_device_id' {' | sed 's/^/$submenu_indentation/'
    fi
}
```

```
if [ "$quick_boot" = 1 ]; then
    echo "    recordfail" | sed "s/^/$submenu_indentation/"
fi
if [ x$type != xrecovery ] ; then
    save_default_entry | grub_add_tab
fi

# Use ELILO's generic "efifb" when it's known to be available.
# FIXME: We need an interface to select vesafb in case efifb can't be used.
if [ "x$GRUB_GFXPAYLOAD_LINUX" = x ]; then
    echo "    load_video" | sed "s/^/$submenu_indentation/"
else
    if [ "x$GRUB_GFXPAYLOAD_LINUX" != xtext ]; then
        echo "    load_video" | sed "s/^/$submenu_indentation/"
    fi
fi
if ([ "$ubuntu_recovery" = 0 ] || [ x$type != xrecovery ]) && \
    ([ "x$GRUB_GFXPAYLOAD_LINUX" != x ] || [ "$gfxpayload_dynamic" = 1 ]); then
    echo "    gfxmode \${linux_gfx_mode}" | sed "s/^/$submenu_indentation/"
fi

echo "    insmod gzio" | sed "s/^/$submenu_indentation/"

if [ x$dirname = x/ ]; then
    if [ -z "${prepare_root_cache}" ]; then
        prepare_root_cache="$(prepare_grub_to_access_device ${GRUB_DEVICE} | grub_add_tab)"
    fi
    printf '%s\n' "${prepare_root_cache}" | sed "s/^/$submenu_indentation/"
else
    if [ -z "${prepare_boot_cache}" ]; then
        prepare_boot_cache="$(prepare_grub_to_access_device ${GRUB_DEVICE_BOOT} | grub_add_tab)"
    fi
    printf '%s\n' "${prepare_boot_cache}" | sed "s/^/$submenu_indentation/"
fi
```

```
if [ x"$quiet_boot" = x0 ] || [ x"$type" != xsimple ]; then
    message="$(gettext_printf "Loading Linux %s ..." "${version}")"
    sed "s/^/$submenu_indentation/" << EOF
    echo    '$(echo "$message" | grub_quote)'
EOF
fi
if test -d /sys/firmware/efi && test -e "${linux}.efi.signed"; then
    sed "s/^/$submenu_indentation/" << EOF
    linux    ${rel_dirname}/${basename}.efi.signed root=${linux_root_device_thisversion} ro ${args}
EOF
else
    sed "s/^/$submenu_indentation/" << EOF
    linux    ${rel_dirname}/${basename} root=${linux_root_device_thisversion} ro ${args}
EOF
fi
if test -n "${initrd}" ; then
    # TRANSLATORS: ramdisk isn't identifier. Should be translated.
    if [ x"$quiet_boot" = x0 ] || [ x"$type" != xsimple ]; then
        message="$(gettext_printf "Loading initial ramdisk ...")"
        sed "s/^/$submenu_indentation/" << EOF
        echo    '$(echo "$message" | grub_quote)'
EOF
    fi
    sed "s/^/$submenu_indentation/" << EOF
    initrd    ${rel_dirname}/${initrd}
EOF
fi
sed "s/^/$submenu_indentation/" << EOF
}
EOF
}

machine=`uname -m`
case "x$machine" in
```

```
xi?86 | xx86_64)
list=`for i in /boot/vmlinuz-* /vmlinuz-* /boot/kernel-* ; do
    if grub_file_is_not_garbage "$i" ; then echo -n "$i " ; fi
done` ;;
*)
list=`for i in /boot/vmlinuz-* /boot/vmlinux-* /vmlinuz-* /vmlinux-* /boot/kernel-* ; do
    if grub_file_is_not_garbage "$i" ; then echo -n "$i " ; fi
done` ;;
esac

case "$machine" in
i?86) GENKERNEL_ARCH="x86" ;;
mips|mips64) GENKERNEL_ARCH="mips" ;;
mipsel|mips64el) GENKERNEL_ARCH="mipsel" ;;
arm*) GENKERNEL_ARCH="arm" ;;
*) GENKERNEL_ARCH="$machine" ;;
esac

prepare_boot_cache=
prepare_root_cache=
boot_device_id=
title_correction_code=

cat << 'EOF'
function gfxmode {
    set gfxpayload="${1}"
EOF
if [ "$vt_handoff" = 1 ]; then
    cat << 'EOF'
    if [ "${1}" = "keep" ]; then
        set vt_handoff=vt.handoff=7
    else
        set vt_handoff=
    fi
fi
```

```
EOF
fi
cat << EOF
}
EOF

# Use ELILO's generic "efifb" when it's known to be available.
# FIXME: We need an interface to select vesafb in case efifb can't be used.
if [ "x$GRUB_GFXPAYLOAD_LINUX" != x ] || [ "$gfxpayload_dynamic" = 0 ]; then
    echo "set linux_gfx_mode=$GRUB_GFXPAYLOAD_LINUX"
else
    cat << EOF
    if [ "\${recordfail}" != 1 ]; then
        if [ -e \${prefix}/gfxblacklist.txt ]; then
            if hwmatch \${prefix}/gfxblacklist.txt 3; then
                if [ \${match} = 0 ]; then
                    set linux_gfx_mode=keep
                else
                    set linux_gfx_mode=text
                fi
            else
                set linux_gfx_mode=text
            fi
        else
            set linux_gfx_mode=keep
        fi
    else
        set linux_gfx_mode=text
    fi
    EOF
fi
cat << EOF
export linux_gfx_mode
EOF
```

```
# Extra indentation to add to menu entries in a submenu. We're not in a submenu
# yet, so it's empty. In a submenu it will be equal to '\t' (one tab).
submenu_indentation=""

is_top_level=true
while [ "x$list" != "x" ] ; do
    linux=`version_find_latest $list`
    case $linux in
        *.efi.signed)
            # We handle these in linux_entry.
            list=`echo $list | tr ' ' '\n' | grep -vx $linux | tr '\n' ' '`
            continue
        ;;
    esac
    gettext_printf "Found linux image: %s\n" "$linux" >&2
    basename=`basename $linux`
    dirname=`dirname $linux`
    rel_dirname=`make_system_path_relative_to_its_root $dirname`
    version=`echo $basename | sed -e "s,^[^0-9]*-,g"`
    alt_version=`echo $version | sed -e "s,\.old$,,g"`
    linux_root_device_thisversion="${LINUX_ROOT_DEVICE}"

    initrd=
    for i in "initrd.img-${version}" "initrd-${version}.img" "initrd-${version}.gz" \
        "initrd-${version}" "initramfs-${version}.img" \
        "initrd.img-${alt_version}" "initrd-${alt_version}.img" \
        "initrd-${alt_version}" "initramfs-${alt_version}.img" \
        "initramfs-genkernel-${version}" \
        "initramfs-genkernel-${alt_version}" \
        "initramfs-genkernel-${GENKERNEL_ARCH}-${version}" \
        "initramfs-genkernel-${GENKERNEL_ARCH}-${alt_version}"; do
        if test -e "${dirname}/${i}" ; then
            initrd="$i"
            break
        fi
    done
done
```

```
    fi
done

config=
for i in "${dirname}/config-${version}" "${dirname}/config-${alt_version}" "/etc/ernels/kernel-config-${version}" ; do
    if test -e "${i}" ; then
        config="${i}"
        break
    fi
done

initramfs=
if test -n "${config}" ; then
    initramfs=`grep CONFIG_INITRAMFS_SOURCE= "${config}" | cut -f2 -d= | tr -d \"`
fi

if test -n "${initrd}" ; then
    gettext_printf "Found initrd image: %s\n" "${dirname}/${initrd}" >&2
elif test -z "${initramfs}" ; then
    # "UUID=" and "ZFS=" magic is parsed by initrd or initramfs. Since there's
    # no initrd or builtin initramfs, it can't work here.
    linux_root_device_thisversion=${GRUB_DEVICE}
fi

if [ "x$is_top_level" = xtrue ] && [ "x${GRUB_DISABLE_SUBMENU}" != xy ]; then
    linux_entry "${OS}" "${version}" simple \
    "${GRUB_CMDLINE_LINUX} ${GRUB_CMDLINE_LINUX_DEFAULT}"

    submenu_indentation="$grub_tab"
    if [ -z "$boot_device_id" ]; then
        boot_device_id="$(grub_get_device_id "${GRUB_DEVICE}")"
    fi
    # TRANSLATORS: %s is replaced with an OS name
```

```
    echo "submenu '$(gettext_printf "Advanced options for %s" "${OS}" | grub_quote)' \${menuentry_id_option
'gnulinux-advanced-$boot_device_id' {"
    is_top_level=false
    fi

    linux_entry "${OS}" "${version}" advanced \
        "${GRUB_CMDLINE_LINUX} ${GRUB_CMDLINE_LINUX_DEFAULT}"
    if [ "x${GRUB_DISABLE_RECOVERY}" != "xtrue" ]; then
        linux_entry "${OS}" "${version}" recovery \
            "${GRUB_CMDLINE_LINUX_RECOVERY} ${GRUB_CMDLINE_LINUX}"
    fi

    list=`echo $list | tr ' ' '\n' | grep -vx $linux | tr '\n' ' '`
done

# If at least one kernel was found, then we need to
# add a closing '}' for the submenu command.
if [ x"$is_top_level" != xtrue ]; then
    echo '}'
fi

echo "$title_correction_code"
```

Le fichier `/etc/grub.d/30_os-prober`

Ce fichier recherche des éventuels systèmes d'exploitation autre que Linux :

