

# Administration & Sécurité des Systèmes d'Exploitation

## Chapitre 1



# Démarrage de Linux & Gestionnaire d'amorçage

Unité Pédagogique Systèmes  
2017-2018



# Gestionnaire d'amorçage (boot manager)

## Objectif :

- Comprendre comment démarre Linux
- Sélectionner, installer et configurer un gestionnaire d'amorçage.

# Introduction

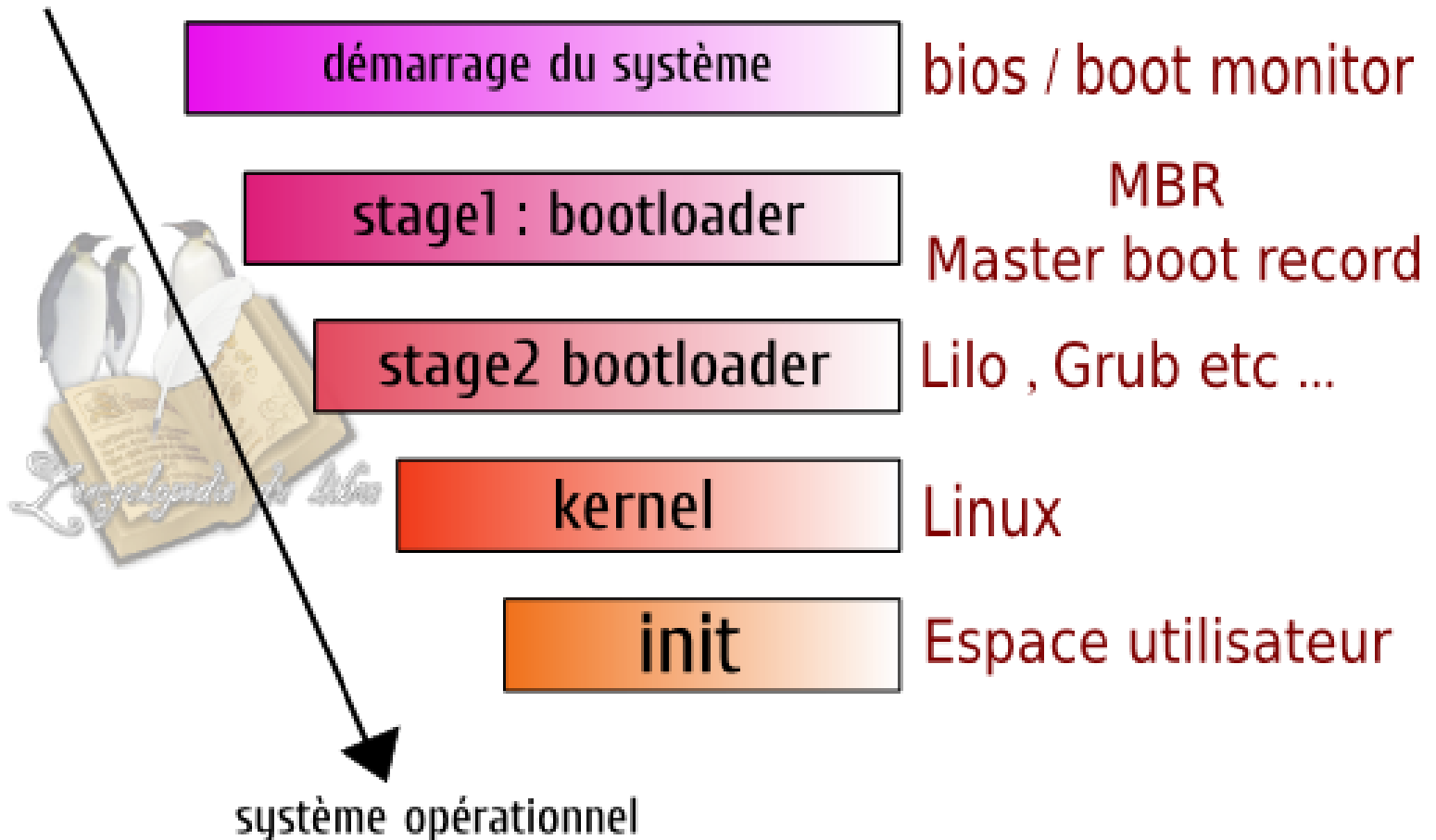
La séquence de démarrage est variable en fonction du système mais peut globalement être découpée selon les étapes suivantes :

- Le démarrage de l'ordinateur ou amorçage,
- L' exécution du chargeur de démarrage ,
- Le démarrage du noyau,
- Le lancement du processus 'init',
- Le lancement des scripts de démarrage.

# Gestionnaire d'amorçage (boot manager)

## The Big Picture !

mise sous tension  
reset



# Démarrage de l'ordinateur

## *Amorçage matériel*

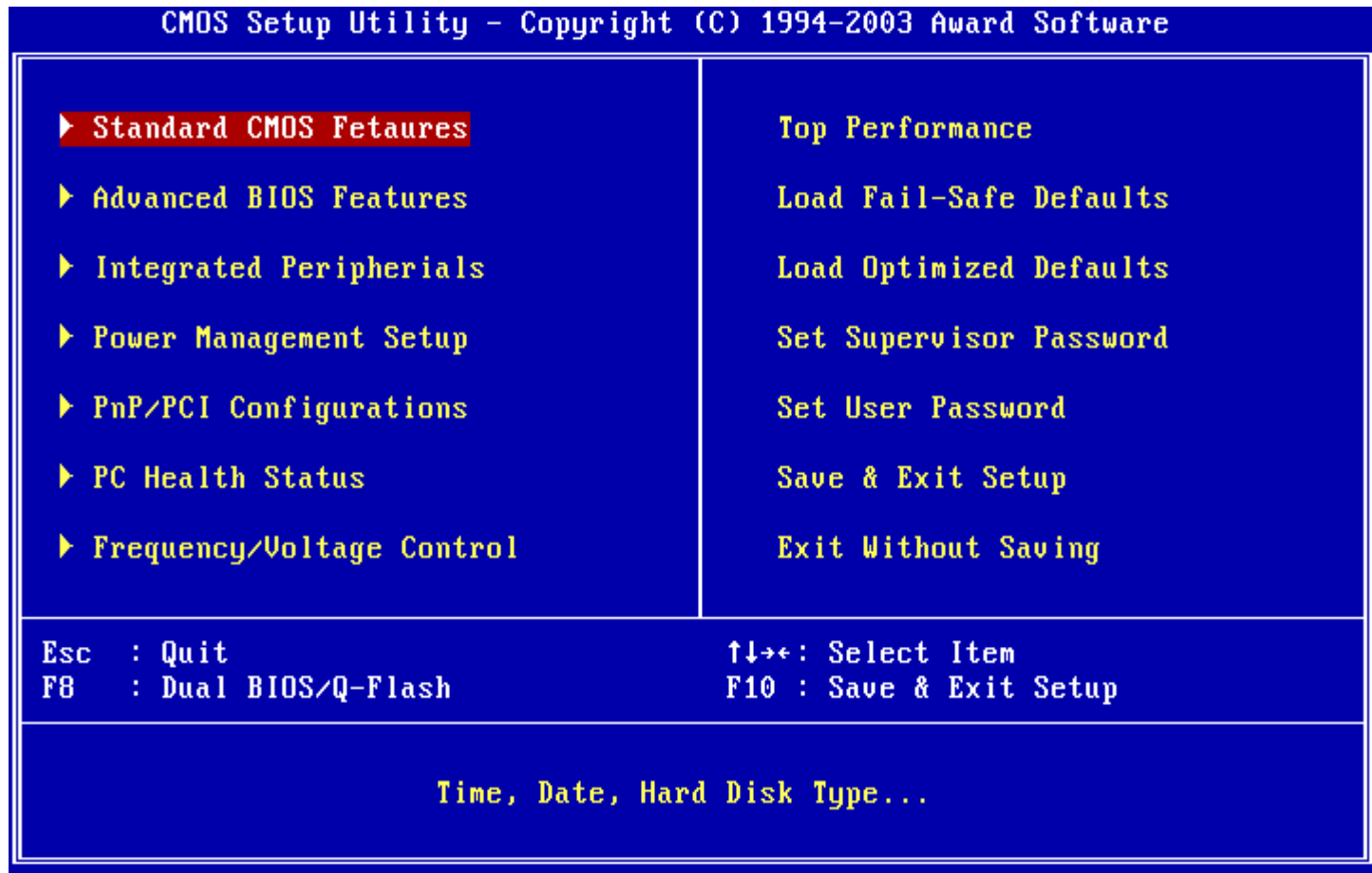
- Après la mise sous tension, un programme stocké en mémoire morte sur la carte mère prend le contrôle.
- Sur les PC, on appelle ce programme le BIOS (*Basic Input/Output System*).
- Ce programme procède en premier lieu à un autotest de la machine

C'est le POST - Power On Self Test

# Démarrage de l'ordinateur

## Le Bios

C'est le programme basique servant d'interface entre le système d'exploitation et la carte mère.