```
root@ubuntu:~# cat /etc/grub.d/30_os-prober
#!/bin/sh
set -e

# grub-mkconfig helper script.
# Copyright (C) 2006,2007,2008,2009 Free Software Foundation, Inc.
```

```
#
# GRUB is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# GRUB is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GRUB. If not, see <http://www.gnu.org/licenses/>.

prefix="/usr"
exec_prefix="/usr"
datarootdir="/usr/share"
quick_boot="1"

export TEXTDOMAIN=grub
export TEXTDOMAINDIR="${datarootdir}/locale"

. "${datarootdir}/grub/grub-mkconfig_lib"

found_other_os=

adjust_timeout () {
  if [ "$quick_boot" = 1 ] && [ "x${found_other_os}" != "x" ]; then
    cat << EOF
  set timeout_style=menu
  if [ "\${timeout}" = 0 ]; then
    set timeout=10
  fi
EOF
```

```
fi
}

if [ "x${GRUB_DISABLE_OS_PROBER}" = "xtrue" ]; then
    exit 0
fi

if [ -z "`which os-prober 2> /dev/null`" -o -z "`which linux-boot-prober 2> /dev/null`" ] ; then
    # missing os-prober and/or linux-boot-prober
    exit 0
fi

OSPROBED="`os-prober | tr ' ' '^' | paste -s -d ' '`"
if [ -z "${OSPROBED}" ] ; then
    # empty os-prober output, nothing doing
    exit 0
fi

osx_entry() {
    found_other_os=1
    if [ x$2 = x32 ]; then
        # TRANSLATORS: it refers to kernel architecture (32-bit)
        bitstr="$(gettext "(32-bit)")"
    else
        # TRANSLATORS: it refers to kernel architecture (64-bit)
        bitstr="$(gettext "(64-bit)")"
    fi
    # TRANSLATORS: it refers on the OS residing on device %s
    onstr="$(gettext_printf "(on %s)" "${DEVICE}")"
    cat << EOF
menuentry '$(echo "${LONGNAME} $bitstr $onstr" | grub_quote)' --class osx --class darwin --class os
\${menuentry_id_option 'osprober-xnu-$2-${grub_get_device_id "${DEVICE}"}' {
EOF
    save_default_entry | grub_add_tab
```

```
prepare_grub_to_access_device ${DEVICE} | grub_add_tab
cat << EOF
load_video
set do_resume=0
if [ /var/vm/sleepimage -nt10 / ]; then
    if xnu_resume /var/vm/sleepimage; then
        set do_resume=1
    fi
fi
if [ \do_resume = 0 ]; then
    xnu_uuid ${OSXUUID} uuid
    if [ -f /Extra/DSDT.aml ]; then
        acpi -e /Extra/DSDT.aml
    fi
    if [ /kernelcache -nt /System/Library/Extensions ]; then
        $1 /kernelcache boot-uuid=\${uuid} rd=*uuid
    else
        $1 /mach_kernel boot-uuid=\${uuid} rd=*uuid
        if [ /System/Library/Extensions.mkext -nt /System/Library/Extensions ]; then
            xnu_mkext /System/Library/Extensions.mkext
        else
            xnu_kextdir /System/Library/Extensions
        fi
    fi
    if [ -f /Extra/Extensions.mkext ]; then
        xnu_mkext /Extra/Extensions.mkext
    fi
    if [ -d /Extra/Extensions ]; then
        xnu_kextdir /Extra/Extensions
    fi
    if [ -f /Extra/devprop.bin ]; then
        xnu_devprop_load /Extra/devprop.bin
    fi
    if [ -f /Extra/splash.jpg ]; then
```

```
        insmod jpeg
        xnu_splash /Extra/splash.jpg
    fi
    if [ -f /Extra/splash.png ]; then
        insmod png
        xnu_splash /Extra/splash.png
    fi
    if [ -f /Extra/splash.tga ]; then
        insmod tga
        xnu_splash /Extra/splash.tga
    fi
fi
}
EOF
}

used_osprober_linux_ids=

wubi=

for OS in ${OSPROBED} ; do
    DEVICE="`echo ${OS} | cut -d ':' -f 1`"
    LONGNAME="`echo ${OS} | cut -d ':' -f 2 | tr '^' ' '`"
    LABEL="`echo ${OS} | cut -d ':' -f 3 | tr '^' ' '`"
    BOOT="`echo ${OS} | cut -d ':' -f 4`"
    if UUID="`${grub_probe} --target=fs_uuid --device ${DEVICE%*}`"; then
        EXPUUID="$UUID"

        if [ x"${DEVICE#*}" != x ] ; then
            EXPUUID="${EXPUUID}@${DEVICE#*}"
        fi

        if [ "x${GRUB_OS_PROBER_SKIP_LIST}" != "x" -a "x`echo ${GRUB_OS_PROBER_SKIP_LIST} | grep -i -e
        '\b'${EXPUUID}'\b'`" != "x" ] ; then
```

```
    echo "Skipped ${LONGNAME} on ${DEVICE} by user request." >&2
    continue
fi
fi

BTRFS="`echo ${OS} | cut -d ':' -f 5`"
if [ "x$BTRFS" = "xbtrfs" ]; then
    BTRFSsuid="`echo ${OS} | cut -d ':' -f 6`"
    BTRFSsubvol="`echo ${OS} | cut -d ':' -f 7`"
fi

if [ -z "${LONGNAME}" ] ; then
    LONGNAME="${LABEL}"
fi

gettext_printf "Found %s on %s\n" "${LONGNAME}" "${DEVICE}" >&2

case ${BOOT} in
    chain)

        case ${LONGNAME} in
            Windows*)
                if [ -z "$wubi" ]; then
                    if [ -x /usr/share/lupin-support/grub-mkimage ] && \
                        /usr/share/lupin-support/grub-mkimage --test; then
                        wubi=yes
                    else
                        wubi=no
                    fi
                fi
                if [ "$wubi" = yes ]; then
                    echo "Skipping ${LONGNAME} on Wubi system" >&2
                    continue
                fi
            fi
        fi
    fi
fi
```

```
;;
esac

found_other_os=1
onstr="$(gettext_printf "(on %s)" "${DEVICE}")"
cat << EOF
menuentry '${(echo "${LONGNAME} $onstr" | grub_quote)}' --class windows --class os \${menuentry_id_option 'osprober-
chain-${grub_get_device_id "${DEVICE}')} {
EOF
    save_default_entry | grub_add_tab
    prepare_grub_to_access_device ${DEVICE} | grub_add_tab

    if [ x"${grub_probe} --device ${DEVICE} --target=partmap`" = xmsdos ]; then
        cat << EOF
parttool \${root} hidden-
EOF
        fi

        case ${LONGNAME} in
Windows\ Vista*|Windows\ 7*|Windows\ Server\ 2008*)
;;
*)
cat << EOF
drivemap -s (hd0) \${root}
EOF
;;
esac

cat <<EOF
chainloader +1
}
EOF

;;
efi)
```

```

    found_other_os=1
    EFIPATH=${DEVICE#* @}
    DEVICE=${DEVICE% *}
    onstr="$(gettext_printf "(on %s)" "${DEVICE}")"
    cat << EOF
menuentry '$(echo "${LONGNAME} $onstr" | grub_quote)' --class windows --class os \
  $menuentry_id_option 'osprober-efi-${grub_get_device_id "${DEVICE}"}' {
EOF
    save_default_entry | sed -e "s/^/\t/"
    prepare_grub_to_access_device ${DEVICE} | sed -e "s/^/\t/"

    cat << EOF
chainloader ${EFIPATH}
}
EOF
;;
linux)
    if [ "x$BTRFS" = "xbtrfs" ]; then
        LINUXPROBED="\`linux-boot-prober btrfs ${BTRFSuuid} ${BTRFSsubvol} 2> /dev/null | tr ' ' '^' | paste -s
-d ' '`"
    else
        LINUXPROBED="\`linux-boot-prober ${DEVICE} 2> /dev/null | tr ' ' '^' | paste -s -d ' '`"
    fi
    prepare_boot_cache=
    boot_device_id=
    is_top_level=true
    title_correction_code=
    OS="${LONGNAME}"

    for LINUX in ${LINUXPROBED} ; do
        LROOT="\`echo ${LINUX} | cut -d ':' -f 1`"
        LBOOT="\`echo ${LINUX} | cut -d ':' -f 2`"
        LLABEL="\`echo ${LINUX} | cut -d ':' -f 3 | tr '^' ' '`"
        LKERNEL="\`echo ${LINUX} | cut -d ':' -f 4`"
    
```

```
LINITRD="`echo ${LINUX} | cut -d ':' -f 5`"
LPARAMS="`echo ${LINUX} | cut -d ':' -f 6- | tr '^' ' '`"

if [ -z "${LLABEL}" ] ; then
    LLABEL="${LONGNAME}"
fi

if [ "${LRROOT}" != "${LB00T}" ]; then
    LKERNEL="${LKERNEL#/boot}"
    LINITRD="${LINITRD#/boot}"
fi

if [ -z "${prepare_boot_cache}" ]; then
    prepare_boot_cache="$(prepare_grub_to_access_device ${LB00T} | grub_add_tab)"
    [ "${prepare_boot_cache}" ] || continue
fi

found_other_os=1
onstr="$(gettext_printf "(on %s)" "${DEVICE}")"
recovery_params="$(echo "${LPARAMS}" | grep 'single\\|recovery') || true"
counter=1
while echo "$used_osprober_linux_ids" | grep 'osprober-gnulinux-$LKERNEL-${recovery_params}-${counter}-$boot_device_id' > /dev/null; do
    counter=$((counter+1));
done
if [ -z "$boot_device_id" ]; then
    boot_device_id="$(grub_get_device_id "${DEVICE}")"
fi
used_osprober_linux_ids="$used_osprober_linux_ids 'osprober-gnulinux-$LKERNEL-${recovery_params}-${counter}-$boot_device_id'"

if [ "x$is_top_level" = xtrue ] && [ "x${GRUB_DISABLE_SUBMENU}" != xy ]; then
    cat << EOF
menuentry '$(echo "$OS $onstr" | grub_quote)' --class gnu-linux --class gnu --class os \ $menuentry_id_option
```