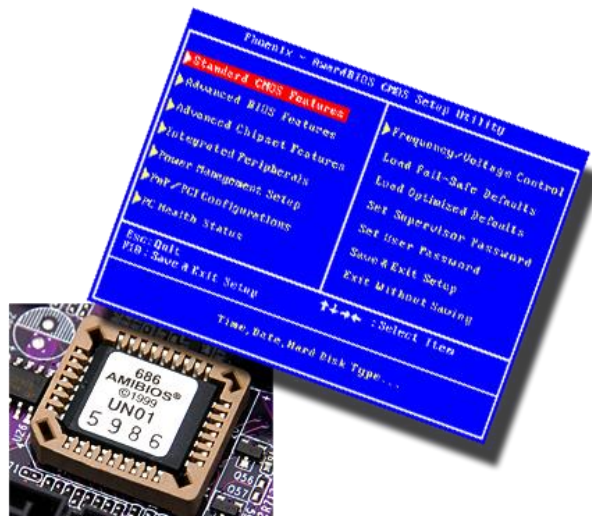


# Démarrage de l'ordinateur

## Le Bios

Il est stocké dans une *ROM* (mémoire morte, c'est-à-dire une mémoire en lecture seule)

Ainsi il utilise les données contenues dans le *CMOS* pour connaître la configuration matérielle du système.



# Gestionnaire d'amorçage (boot manager)

## Rôle du Bios

Indépendant du système d'exploitation (Windows, Mac OS, GNU/Linux), le BIOS est le **chef d'orchestre** de la partie matérielle de votre machine.

C'est depuis son interface que l'on décide, par exemple, d'amorcer la machine depuis le disque dur ou depuis un CD pour réinstaller sa machine.

il détecte les éventuels problèmes matériels et communique au travers de séquences de « **bip** » afin d'indiquer la nature des erreurs (mémoire défectueuse, disque dur endommagé, etc.).



# Gestionnaire d'amorçage (boot manager)

## 1- Amorçage de Linux : étape du Bios

Quand un système démarre, ou est redémarré, le processeur exécute du code à une **adresse fixe**.

Cette adresse fixe est celle du **BIOS** (*Basic Input/Output System*), qui est stocké dans une ROM sur les cartes mères.

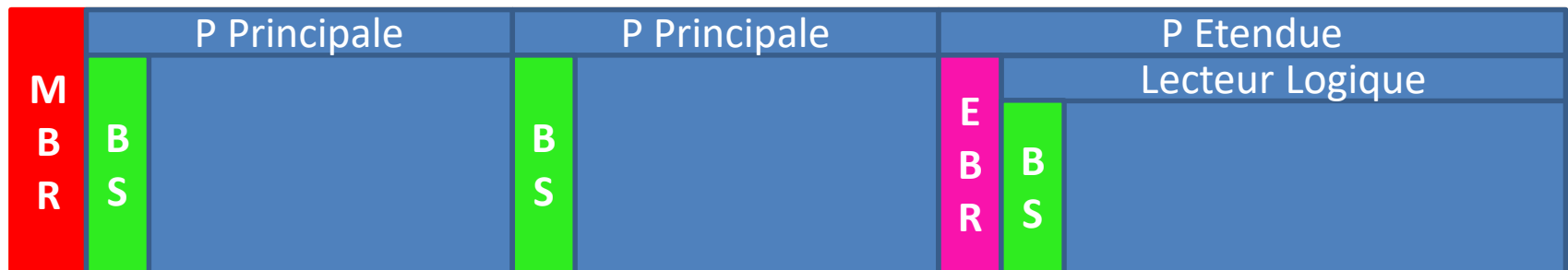
Le BIOS doit déterminer **quels périphériques** sont candidats pour démarrer dessus.

# Gestionnaire d'amorçage (boot manager)

## 2- Amorçage de Linux: étape du MBR

Quand un périphérique sur lequel on peut démarrer est trouvé, le **premier programme se trouvant dans le MBR** est chargé en RAM puis exécuté.

Ce chargeur de démarrage fait au plus **512 octets** (un secteur) et son rôle est de charger le deuxième programme.



# Chargeur de démarrage

## *Structure du MBR*



Les 446 premiers octets contiennent le programme de chargement initial « boot loader » qui va lancer le chargeur secondaire .

# Chargeur de démarrage

Le programme de partition ou **Master Boot Code** contenu dans le MBR réalise les opérations suivantes :

- Passage en revue de la table des partitions pour déterminer la partition active,
- Détermination de l'adresse du secteur de début de la partition active,
- Chargement d'une partie du bloc de boot de la partition active en mémoire,
- Transfert du contrôle au bloc de boot de la partition active.

# Chargeur de démarrage

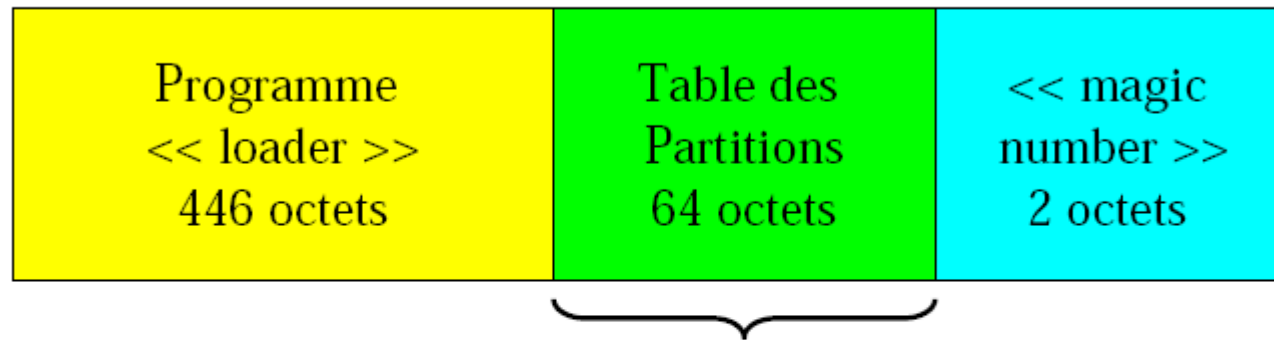
Si le Master Boot Code ne peut réaliser une de ces opérations, on obtient un des messages suivants :

- Invalid partition table.
- Error loading operating system.
- Missing operating system.



# Chargeur de démarrage

## *Structure du MBR*



Les 64 octets qui suivent décrivent les **partitions** :

- ❖ Taille, localisation, type et statut (16 octets).
- ❖ Cette table ne contient que quatre entrées, donc un disque ne peut contenir que quatre partitions dites primaires ou principales.

# Chargeur de démarrage

## *Structure du MBR*



Les deux derniers octets constituent  
le 'magic number'  
(une valeur numérique que certains systèmes utilisent  
pour vérifier la signature du secteur).

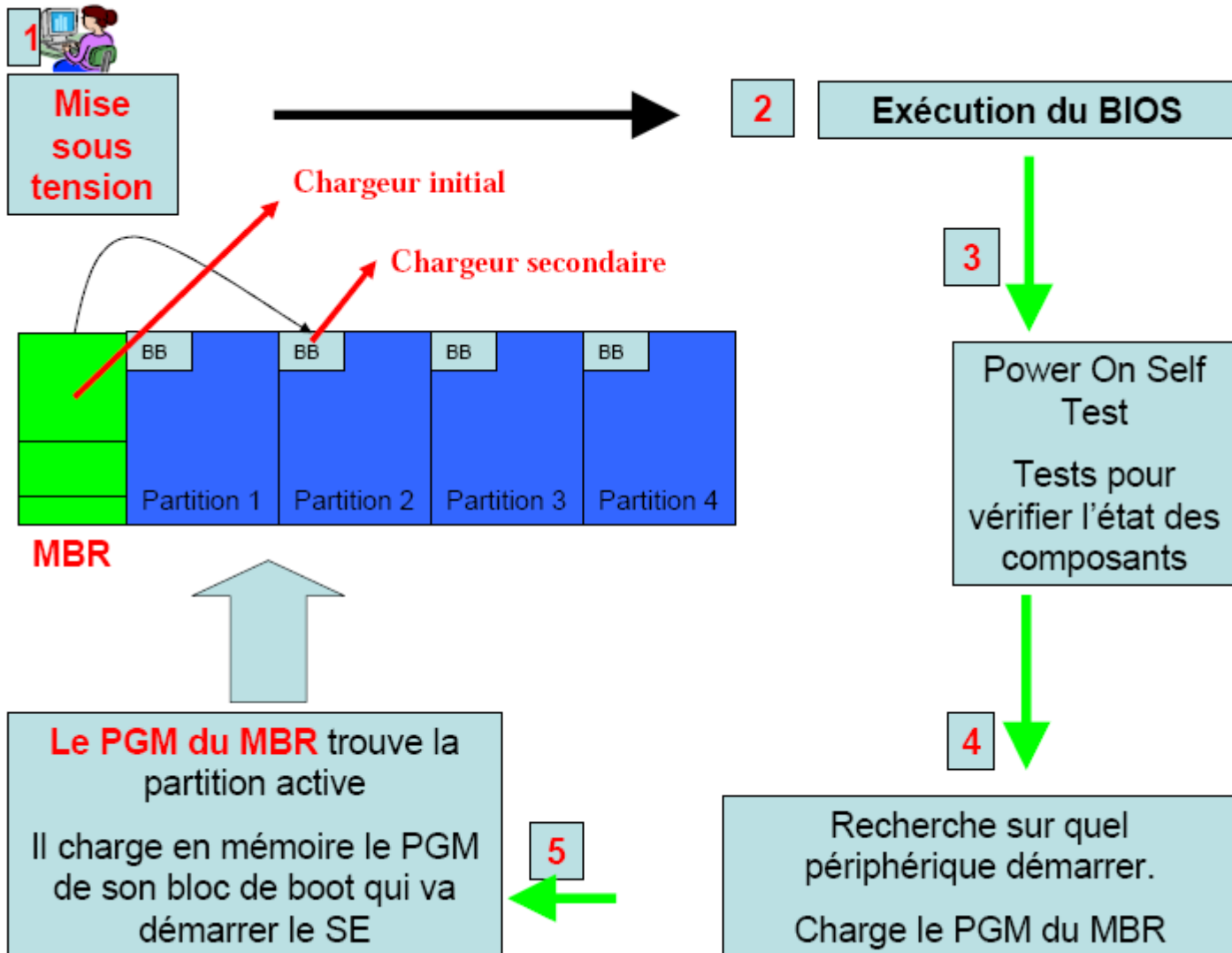
Les secteurs de boot sont marqués par ce que l'on nomme le «magic number ». Il s'agit d'un identifiant caractéristique mis à la valeur fixe 0xAA55 qui permet à la machine de déterminer s'il s'agit d'un secteur d'amorce ou pas.