```
'osprober-gnulinix-simple-$boot_device_id' {
EOF
    save_default_entry | grub_add_tab
    printf '%s\n' "${prepare_boot_cache}"
    cat << EOF
linux ${LKERNEL} ${LPARAMS}
EOF
    if [ -n "${LINITRD}" ] ; then
        cat << EOF
initrd ${LINITRD}
EOF
    fi
    cat << EOF
}
EOF
    echo "submenu '$(gettext_printf "Advanced options for %s" "${OS}" $onstr" | grub_quote)'
\menuentry_id_option 'osprober-gnulinix-advanced-$boot_device_id' {"
    is_top_level=false
    fi
    title="${LLABEL} $onstr"
    cat << EOF
    menuentry '$(echo "$title" | grub_quote)' --class gnu-linux --class gnu --class os \menuentry_id_option
'osprober-gnulinix-$LKERNEL-${recovery_params}-$boot_device_id' {
EOF
    save_default_entry | sed -e "s/^/$grub_tab$grub_tab/"
    printf '%s\n' "${prepare_boot_cache}" | grub_add_tab
    cat << EOF
linux ${LKERNEL} ${LPARAMS}
EOF
    if [ -n "${LINITRD}" ] ; then
        cat << EOF
initrd ${LINITRD}
EOF
    fi
```

```

        cat << EOF
    }
EOF
    if [ x"$title" = x"$GRUB_ACTUAL_DEFAULT" ] || [ x"Previous Linux versions>$title" = x"$GRUB_ACTUAL_DEFAULT"
]; then
        replacement_title="$(echo "Advanced options for ${OS} $onstr" | sed 's,>,>>,g')>$(echo "$title" | sed
's,>,>>,g')"
        quoted="$(echo "$GRUB_ACTUAL_DEFAULT" | grub_quote)"
        title_correction_code="${title_correction_code}if [ \"x\\$default\" = '$quoted' ]; then default='$(echo
"$replacement_title" | grub_quote)'; fi;"
        grub_warn "$(gettext_printf "Please don't use old title \\`%s' for GRUB_DEFAULT, use \\`%s' (for versions
before 2.00) or \\`%s' (for 2.00 or later)" "$GRUB_ACTUAL_DEFAULT" "$replacement_title" "gnulinux-advanced-
$boot_device_id>gnulinux-$version-$type-$boot_device_id")"
    fi
done
if [ x"$is_top_level" != xtrue ]; then
    echo '}'
fi
echo "$title_correction_code"
;;
macosx)
    if [ "${UUID}" ]; then
        OSXUUID="${UUID}"
        osx_entry xnu_kernel 32
        osx_entry xnu_kernel64 64
    fi
;;
hurd)
    found_other_os=1
    onstr="$(gettext_printf "(on %s)" "${DEVICE}")"
    cat << EOF
menuentry '$(echo "${LONGNAME} $onstr" | grub_quote)' --class hurd --class gnu --class os \${menuentry_id_option
'osprober-gnuhurd-/boot/gnumach.gz-false-$(grub_get_device_id "${DEVICE}")' {
EOF

```

```

    save_default_entry | grub_add_tab
    prepare_grub_to_access_device ${DEVICE} | grub_add_tab
    grub_device="\${grub_probe} --device ${DEVICE} --target=drive`"
    mach_device="\`echo "\${grub_device}" | sed -e 's/(\(hd.*\),msdos\(.*\))/\1s\2/'`"
    grub_fs="\${grub_probe} --device ${DEVICE} --target=fs`"
    case "\${grub_fs}" in
*fs)    hurd_fs="\${grub_fs}" ;;
*)      hurd_fs="\${grub_fs}fs" ;;
    esac
    cat << EOF
multiboot /boot/gnumach.gz root=device:${mach_device}
module /hurd/${hurd_fs}.static ${hurd_fs} --readonly \\\
    --multiboot-command-line='\${kernel-command-line}' \\\
    --host-priv-port='\${host-port}' \\\
    --device-master-port='\${device-port}' \\\
    --exec-server-task='\${exec-task}' -T typed '\${root}' \\\
    '\$(task-create)' '\$(task-resume)'
module /lib/ld.so.1 exec /hurd/exec '\$(exec-task=task-create)'
}
EOF
;;
minix)
    cat << EOF
menuentry "\${LONGNAME} (on ${DEVICE}, Multiboot)" {
EOF
    save_default_entry | sed -e "s/^/\t/"
    prepare_grub_to_access_device ${DEVICE} | sed -e "s/^/\t/"
    cat << EOF
multiboot /boot/image_latest
}
EOF
;;
*)
    echo -n " "

```

```
# TRANSLATORS: %s is replaced by OS name.
gettext_printf "%s is not yet supported by grub-mkconfig.\n" "${LONGNAME}" >&2
;;
esac
done

adjust_timeout
```

Les fichiers `/etc/grub.d/40_custom` et `/etc/grub.d/41_custom`

Ces deux fichiers sont fournis en tant que modèles à personnaliser :

```
root@ubuntu:~# cat /etc/grub.d/40_custom
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```



La personnalisation d'un fichier Grub2 est couverte dans la section **Initramfs** de cet unité.

Initramfs

Initrd *INITial Ram File System* est une image d'un système minimal initialisé au démarrage du système.

Ce fichier utilise le système de fichier **cramFS** qui est un système de fichier compressé au format gzip et archivé via cpio.

L'image est chargée en mémoire vive et permet ainsi d'avoir un système minimal pouvant ensuite charger le système de fichier principal.

Examiner l'image existante

Pour examiner l'image existante, il convient d'abord de la copier vers /tmp et de la décompresser :

```
root@ubuntu:~# cp /boot/initrd.img-3.13.0-32-generic /tmp/custom.gz
root@ubuntu:~# cd /tmp
root@ubuntu:/tmp# gunzip custom.gz
```

Ensuite il convient d'extraire l'image grâce à la commande **cpio** :

```
root@ubuntu:/tmp# mkdir initrd
root@ubuntu:/tmp# cd initrd
root@ubuntu:/tmp/initrd# cpio -idvB < ../custom
...
```

Installez maintenant le paquet **tree** :

```
root@ubuntu:/tmp/initrd# apt-get install tree
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les NOUVEAUX paquets suivants seront installés :
  tree
0 mis à jour, 1 nouvellement installés, 0 à enlever et 171 non mis à jour.
Il est nécessaire de prendre 36,7 ko dans les archives.
Après cette opération, 112 ko d'espace disque supplémentaires seront utilisés.
Réception de : 1 http://fr.archive.ubuntu.com/ubuntu/ trusty/universe tree i386 1.6.0-1 [36,7 kB]
36,7 ko réceptionnés en 0s (260 ko/s)
Sélection du paquet tree précédemment désélectionné.
(Lecture de la base de données... 168406 fichiers et répertoires déjà installés.)
Préparation du décompactage de .../archives/tree_1.6.0-1_i386.deb ...
Décompactage de tree (1.6.0-1) ...
Traitement déclenché pour man-db (2.6.7.1-1) ...
```

Paramétrage de tree (1.6.0-1) ...

Utilisez la commande **tree** pour examiner le contenu de l'image :

```
root@ubuntu:/tmp/initrd# tree | more
```

```
.
├── bin
│   ├── busybox
│   ├── cpio
│   ├── date
│   ├── dd
│   ├── dmesg
│   ├── fstype
│   ├── halt
│   ├── insmod
│   ├── ipconfig
│   ├── kbd_mode
│   ├── kmod
│   ├── loadkeys
│   ├── losetup
│   ├── mount -> busybox
│   ├── nfsmount
│   ├── ntfs-3g
│   ├── pivot_root
│   ├── poweroff
│   ├── reboot
│   ├── resume
│   ├── run-init
│   ├── setfont
│   ├── sh -> busybox
│   ├── sleep -> busybox
│   └── udevadm
└── conf
    └── arch.conf
```

```
|   |   | conf.d
|   |   | | resume
|   |   | | initramfs.conf
|   |   |
|   |   | etc
|   |   | | console-setup
|   |   |
|--Plus--
```

Comme vous pouvez le constater, l'image contient une arborescence Linux minimaliste :

```
root@ubuntu:/tmp/initrd# ls
bin  conf  etc  init  lib  run  sbin  scripts
```



Utilisez le manuel de la commande **cpio** pour comprendre les options utilisées.