# Chargeur de démarrage

Sur les PC, le chargeur de démarrage est **situé sur le premier secteur du périphérique d'amorçage**, c'est le MBR (Master Boot Record).

- La taille de ce MBR (un secteur soit **512** octets) rend quasiment **impossible le stockage** d'un chargeur de démarrage complet.
- Ainsi, sur la plupart des systèmes, **le chargeur initial appelle un chargeur de démarrage secondaire** situé sur une partition du disque.

# Chargeur de démarrage



# Chargeur de démarrage

- Le rôle principal du chargeur de démarrage est de **localiser le noyau du système** d'exploitation sur le disque, **le charger et l'exécuter**.
- La majorité des chargeurs de démarrage sont interactifs, pour permettre la spécification d'un noyau alternatif (par exemple un noyau de sauvegarde dans le cas où la dernière version compilée ne fonctionne pas) ou le passage de paramètres optionnels au noyau.



# Gestionnaire d'amorçage (boot manager)

## Les Familles de boot loader:

### Microsoft :

NTLDR (Windows NT)

*WINLOAD (Vista)*

### Open source :

GRUB (GRand Unified Bootloader)

LILO (Linux loader)

### Apple :

Boot Camp

# Chargeur de démarrage

Sous Linux, les chargeurs de démarrage sont généralement

**Grub** - GRand Unified Bootloader

**Lilo** - LInux LOader

# Chargeur de démarrage

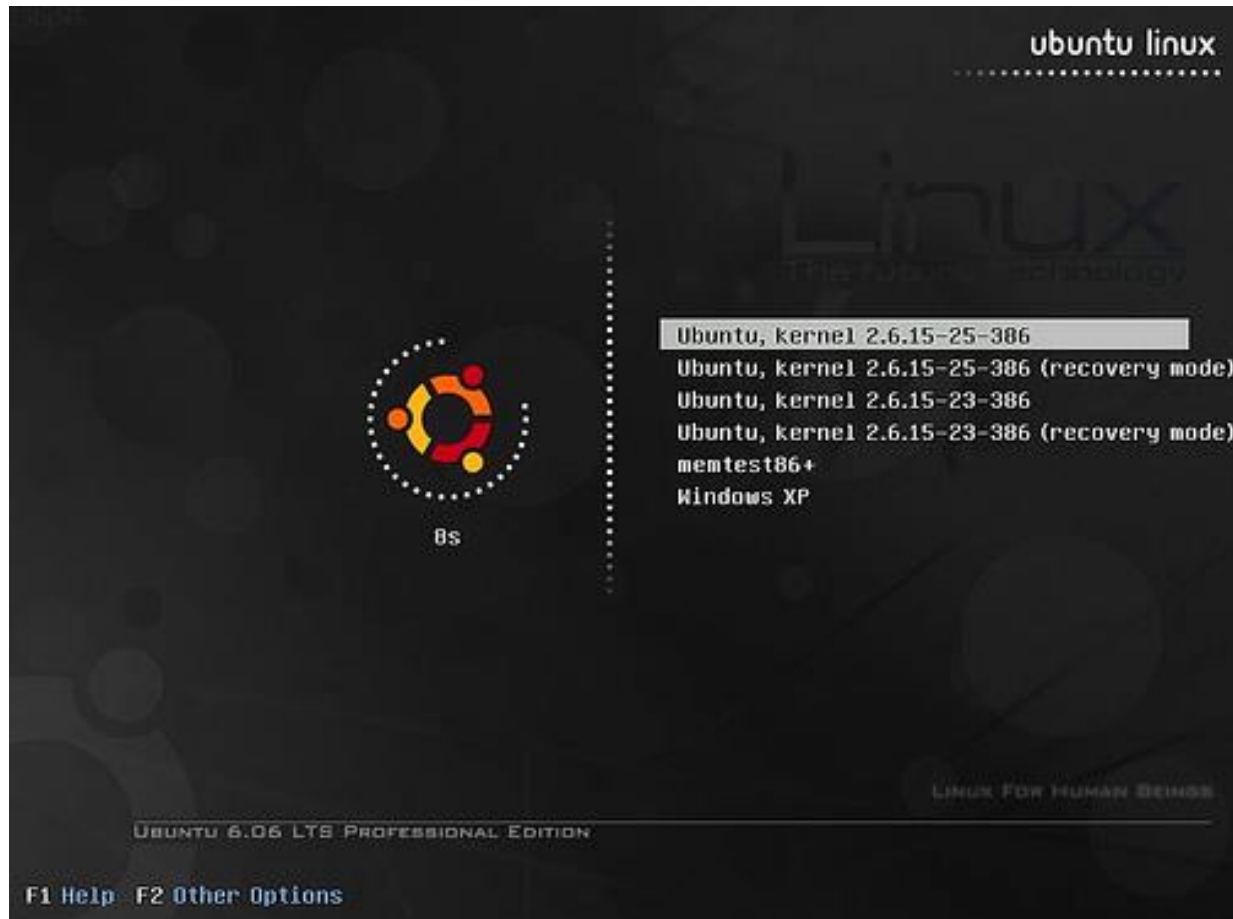
Chacun peut être installé :

- soit en tant que chargeur secondaire (si le MBR installé par DOS pointe vers eux),
- soit en tant que chargeur en deux parties (chargeur initial + chargeur secondaire).

# Chargeur de démarrage

## 3- Amorçage de Linux: Le boot loader

Quand le boot loader est chargé en RAM et exécuté, un *splash-screen* est souvent affiché proposant les images d'OSs disponibles



# Gestionnaire d'amorçage (boot manager)

## Caractéristiques du boot manager:

Un gestionnaire d'amorçage permet de sélectionner l'image à partir de laquelle on désire démarrer.

On peut éventuellement lui passer des paramètres.

Pour Linux, il s'agit de déterminer le noyau sur lequel on veut démarrer et de lui passer des paramètres (comme la partition à utiliser comme racine).



# Gestionnaire d'amorçage (boot manager)

## Cohabitation des boot manager:

- De nombreux systèmes d'exploitations installent leur propre gestionnaire de démarrage sans se préoccuper de l'existence d'autre gestionnaire.
- Ainsi, si on veut faire cohabiter linux avec d'autres systèmes, il est plus sûr de l'installer en dernier.
- **GRUB** est le gestionnaire par défaut de nombreuses distributions linux actuelles.

# Gestionnaire d'amorçage (boot manager)

## GRUB vs LILO:

### Avantages de Grub :

- + Possède un invite de commandes.
- + Plus sécurisé
- + Grub conserve les informations sur le BOOT dans le système de fichiers , donc possibilité d'extension
- ne supporte pas le chargement depuis LVM ou RAID ou des systèmes de fichiers très spécifiques.

### Limites de LILO:

- LILO conserve les informations sur le BOOT dans le MBR → impossible d'ajouter des nouvelles fonctionnalités.

# Gestionnaire d'amorçage (boot manager)

**LILO**

# Gestionnaire d'amorçage (boot manager)

## Installer un boot manager: Lilo

Fichier de configuration: **/etc/lilo.conf**

Pour l'installer LILO dans le MBR, il faut utiliser la commande **lilo**

La commande lilo permet d'écrire dans le MBR les éléments contenus dans le fichier **/etc/lilo.conf**.

# Gestionnaire d'amorçage (boot manager)

## Exemple de contenu de /etc/lilo.conf :

### # LILO global section

```
boot = /dev/hda # Cible d'installation de LILO : le MBR
timeout=15 # le temps d'attente
default=fedora # l'OS à démarrer après le timeout
vga = normal # (normal, extended, ou ask)
read-only # Monte le système de fichiers en lecture seule
```

### # LILO Linux section

```
image=/boot/vmlinuz # Image (noyau) à charger
label=fedora # Nom de l'entrée du menu
root=/dev/hda1 # Partition racine pour le noyau
initrd=/boot/initrd # disque en RAM
```

### # LILO DOS/Windows section

```
other=/dev/hda3 #utilisé pour les OS non LINUX
label=windows
```



# Gestionnaire d'amorçage (boot manager)

## Protection de Lilo

Éditer le fichier de configuration **lilo.conf** en y ajoutant une clause **password** avant la définition des images :