Le script **init**

Le script **init** est lancé lors du chargement de l'image :

```
root@ubuntu:/tmp/initrd# more init
#!/bin/sh

[ -d /dev ] || mkdir -m 0755 /dev
[ -d /root ] || mkdir -m 0700 /root
[ -d /sys ] || mkdir /sys
[ -d /proc ] || mkdir /proc
[ -d /tmp ] || mkdir /tmp
mkdir -p /var/lock
mount -t sysfs -o nodev,noexec,nosuid sysfs /sys
mount -t proc -o nodev,noexec,nosuid proc /proc
# Some things don't work properly without /etc/mtab.
```

```
ln -sf /proc/mounts /etc/mtab

grep -q '\<quiet\>' /proc/cmdline || echo "Loading, please wait..."

# Note that this only becomes /dev on the real filesystem if udev's scripts
# are used; which they will be, but it's worth pointing out
if ! mount -t devtmpfs -o mode=0755 udev /dev; then
    echo "W: devtmpfs not available, falling back to tmpfs for /dev"
    mount -t tmpfs -o mode=0755 udev /dev
    [ -e /dev/console ] || mknod -m 0600 /dev/console c 5 1
    [ -e /dev/null ] || mknod /dev/null c 1 3
fi
mkdir /dev/pts
mount -t devpts -o noexec,nosuid,gid=5,mode=0620 devpts /dev/pts || true
mount -t tmpfs -o "noexec,nosuid,size=10%,mode=0755" tmpfs /run
mkdir /run/initramfs
# compatibility symlink for the pre-oneiric locations
ln -s /run/initramfs /dev/.initramfs

# Export the dpkg architecture
export DPKG_ARCH=
. /conf/arch.conf

--Plus-- (16%)
```



Passez en revue le contenu du script.

La commande **mkinitramfs**

La commande **mkinitramfs** permet de créer facilement une image initramfs.

Afin d'inclure des modules supplémentaires dans un nouveau fichier initramfs, il convient d'éditer le fichier **/etc/initramfs-tools/modules** :

[/etc/initramfs-tools/modules](#)

```
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod
# ehci-hcd
# uhci-mod
# ohci-mod
# usb-storage
# scsi-mod
# sd-mod
```

Exécutez ensuite la commande suivante :

```
# mkinitramfs -o usbinitramfs-`uname -r`.img [Entrée]
```

Notez la présence de votre nouvelle image **/tmp/initrd/usbinitramfs-3.13.0-32-generic.img**.

Déplacez votre fichier **usbinitramfs-3.13.0-32-generic.img** au répertoire /boot :

```
root@ubuntu:/tmp/initrd# mv usbinitramfs-3.13.0-32-generic.img /boot
root@ubuntu:/tmp/initrd# cd /boot
root@ubuntu:/boot# ls
```

```
abi-3.13.0-32-generic      grub                        memtest86+.bin  memtest86+_multiboot.bin
usbinitramfs-3.13.0-32-generic.img
config-3.13.0-32-generic  initrd.img-3.13.0-32-generic  memtest86+.elf  System.map-3.13.0-32-generic
vmlinuz-3.13.0-32-generic
```

Créez maintenant le fichier **/etc/grub.d/09_ubuntu** :

```
#!/bin/sh -e
cat << EOF
menuentry 'Ubuntu avec USB' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option
'gnulinux-simple-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
    recordfail
    load_video
    gfxmode $linux_gfx_mode
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d-9797-9342a7b82cb2
    else
        search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
    fi
    linux    /boot/vmlinuz-3.13.0-32-generic root=UUID=70eb8bc5-1759-433d-9797-9342a7b82cb2 ro  quiet splash
    $vt_handoff
    initrd   /boot/usbinitramfs-3.13.0-32-generic.img
}
EOF
```

Rendez ce fichier exécutable :

```
root@ubuntu:/boot# cd /etc/grub.d
```

```
root@ubuntu:/etc/grub.d# chmod +x 09_usbubuntu
```

Mettez à jour grub afin que celui-ci prend en compte le nouveau fichier :

```
root@ubuntu:/etc/grub.d# update-grub
Création du fichier de configuration GRUB...
Attention : Définir GRUB_TIMEOUT à une valeur non nulle si GRUB_HIDDEN_TIMEOUT est définie n'est plus possible.
Image Linux trouvée : /boot/vmlinuz-3.13.0-32-generic
Image mémoire initiale trouvée : /boot/initrd.img-3.13.0-32-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
fait
```

Contrôlez le fichier /boot/grub/grub.cfg :

```
...
### BEGIN /etc/grub.d/05_debian_theme ###
set menu_color_normal=white/black
set menu_color_highlight=black/light-gray
if background_color 44,0,30; then
  clear
fi
### END /etc/grub.d/05_debian_theme ###

### BEGIN /etc/grub.d/09_usbubuntu ###
menuentry 'Ubuntu avec USB' --class ubuntu --class gnu-linux --class gnu --class os 'gnulinux-
simple-70eb8bc5-1759-433d-9797-9342a7b82cb2' {
  recordfail
  load_video
  gfxmode
  insmod gzio
  insmod part_msdos
  insmod ext2
  set root='hd0,msdos1'
```

```
if [ x = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-
baremetal=ahci0,msdos1 70eb8bc5-1759-433d
-9797-9342a7b82cb2
else
    search --no-floppy --fs-uuid --set=root 70eb8bc5-1759-433d-9797-9342a7b82cb2
fi
linux    /boot/vmlinuz-3.13.0-32-generic root=UUID=70eb8bc5-1759-433d-9797-9342a7b82cb2 ro quiet splash
initrd   /boot/usbinitramfs-3.13.0-32-generic.img
}
### END /etc/grub.d/09_usbubuntu ###

### BEGIN /etc/grub.d/10_linux ###
function gfxmode {
    set gfxpayload="${1}"
    ...
}
```

Processus de Démarrage de Linux

Le processus de démarrage de Linux peut être divisé en 6 étapes :

Etape	Description
Chargement, configuration et exécution du chargeur du noyau	Le fichier bootsect.s est chargé en mémoire par le BIOS. Une fois configuré celui-ci charge le reste du noyau en mémoire
Configuration des paramètres et bascule vers le mode 32 bits	Le fichier boot.s met en place un IDT (<i>Interrupt Descriptor Table</i>) temporaire et GDT (<i>Global Descriptor Table</i>) temporaire et gère le basculement vers le mode 32 bits
Décompression du Noyau	Le fichier head.s décompresse le noyau
Initialisation du noyau et de la mémoire	Le fichier head.s crée un GDT et IDT définitif
Configuration du noyau	Le fichier main.c met en place les contraintes de mémoire et configure la mémoire virtuelle
Création du processus Init	Le fichier main.c crée le processus init

La fonction **init_post()** essaie ensuite d'exécuter un des processus suivant dans l'ordre :

- /sbin/init
- /etc/init
- /bin/init
- /bin/sh

Dans le cas d'un échec à ce stade le message **Kernel Panic** sera afficher.

Processus Init

Le premier processus lancé par le noyau est **Init**. L'exécutable lancé est **/sbin/init**. Son rôle est de d'initialiser le système et de lancer certains autres services. Les tâches accomplies par init sont :

- le montage de /proc et de /sys
- configuration des paramètres du noyau présents dans **/etc/sysctl.conf**,
- l'activation de SELinux,
- la mise à l'heure du système,
- la définition des consoles textes,
- la définition du nom de la machine,
- la détection des périphériques USB,
- la mise en place du support RAID et LVM,
- l'activation des quotas de disque,
- le montages des systèmes de fichiers,
- le re-montage du système de fichiers racine en lecture/écriture,
- l'activation du swap,
- le lancement de syslog,
- le chargement des modules du noyau,
- le nettoyage des fichiers temporaires,
- la définition des variables d'environnement tels PATH et RUNLEVEL

Niveaux d'exécution

Il existe 8 niveaux d'exécution ou **RUNLEVELS** sous Linux. Quatre des 8 sont réservés :

RUNLEVEL	Description
0	Arrêt de la machine
1	Mode mono-utilisateur pour la maintenance
6	Redémarrage de la machine
S ou s	Mode mono-utilisateur avec seul la partition racine montée

Les autres quatre RUNLEVELS sont définis par chaque distribution. Par exemple, sous Ubuntu, ils sont :

RUNLEVEL	Description
2	Mode multi-utilisateur
3	Mode multi-utilisateur - Non-utilisé
4	Mode multi-utilisateur - Non-utilisé
5	Mode multi-utilisateur - Non-utilisé

Il existe aussi 3 pseudo-niveaux d'exécution **a**, **b** et **c**. Ces pseudo-niveaux permettent à init de faire quelque chose sans changer de niveau d'exécution.

Pour connaître le niveau d'exécution actuel de la machine, saisissez la commande suivante :

```
root@ubuntu:~# runlevel
N 2
```

La lettre N indique que le système n'a pas changé de niveau d'exécution depuis son démarrage.