*password = mot\_de\_passe*

**Inconvénient :** *Ce mot de passe est écrit en clair et peut être lu par n'importe quelle personne qui se connecte à la machine*

Pour y remédier on peut retirer le droit de lecture à tout autre utilisateur (excepté root bien évidemment).

```
# chown root.root /etc/lilo.conf
```

```
# chmod 600 /etc/lilo.conf
```

# Gestionnaire d'amorçage (boot manager)

**GRUB**

# Gestionnaire d'amorçage (boot manager)

## Installer un boot manager: Grub

**Fichier de configuration:** `/boot/grub/menu.lst` ou `/boot/grub/grub.conf`

Pour l'installer Grub dans le MBR, il faut utiliser la commande **grub**

La commande grub permet d'écrire dans le MBR les éléments contenus dans le fichier `/boot/grub/menu.lst` ou `grub.conf`

# Gestionnaire d'amorçage (boot manager)

## Exemple de contenu de /boot/grub/menu.lst :

```
# GRUB default values
```

```
timeout 10 # Démarrer le noyau par défaut après 10 secondes.
```

```
default 0 # Noyau par défaut.
```

```
# Grub for Linux section 0
```

```
title GNU/Linux # Titre
```

```
root (hd0,1) # /dev/hda2 système de fichiers racine
```

```
# Noyau et paramètres à passer au noyau.
```

```
kernel /boot/vmlinuz root=/dev/hda2 read-only
```

```
initrd /boot/initrd # (INITial RamDisk) image d'un système minimal initialisé  
au démarrage du système.
```

```
boot
```

```
# Grub for DOS/Windows section
```

```
title Windows
```

```
root (hd0,2) # /dev/hda3
```

```
makeactive # Positionnez le drapeau active de la partition
```

```
chainloader+1 # Chargez le gestionnaire d'amorçage
```

# Gestionnaire d'amorçage (boot manager)

## Convention de nommage

La syntaxe des périphériques utilisée dans GRUB est un tout petit peu différente de ce que vous avez pu voir.

### Exemple : (hd0,1)

Tout d'abord, GRUB exige que les noms de périphériques se trouvent entre ( et ).

**hd** signifie qu'il s'agit d'un disque dur.

Le premier nombre **0** indique le **numéro du disque**, qui est ici le premier disque, alors que le second entier **1** indique le **numéro de la partition**

Notez que les numéros de partitions sont déterminés à partir de **zéro**, et non depuis un.

# Gestionnaire d'amorçage (boot manager)

## Convention de nommage

**(hd0,4)**

Ceci désigne le premier lecteur logique du premier disque dur. Notez que les numéros des lecteurs logiques sont comptés à partir de 4.

Remarquez que GRUB **ne distingue pas** l'IDE du SCSI, il compte simplement les disques depuis zéro, sans faire attention à leur type.

**(hd0,0)/vmlinuz**

Cette ligne désigne le fichier nommé **vmlinuz** qui se trouve sur la première partition du premier disque dur.



# Gestionnaire d'amorçage (boot manager)

## Protection du grub

Cela est fortement recommandée car GRUB permet d'offrir un accès à une interface de type shell permettant de modifier la configuration de celui ci, d'obtenir des informations système et de booter en single user mode

### Protection de GRUB avec un mot de passe crypté

```
#grub
grub> md5crypt
password : *****
Encrypted : kw485/fgf$&ee
```

On doit copier le code encrypté et l'insérer dans le fichier de configuration /etc/grub.conf, avant la définition des images :

```
Password --md5 kw485/fgf$&ee
```

Pour protéger une entrée en particulier, il faut mettre **lock** après le **title** de l'entrée

# Gestionnaire d'amorçage (boot manager)

## Réinstallé lu grub

Pour installer Grub dans l'MBR il faut tout d'abord :

Trouver le numéro de la partition sur laquelle est installée LINUX, de la forme (hd0,5).

**# grub**

**grub> find /boot/grub/stage1**

Renseignez la partition sur laquelle est installée Grub dans notre exemple :  
root (hd0,5)

**grub> root (hd0,5)**

déterminez où installer Grub :