Pour modifier le niveau d'exécution courant, il convient d'utiliser la commande **init** ou **telinit** suivie du numéro du nouveau niveau d'exécution. Ces commandes peuvent prendre plusieurs options :

Option	Description
Q ou q	Demande à Init de relire le fichier /etc/inittab
-t	Permet de modifier le temps accordé par Init aux processus entre l'envoi du signal SIGTERM et l'envoi du signal SIGKILL

Le Système de Démarrage Unix SysV

Répertoire **init.d**

Le répertoire **/etc/init.d** contient les scripts permettant de lancer les services du système :

```
root@ubuntu:~# cd /etc/init.d
root@ubuntu:/etc/init.d# ls
acpid          friendly-recovery  rc              sudo
anacron        grub-common       rc.local        udev
apparmor       halt              rcS             umountfs
apport         irqbalance        README          umountnfs.sh
avahi-daemon   kerneloops        reboot          umountroot
bluetooth     killprocs         resolvconf      unattended-upgrades
brltty        kmod              rsync           urandom
console-setup lightdm            rsyslog         vboxadd
cron           networking        saned           vboxadd-service
cups           ondemand          sendsigs        vboxadd-x11
cups-browsed  pppd-dns          single          x11-common
dbus          procps            skeleton
dns-clean    pulseaudio        speech-dispatcher
```

Le script **rc.S**

D'après l'étude du fichier **inittab**, nous savons que le script **/etc/init.d/rcS** est exécuté en premier. Ce script appelle tous les scripts dans **/etc/init.d** qui sont référencés par les liens dans **/etc/rcS.d/** et ceci dans un ordre numérique/alphabétique :

```
root@ubuntu:/etc/init.d# cat /etc/init.d/rcS
#!/bin/sh
#
# rcS
#
# Call all S??* scripts in /etc/rcS.d/ in numerical/alphabetical order
#
```

```
exec /etc/init.d/rc S
```

Répertoires rcx.d

Les répertoires **/etc/rc0.d** à **/etc/rc6.d** contiennent des liens vers les scripts du répertoire **init.d**.

Pour mieux comprendre, saisissez les commandes suivantes :

```
root@ubuntu:/etc/init.d# for rep in /etc/rc[2345].d; do echo "dans $rep : "; ls $rep/S*; done
dans /etc/rc2.d :
/etc/rc2.d/S20kerneloops      /etc/rc2.d/S30vboxadd        /etc/rc2.d/S50saned          /etc/rc2.d/S99grub-
common
/etc/rc2.d/S20rsync           /etc/rc2.d/S30vboxadd-x11    /etc/rc2.d/S70dns-clean     /etc/rc2.d/S99ondemand
/etc/rc2.d/S20speech-dispatcher /etc/rc2.d/S35vboxadd-service /etc/rc2.d/S70pppd-dns      /etc/rc2.d/S99rc.local
dans /etc/rc3.d :
/etc/rc3.d/S20kerneloops      /etc/rc3.d/S30vboxadd        /etc/rc3.d/S50saned          /etc/rc3.d/S99grub-
common
/etc/rc3.d/S20rsync           /etc/rc3.d/S30vboxadd-x11    /etc/rc3.d/S70dns-clean     /etc/rc3.d/S99ondemand
/etc/rc3.d/S20speech-dispatcher /etc/rc3.d/S35vboxadd-service /etc/rc3.d/S70pppd-dns      /etc/rc3.d/S99rc.local
dans /etc/rc4.d :
/etc/rc4.d/S20kerneloops      /etc/rc4.d/S30vboxadd        /etc/rc4.d/S50saned          /etc/rc4.d/S99grub-
common
/etc/rc4.d/S20rsync           /etc/rc4.d/S30vboxadd-x11    /etc/rc4.d/S70dns-clean     /etc/rc4.d/S99ondemand
/etc/rc4.d/S20speech-dispatcher /etc/rc4.d/S35vboxadd-service /etc/rc4.d/S70pppd-dns      /etc/rc4.d/S99rc.local
dans /etc/rc5.d :
/etc/rc5.d/S20kerneloops      /etc/rc5.d/S30vboxadd        /etc/rc5.d/S50saned          /etc/rc5.d/S99grub-
common
/etc/rc5.d/S20rsync           /etc/rc5.d/S30vboxadd-x11    /etc/rc5.d/S70dns-clean     /etc/rc5.d/S99ondemand
/etc/rc5.d/S20speech-dispatcher /etc/rc5.d/S35vboxadd-service /etc/rc5.d/S70pppd-dns      /etc/rc5.d/S99rc.local
```



Notez que chaque répertoire correspondant à un niveau d'exécution contient des liens



pointant vers un script dans le répertoire **/etc/init.d**. La lettre **S** indique au script **rc** que le script dans **/etc/init.d** doit être exécuté avec l'option **start**. De cette façon les processus sont lancés dans le niveau d'exécution spécifié. Le numéro qui suit la lettre **S** indique l'ordre de lancement par le script **rc**. Si deux scripts dans un répertoire **/etc/rcX.d** ont le même numéro, l'ordre alphabétique prime. Notez aussi la présence du lien **S99rc.local** qui lance le script **rc.local** en dernier. Ce script peut contenir des commandes spécifiées par root à lancer lors du démarrage de la machine.

Rappelez la commande précédente et modifiez la lettre S en **K** :

```
root@ubuntu:/etc/init.d# for rep in /etc/rc[016].d; do echo "dans $rep :"; ls $rep/K*; done
dans /etc/rc0.d :
/etc/rc0.d/K10unattended-upgrades  /etc/rc0.d/K20rsync                /etc/rc0.d/K65vboxadd-service
/etc/rc0.d/K70vboxadd-x11
/etc/rc0.d/K20kerneloops           /etc/rc0.d/K20speech-dispatcher  /etc/rc0.d/K70vboxadd
dans /etc/rc1.d :
/etc/rc1.d/K20kerneloops  /etc/rc1.d/K20saned              /etc/rc1.d/K65vboxadd-service  /etc/rc1.d/K70vboxadd-
x11
/etc/rc1.d/K20rsync       /etc/rc1.d/K20speech-dispatcher  /etc/rc1.d/K70vboxadd
dans /etc/rc6.d :
/etc/rc6.d/K10unattended-upgrades  /etc/rc6.d/K20rsync                /etc/rc6.d/K65vboxadd-service
/etc/rc6.d/K70vboxadd-x11
/etc/rc6.d/K20kerneloops           /etc/rc6.d/K20speech-dispatcher  /etc/rc6.d/K70vboxadd
```

Ici le principe est le même sauf que la lettre **K** indique au script **rc** que le script dans **/etc/init.d** doit être lancé avec l'option **stop**.

rc.local

Le script **rc.local** est lancé dans les niveaux d'exécution **2, 3, 4 et 5**. C'est dans ce script que **root** peut ajouter des commandes.

La commande **update-rc.d**

La commande **update-rc.d** est utilisée pour gérer les liens dans les répertoires rc[x].d. Cette commande permet d'insérer les liens vers un script dans /etc/init.d et prend la forme suivante :

```
update-rc.d <service> start <priorité de démarrage> <runlevels de démarrage> . stop <priorité d'arrêt> <runlevels d'arrêt> .
```

Par exemple la commande suivante crée les liens **S** dans rc2.d à rc5.d avec une priorité de 20 et les liens **K** dans rc0.d, rc1.d et rc6.d avec une priorité de 20 pour le script /etc/init.d/ssh :

```
# update-rc.d ssh start 20 2 3 4 5 . stop 20 0 1 6 . [Entrée]
```



Il existe aussi un mot clef : **default**. Ce mot clef indique à **update-rc.d** d'utiliser les valeurs spécifiées dans la command précédente.

Pour supprimer les liens il convient d'utiliser la commande suivante :

```
# update-rc.d -f ssh remove [Entrée]
```



Cette commande laisse le script ssh dans le répertoire /etc/init.d.

Options de la commande

Les options de cette commande sont :

```
root@debian:/etc/init.d# update-rc.d --help
update-rc.d: using dependency based boot sequencing
update-rc.d: error: --help
usage: update-rc.d [-n] [-f] <basename> remove
      update-rc.d [-n] <basename> defaults [NN | SS KK]
      update-rc.d [-n] <basename> start|stop NN runlvl [runlvl] [...] .
      update-rc.d [-n] <basename> disable|enable [S|2|3|4|5]
      -n: not really
      -f: force
```

The disable|enable API is not stable and might change in the future.

Le Système de Démarrage Upstart

Ubuntu utilise actuellement le système de démarrage **Upstart** qui remplace progressivement le système de démarrage SysV. La configuration du démarrage se trouvent dans les nombreux fichiers du répertoire **/etc/init/** :

```
root@ubuntu:/etc/init.d# cd /etc/init
root@ubuntu:/etc/init# ls
acpid.conf          lightdm.conf        pulseaudio.conf
alsa-restore.conf   modemmanager.conf  rc.conf
alsa-state.conf     mountall-bootclean.sh.conf rcS.conf
alsa-store.conf     mountall.conf       rc-sysinit.conf
anacron.conf        mountall-net.conf   resolvconf.conf
appport.conf        mountall-reboot.conf rfkill-restore.conf
avahi-cups-reload.conf mountall.sh.conf    rfkill-store.conf
avahi-daemon.conf   mountall-shell.conf rsyslog.conf
bluetooth.conf      mountdevsubfs.sh.conf setvtrgb.conf
bootmisc.sh.conf    mounted-debugfs.conf shutdown.conf
checkfs.sh.conf     mounted-dev.conf    startpar-bridge.conf
checkroot-bootclean.sh.conf mounted-proc.conf  systemd-logind.conf
checkroot.sh.conf   mounted-run.conf    tty1.conf
```

console.conf	mounted-tmp.conf	tty2.conf
console-font.conf	mounted-var.conf	tty3.conf
console-setup.conf	mountkernfs.sh.conf	tty4.conf
container-detect.conf	mountnfs-bootclean.sh.conf	tty5.conf
control-alt-delete.conf	mountnfs.sh.conf	tty6.conf
cron.conf	mtab.sh.conf	udev.conf
cups-browsed.conf	networking.conf	udev-fallback-graphics.conf
cups.conf	network-interface.conf	udev-finish.conf
dbus.conf	network-interface-container.conf	udevmonitor.conf
dmesg.conf	network-interface-security.conf	udevtrigger.conf
failsafe.conf	network-manager.conf	ufw.conf
failsafe-x.conf	passwd.conf	upstart-file-bridge.conf
flush-early-job-log.conf	plymouth.conf	upstart-socket-bridge.conf
friendly-recovery.conf	plymouth-log.conf	upstart-udev-bridge.conf
gpu-manager.conf	plymouth-ready.conf	ureadahead.conf
hostname.conf	plymouth-shutdown.conf	ureadahead-other.conf
hwclock.conf	plymouth-splash.conf	usb-modeswitch-upstart.conf
hwclock-save.conf	plymouth-stop.conf	wait-for-state.conf
irqbalance.conf	plymouth-upstart-bridge.conf	whoopsie.conf
kmod.conf	procps.conf	

Initialisation du Système

L'initialisation du système est configurée dans le fichier **/etc/init/rc-sysinit.conf** :

```
root@ubuntu:/etc/init# cat /etc/init/rc-sysinit.conf
# rc-sysinit - System V initialisation compatibility
#
# This task runs the old System V-style system initialisation scripts,
# and enters the default runlevel when finished.

description "System V initialisation compatibility"
author      "Scott James Remnant <scott@netsplit.com>"
```

```
start on (filesystem and static-network-up) or failsafe-boot
stop on runlevel
```

```
# Default runlevel, this may be overridden on the kernel command-line
# or by faking an old /etc/inittab entry
env DEFAULT_RUNLEVEL=2
```

```
emits runlevel
```

```
# There can be no previous runlevel here, but there might be old
# information in /var/run/utmp that we pick up, and we don't want
# that.
#
# These override that
env RUNLEVEL=
env PREVLEVEL=
```

```
console output
env INIT_VERBOSE
```

```
task
```

```
script
    # Check for default runlevel in /etc/inittab
    if [ -r /etc/inittab ]
    then
        eval "$(sed -nre 's/^[^#][^:]*:([0-6sS]):initdefault:.*\/DEFAULT_RUNLEVEL="\1";/p' /etc/inittab || true)"
    fi

    # Check kernel command-line for typical arguments
    for ARG in $(cat /proc/cmdline)
    do
        case "${ARG}" in
            -b|emergency)
```

```
# Emergency shell
[ -n "${FROM_SINGLE_USER_MODE}" ] || sulogin
;;
[0123456sS])
# Override runlevel
DEFAULT_RUNLEVEL="${ARG}"
;;
-s|single)
# Single user mode
[ -n "${FROM_SINGLE_USER_MODE}" ] || DEFAULT_RUNLEVEL=S
;;
esac
done

# Run the system initialisation scripts
[ -n "${FROM_SINGLE_USER_MODE}" ] || /etc/init.d/rcS

# Switch into the default runlevel
telinit "${DEFAULT_RUNLEVEL}"
end script
```

Dans ce fichier on constate la présence d'une ligne qui vérifie la présence du fichier **/etc/inittab** :

```
...
# Check for default runlevel in /etc/inittab
if [ -r /etc/inittab ]
then
eval "$(sed -nre 's/^[^#][^:]*:([0-6sS]):initdefault:.*\/DEFAULT_RUNLEVEL="\1";/p' /etc/inittab || true)"
fi
...
```

Ce fichier était utilisé par le système de démarrage SysV pour configurer le démarrage du système. Se trouvait notamment dans ce fichier le niveau d'exécution par défaut dans lequel le système va être mis à la fin du démarrage. Ce niveau d'exécution est appelé le niveau d'exécution par défaut. Si ce fichier existe et s'il contient une ligne de type **id:2:initdefault:**, le niveau d'exécution par défaut est fixé à 2. Dans le cas inverse le niveau

d'exécution par défaut est fixé par la directive **env DEFAULT_RUNLEVEL=2**. Notez qu'à la fin du fichier, l'exécution du script **/etc/init/rcS.conf** est appelée :

```
root@ubuntu:/etc/init# cat /etc/init/rc.conf
# rc - System V runlevel compatibility
#
# This task runs the old System V-style rc script when changing between
# runlevels.

description "System V runlevel compatibility"
author      "Scott James Remnant <scott@netsplit.com>"

emits deconfiguring-networking
emits unmounted-remote-filesystems

start on runlevel [0123456]
stop on runlevel [!$RUNLEVEL]

export RUNLEVEL
export PREVLEVEL

console output
env INIT_VERBOSE

task

script
if [ "$RUNLEVEL" = "0" -o "$RUNLEVEL" = "1" -o "$RUNLEVEL" = "6" ]; then
    status plymouth-shutdown 2>/dev/null >/dev/null && start wait-for-state WAITER=rc WAIT_FOR=plymouth-shutdown
|| :
fi
/etc/init.d/rc $RUNLEVEL
end script
```

Runlevels

La gestion des Runlevels est configurée dans le fichier **/etc/init/rc.conf** :

```
root@ubuntu:/etc/init# cat /etc/init/rc.conf
# rc - System V runlevel compatibility
#
# This task runs the old System V-style rc script when changing between
# runlevels.

description "System V runlevel compatibility"
author      "Scott James Remnant <scott@netsplit.com>"

emits deconfiguring-networking
emits unmounted-remote-filesystems

start on runlevel [0123456]
stop on runlevel [!$RUNLEVEL]

export RUNLEVEL
export PREVLEVEL

console output
env INIT_VERBOSE

task

script
if [ "$RUNLEVEL" = "0" -o "$RUNLEVEL" = "1" -o "$RUNLEVEL" = "6" ]; then
    status plymouth-shutdown 2>/dev/null >/dev/null && start wait-for-state WAITER=rc WAIT_FOR=plymouth-shutdown
|| :
fi
/etc/init.d/rc $RUNLEVEL
```

```
end script
```

[CTL]-[ALT]-[DEL]

Le comportement associé avec la combinaison de touches [CTL]-[ALT]-[DEL] est configuré dans le fichier **/etc/init/control-alt-delete.conf** :

```
root@ubuntu:/etc/init# cat /etc/init/control-alt-delete.conf
# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete key combination is
# pressed, and performs a safe reboot of the machine.

description "emergency keypress handling"
author      "Scott James Remnant <scott@netsplit.com>"

start on control-alt-delete

task
exec shutdown -r now "Control-Alt-Delete pressed"
```

Configuration des Services Upstart

La configuration des services Upstart peut être visualisée en utilisant la commande **initctl** :

```
root@ubuntu:~# initctl show-config
avahi-daemon
  start on (filesystem and started dbus)
  stop on stopping dbus
mountnfs-bootclean.sh
  start on virtual-filesystems
rsyslog
```

```
    start on filesystem
    stop on runlevel [06]
tty4
    start on (runlevel [23] and ((not-container or container CONTAINER=lxc) or container CONTAINER=lxc-libvirt))
    stop on runlevel [!23]
udev
    start on virtual-filesystems
    stop on runlevel [06]
upstart-udev-bridge
    emits *-device-added
    emits *-device-removed
    emits *-device-changed
    emits *-device-online
    emits *-device-offline
    start on starting udev
    stop on stopped udev
whoopsie
    start on runlevel [2345]
    stop on runlevel [!2345]
avahi-cups-reload
    start on started avahi-daemon
mountall-net
    start on net-device-up
passwd
    start on filesystem
rc
    emits deconfiguring-networking
    emits unmounted-remote-filesystems
    start on runlevel [0123456]
    stop on runlevel [!$RUNLEVEL]
startpar-bridge
    start on (started JOB!=startpar-bridge or stopped JOB!=startpar-bridge)
ureadahead-other
    start on mounted DEVICE=[/UL]* MOUNTPPOINT=/*
```

```
apport
  start on runlevel [2345]
  stop on runlevel [!2345]
systemd-logind
  start on started dbus
  stop on stopping dbus
tty5
  start on (runlevel [23] and not-container)
  stop on runlevel [!23]
console-setup
  start on ((virtual-filesystems or starting rcS) or starting mountall-shell)
gpu-manager
  start on (((starting lightdm or starting kdm) or starting xdm) or starting lxdm)
hwclock-save
  start on runlevel [06]
irqbalance
  start on runlevel [2345]
  stop on runlevel [!2345]
plymouth-log
  start on filesystem
mountall.sh
  start on local-filesystems
failsafe
  emits failsafe-boot
  start on (filesystem and net-device-up IFACE=lo)
  stop on (static-network-up or starting rc-sysinit)
rfkill-store
  start on runlevel [016]
modemmanager
  start on starting network-manager
  stop on stopped network-manager
dbus
  start on local-filesystems
  stop on deconfiguring-networking
```

```
resolvconf
    start on mounted MOUNTPoint=/run
    stop on runlevel [06]
failsafe-x
    start on (stopped lightdm EXIT_STATUS=[!0] or stopped gdm EXIT_STATUS=[!0])
    stop on runlevel [06]
mounted-var
    start on mounted MOUNTPoint=/var
plymouth-shutdown
    start on (runlevel [016] and (stopped xdm or stopped uxlanch))
plymouth
    start on starting mountall
udev-fallback-graphics
    start on (startup and (((graphics-device-added PRIMARY_DEVICE_FOR_DISPLAY=1 or drm-device-added
PRIMARY_DEVICE_FOR_DISPLAY=1) or stopped udevtrigger) or container))
usb-modeswitch-upstart
    start on usb-modeswitch-upstart
checkroot.sh
    start on mounted MOUNTPoint=/
control-alt-delete
    start on control-alt-delete
hwclock
    start on starting mountall
mounted-proc
    start on mounted MOUNTPoint=/proc TYPE=proc
network-manager
    start on ((local-filesystems and started dbus) and static-network-up)
    stop on stopping dbus
cups-browsed
    start on (filesystem and (started cups or runlevel [2345]))
    stop on runlevel [016]
alsa-store
    start on runlevel [!2345]
setvtrgb
```

```
    start on (started plymouth-splash or started tty1)
shutdown
    start on power-status-changed
cron
    start on runlevel [2345]
    stop on runlevel [!2345]
lightdm
    emits login-session-start
    emits desktop-session-start
    emits desktop-shutdown
    start on (((filesystem and runlevel [!06]) and started dbus) and plymouth-ready) or runlevel PREVLEVEL=S)
    stop on runlevel [016]
mountkernfs.sh
    start on virtual-filesystems
alsa-restore
    start on runlevel [2345]
mountall
    emits virtual-filesystems
    emits local-filesystems
    emits remote-filesystems
    emits all-swaps
    emits filesystem
    emits mounting
    emits mounted
    start on startup
    stop on starting rcS
mounted-debugfs
    start on mounted MOUNTPoint=/sys/kernel/debug TYPE=debugfs
console
    start on (stopped rc RUNLEVEL=[2345] and container CONTAINER=lxc)
    stop on runlevel [!2345]
mounted-run
    start on mounted MOUNTPoint=/run TYPE=tmpfs
acpid
```

```
    start on runlevel [2345]
    stop on runlevel [!2345]
bluetooth
    start on started dbus
    stop on stopping dbus
checkfs.sh
    start on mounted MOUNTPPOINT=/
checkroot-bootclean.sh
    start on mounted MOUNTPPOINT=/
mountnfs.sh
    start on remote-file systems
ufw
    start on ((starting network-interface or starting network-manager) or starting networking)
    stop on runlevel [!023456]
kmod
    start on (startup and started udev)
plymouth-stop
    start on (((((((starting gdm or starting kdm) or starting xdm) or starting lxdm) or starting lightdm) or
starting uxlaunch) or starting ubiquity) or starting oem-config) or stopped rc RUNLEVEL=[2345]) or starting rcS)
or starting mountall-shell)
    stop on stopped plymouth
rcS
    start on runlevel S
    stop on runlevel [!S]
wait-for-state
    stop on (started $WAIT_FOR or stopped $WAIT_FOR)
bootmisc.sh
    start on virtual-file systems
flush-early-job-log
    start on filesystem
friendly-recovery
    emits recovery
    emits startup
    emits mounted
```

```
    start on recovery
rc-sysinit
    emits runlevel
    start on ((filesystem and static-network-up) or failsafe-boot)
    stop on runlevel
upstart-socket-bridge
    emits socket
    start on net-device-up IFACE=lo
    stop on runlevel [!2345]
cups
    start on (filesystem and (started dbus or runlevel [2345]))
    stop on runlevel [016]
pulseaudio
    stop on runlevel [016]
mountdevsubfs.sh
    start on virtual-filesystems
tty2
    start on (runlevel [23] and ((not-container or container CONTAINER=lxc) or container CONTAINER=lxc-libvirt))
    stop on runlevel [!23]
upstart-file-bridge
    emits file
    start on filesystem
    stop on runlevel [!2345]
udevtrigger
    start on ((startup and started udev) and not-container)
anacron
    start on runlevel [2345]
    stop on runlevel [!2345]
mtab.sh
    start on virtual-filesystems
tty3
    start on (runlevel [23] and ((not-container or container CONTAINER=lxc) or container CONTAINER=lxc-libvirt))
    stop on runlevel [!23]
container-detect
```

```
    emits container
    emits not-container
    start on mounted MOUNTPoint=/run
mounted-dev
    start on mounted MOUNTPoint=/dev
udev-finish
    start on (((startup and filesystem) and started udev) and stopped udevtrigger) and stopped udevmonitor)
alsa-state
    start on runlevel [2345]
hostname
    start on startup
mountall-reboot
    start on stopped mountall EXIT_STATUS=4
network-interface
    emits net-device-up
    emits net-device-down
    emits static-network-up
    start on net-device-added
    stop on net-device-removed INTERFACE=$INTERFACE
plymouth-ready
    emits plymouth-ready
    start on (startup or started plymouth-splash)
tty1
    start on (stopped rc RUNLEVEL=[2345] and ((not-container or container CONTAINER=lxc) or container
CONTAINER=lxc-libvirt))
    stop on runlevel [!2345]
mountall-shell
    start on (stopped mountall EXIT_STATUS=[!4] or stopped mountall EXIT_SIGNAL=?*)
    stop on runlevel [06]
mounted-tmp
    start on (mounted MOUNTPoint=/tmp or mounted MOUNTPoint=/usr)
plymouth-splash
    start on (started plymouth and ((graphics-device-added PRIMARY_DEVICE_FOR_DISPLAY=1 or drm-device-added
PRIMARY_DEVICE_FOR_DISPLAY=1) or stopped udev-fallback-graphics))
```

```
plymouth-upstart-bridge
    start on (startup or runlevel [06])
    stop on (stopping plymouth or stopping plymouth-shutdown)
udevmonitor
    start on (startup and starting udevtrigger)
    stop on stopped udevtrigger
mountall-bootclean.sh
    start on virtual-filesystems
network-interface-security
    start on ((starting network-interface or starting network-manager) or starting networking)
    stop on ((stopped network-interface JOB=$JOB INTERFACE=$INTERFACE or stopped network-manager JOB=$JOB) or
stopped networking JOB=$JOB)
networking
    emits static-network-up
    emits net-device-up
    emits deconfiguring-networking
    start on (((local-filesystems and (stopped udevtrigger or container)) or runlevel [2345]) or stopped networking
RESULT=failed PROCESS=post-stop EXIT_STATUS=100)
    stop on unmounted-remote-filesystems
tty6
    start on (runlevel [23] and not-container)
    stop on runlevel [!23]
dmesg
    start on runlevel [2345]
procps
    start on (virtual-filesystems or static-network-up)
rfkill-restore
    start on local-filesystems
console-font
    start on starting plymouth-splash
network-interface-container
    emits net-device-added
    start on container
ureadahead
```

```
start on starting mountall
stop on stopped rc
```

Rappelez-vous que les services Upstart sont configurés par des fichiers dans **/etc/init** :

```
root@ubuntu:~# ls /etc/init/
acpid.conf                lightdm.conf              pulseaudio.conf
alsa-restore.conf         modemmanager.conf         rc.conf
alsa-state.conf           mountall-bootclean.sh.conf rcS.conf
alsa-store.conf           mountall.conf             rc-sysinit.conf
anacron.conf              mountall-net.conf         resolvconf.conf
apport.conf               mountall-reboot.conf      rfkill-restore.conf
avahi-cups-reload.conf    mountall.sh.conf          rfkill-store.conf
avahi-daemon.conf         mountall-shell.conf        rsyslog.conf
bluetooth.conf            mountdevsubfs.sh.conf     setvtrgb.conf
bootmisc.sh.conf          mounted-debugfs.conf      shutdown.conf
checkfs.sh.conf           mounted-dev.conf          startpar-bridge.conf
checkroot-bootclean.sh.conf mounted-proc.conf          systemd-logind.conf
checkroot.sh.conf         mounted-run.conf          tty1.conf
console.conf              mounted-tmp.conf          tty2.conf
console-font.conf         mounted-var.conf          tty3.conf
console-setup.conf        mountkernfs.sh.conf       tty4.conf
container-detect.conf     mountnfs-bootclean.sh.conf tty5.conf
control-alt-delete.conf   mountnfs.sh.conf          tty6.conf
cron.conf                 mtab.sh.conf              udev.conf
cups-browsed.conf         networking.conf           udev-fallback-graphics.conf
cups.conf                 network-interface.conf    udev-finish.conf
dbus.conf                 network-interface-container.conf udevmonitor.conf
dmesg.conf                network-interface-security.conf udevtrigger.conf
failsafe.conf             network-manager.conf      ufw.conf
failsafe-x.conf           passwd.conf               upstart-file-bridge.conf
flush-early-job-log.conf  plymouth.conf             upstart-socket-bridge.conf
friendly-recovery.conf    plymouth-log.conf         upstart-udev-bridge.conf
gpu-manager.conf          plymouth-ready.conf       ureadahead.conf
```

hostname.conf	plymouth-shutdown.conf	ureadahead-other.conf
hwclock.conf	plymouth-splash.conf	usb-modeswitch-upstart.conf
hwclock-save.conf	plymouth-stop.conf	wait-for-state.conf
irqbalance.conf	plymouth-upstart-bridge.conf	whoopsie.conf
kmod.conf	procs.conf	

Prenons comme exemple le fichier **network-manager.conf** :

```
root@ubuntu:~# cat /etc/init/network-manager.conf
# network-manager - network connection manager
#
# The Network Manager daemon manages the system's network connections,
# automatically switching between the best available.

description "network connection manager"

start on (local-filesystems
        and started dbus
        and static-network-up)
stop on stopping dbus

expect fork
respawn

script
    # set $LANG so that messages appearing on the GUI will be translated. See LP: 875017
    if [ -r /etc/default/locale ]; then
        . /etc/default/locale
        export LANG LANGUAGE LC_MESSAGES LC_ALL
    fi

    exec NetworkManager
end script
```

Le statut actuel des services Upstart peut être visualisé en utilisant la commande suivante :

```
root@ubuntu:~# initctl list
avahi-daemon start/running, process 691
mountnfs-bootclean.sh start/running
rsyslog start/running, process 614
tty4 start/running, process 744
udev start/running, process 272
upstart-udev-bridge start/running, process 267
whoopsie start/running, process 1009
avahi-cups-reload stop/waiting
mountall-net stop/waiting
passwd stop/waiting
rc stop/waiting
startpar-bridge stop/waiting
ureadahead-other stop/waiting
apport start/running
systemd-logind start/running, process 670
tty5 start/running, process 748
console-setup stop/waiting
gpu-manager stop/waiting
hwclock-save stop/waiting
irqbalance stop/waiting
plymouth-log stop/waiting
mountall.sh start/running
failsafe stop/waiting
rfkill-store stop/waiting
modemmanager stop/waiting
dbus start/running, process 555
resolvconf start/running
failsafe-x stop/waiting
mounted-var stop/waiting
plymouth-shutdown stop/waiting
plymouth stop/waiting
```

```
udev-fallback-graphics stop/waiting
usb-modeswitch-upstart stop/waiting
checkroot.sh start/running
control-alt-delete stop/waiting
hwclock stop/waiting
mounted-proc stop/waiting
network-manager stop/waiting
cups-browsed start/running, process 921
alsa-store stop/waiting
setvtrgb stop/waiting
shutdown stop/waiting
cron start/running, process 897
lightdm start/running, process 891
mountkernfs.sh start/running
alsa-restore stop/waiting
mountall stop/waiting
mounted-debugfs stop/waiting
console stop/waiting
mounted-run stop/waiting
acpid start/running, process 801
bluetooth start/running, process 681
checkfs.sh start/running
checkroot-bootclean.sh start/running
mountnfs.sh start/running
ufw start/running
kmod stop/waiting
plymouth-stop stop/waiting
rcS stop/waiting
wait-for-state stop/waiting
bootmisc.sh start/running
flush-early-job-log stop/waiting
friendly-recovery stop/waiting
rc-sysinit stop/waiting
upstart-socket-bridge start/running, process 401
```

```
cups start/running, process 4826
pulseaudio stop/waiting
mountdevsubfs.sh start/running
tty2 start/running, process 754
upstart-file-bridge start/running, process 612
udevtrigger stop/waiting
anacron stop/waiting
mtab.sh start/running
tty3 start/running, process 755
container-detect stop/waiting
mounted-dev stop/waiting
udev-finish stop/waiting
alsa-state stop/waiting
hostname stop/waiting
mountall-reboot stop/waiting
network-interface (lo) start/running
network-interface (eth0) start/running
plymouth-ready (started) start/running, process 963
tty1 start/running, process 1072
mountall-shell stop/waiting
mounted-tmp stop/waiting
plymouth-splash stop/waiting
plymouth-upstart-bridge stop/waiting
udevmonitor stop/waiting
mountall-bootclean.sh start/running
network-interface-security (network-interface/eth0) start/running
network-interface-security (network-interface/lo) start/running
network-interface-security (networking) start/running
networking start/running
tty6 start/running, process 758
dmesg stop/waiting
procps stop/waiting
rfkill-restore stop/waiting
console-font stop/waiting
```

```
network-interface-container stop/waiting
ureadahead stop/waiting
```

Arrêt du Système

La commande shutdown

Lors de l'arrêt de la machine, Linux procède, entre autre, aux tâches suivantes :

- Il prévient les utilisateurs,
- Il arrête tous les services,
- Il inscrit toutes les données sur disque,
- Il démonte les systèmes de fichiers.

La commande utilisée pour arrêter le système est la commande **shutdown** :

```
shutdown [-t sec] [-akrhHPfnc] heure [message]
```

Options de la commande

Les options de cette commande sont :

```
root@ubuntu:/etc/init.d# shutdown --help
Utilisation: shutdown [OPTION]... TEMPS [MESSAGE]
Permet d'éteindre ou de redémarrer l'ordinateur.
```

Options:

-r	reboot after shutdown
-h	halt or power off after shutdown
-H	halt after shutdown (implies -h)

```
-P          power off after shutdown (implies -h)
-c          cancel a running shutdown
-k          only send warnings, don't shutdown
-q, --quiet reduce output to errors only
-v, --verbose increase output to include informational messages
--help      display this help and exit
--version   output version information and exit
```

TEMPS peut avoir différents formats, le plus commun étant simplement le mot 'now' qui va éteindre le système immédiatement. Les autres formats valides sont +m, où m est le nombre de minutes à attendre avant d'éteindre et hh:mm qui spécifie l'heure d'extinction (sur 24 heures).

Les utilisateurs connectés sont avertis par un message envoyé sur leurs terminaux, que vous pouvez compléter par un MESSAGE optionnel. Les messages peuvent être envoyés sans éteindre effectivement le système en utilisant l'option -k.

Si TEMPS est spécifié, la commande restera au premier plan jusqu'à ce que l'arrêt se produise. Elle peut être annulée par Ctrl-C, ou par un autre utilisateur en utilisant l'option -c.

Par défaut, le système passe en mode mono-utilisateur (niveau d'exécution 1). Vous pouvez changer ce comportement avec les options -r ou -h, qui indiquent respectivement que l'on souhaite redémarrer ou éteindre l'ordinateur. L'option -h peut être précisée avec -H ou -P pour soit aller à l'état "Halt", soit aller jusqu'à mettre le système hors tension. L'option par défaut dépend des scripts d'arrêt.

Report bugs to <upstart-devel@lists.ubuntu.com>

Parmi les options les plus importantes, on note :

Option	Description
-h	Arrêter le système
-r	Re-démarrer le système
-c	Annuler l'opération shutdown en cours
-f	Re-démarrer rapidement sans vérifier les systèmes de fichiers
-F	Forcer la vérification des systèmes de fichiers lors du prochain démarrage

L'option **time** peut prendre plusieurs valeurs :

Valeur	Description
hh:mm	L'heure à laquelle l'opération aura lieu
+m	Nombre de minutes avant que l'opération aura lieu
now	L'opération est immédiate



Si l'opération est programmée pour dans moins de 5 minutes, les connexions supplémentaires sont interdites, y comprises les tentatives de connexion de root.

L'utilisation de la commande **shutdown** peut être accordée à d'autres utilisateurs de root en utilisant le fichier **/etc/shutdown.allow**

La commande reboot

Cette commande redémarre le système. Quand le système fonctionne normalement, l'exécution de reboot appelle la commande **shutdown -r**.

Options de la commande

Les options de cette commande sont :

```
root@ubuntu:/etc/init.d# reboot --help
```

Utilisation: reboot [OPTION]...
Redémarrer le système.

Options:

-n, --no-sync	don't sync before reboot or halt
-f, --force	force reboot or halt, don't call shutdown(8)
-p, --poweroff	switch off the power when called as halt
-w, --wtmp-only	don't actually reboot or halt, just write wtmp record
-q, --quiet	reduce output to errors only
-v, --verbose	increase output to include informational messages
--help	display this help and exit
--version	output version information and exit

Cette commande permet de demander au noyau de redémarrer ou d'arrêter le système; lorsqu'elle est exécutée sans l'option -f, ou si le niveau d'exécution du système est différent de 0 ou 6, c'est la commande /sbin/shutdown qui est lancée.

Report bugs to <upstart-devel@lists.ubuntu.com>

La commande halt

Cette commande arrête le système. Quand le système fonctionne normalement, l'exécution de halt appelle la commande **shutdown -h**.

Options de la commande

Les options de cette commande sont :

```
root@ubuntu:/etc/init.d# halt --help
Utilisation: halt [OPTION]...
Arrêter le système.
```

Options:

-n, --no-sync	don't sync before reboot or halt
-f, --force	force reboot or halt, don't call shutdown(8)
-p, --poweroff	switch off the power when called as halt
-w, --wtmp-only	don't actually reboot or halt, just write wtmp record
-q, --quiet	reduce output to errors only
-v, --verbose	increase output to include informational messages
--help	display this help and exit
--version	output version information and exit

Cette commande permet de demander au noyau de redémarrer ou d'arrêter le système; lorsqu'elle est exécutée sans l'option -f, ou si le niveau d'exécution du système est différent de 0 ou 6, c'est la commande /sbin/shutdown qui est lancée.

Report bugs to <upstart-devel@lists.ubuntu.com>

La commande poweroff

Cette commande arrête le système et coupe l'alimentation électrique. Elle est l'équivalente de la commande **halt -p**. Quand le système fonctionne normalement, l'exécution de **poweroff** appelle la commande **shutdown -hP**.

Options de la commande

Les options de cette commande sont :

```
root@ubuntu:/etc/init.d# poweroff --help
Utilisation: poweroff [OPTION]...
Éteindre le système.
```

Options:

-n, --no-sync	don't sync before reboot or halt
-f, --force	force reboot or halt, don't call shutdown(8)
-p, --poweroff	switch off the power when called as halt
-w, --wtmp-only	don't actually reboot or halt, just write wtmp record
-q, --quiet	reduce output to errors only
-v, --verbose	increase output to include informational messages
--help	display this help and exit
--version	output version information and exit

Cette commande permet de demander au noyau de redémarrer ou d'arrêter le système; lorsqu'elle est exécutée sans l'option -f, ou si le niveau d'exécution du système est différent de 0 ou 6, c'est la commande /sbin/shutdown qui est lancée.

Report bugs to <upstart-devel@lists.ubuntu.com>

<html> <center> Copyright © 2004-2016 Hugh Norris.

Ce(tte) oeuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 3.0 France. </center> </html>