**grub> setup (hd0)**

installe Grub dans le MBR

# Administration & Sécurité des Systèmes d'Exploitation

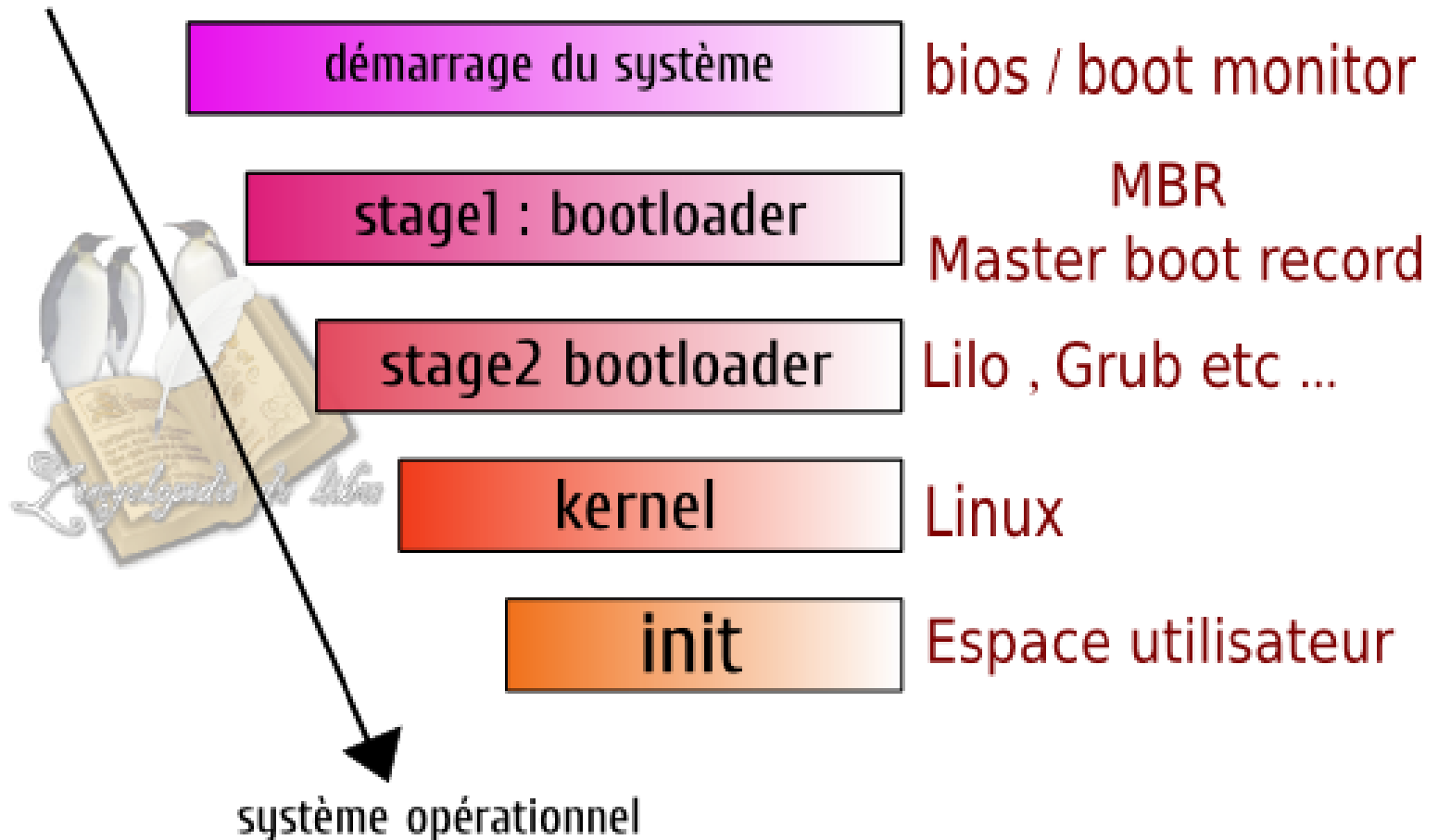


## Les RunLevel et le processus de démarrage init

Unité Pédagogique Systèmes  
2017-2018

# Init

mise sous tension  
reset



# Lancement du 1er processus *init*

Lorsque la machine démarre, le noyau du système est **chargé** et **décompressé** en mémoire vive (RAM), il **s'exécute** et s'initialise :

- Réservation de la mémoire
- Prise en compte de la zone d'échange (swap)
- Détection du matériel et chargement des pilotes des périphériques
- Montage du système de fichiers
- Et enfin lancement du 1er processus *init*.

# Lancement du 1er processus *init*

**Init** est l'unique processus **lancé directement par le kernel**, c'est le **père** de tous les autres processus (**pid=1**) Il a pour tâche de **lancer** chacun des processus, démons, sessions de login et de **gérer** l'arrêt du système

```
$ pstree
init--+-automount
      |-crond
      |-3*[gvim]
      |-identd---identd---3*[identd]
      |-inetd
      |-kdm--+-X
      |      |-kdm
      |      `--kdm---kwm--+-kbgndwm
      |                      |-kfm--+-konsole---bash---su---bash
      |                      |      `--konsole---bash---pstree
      |                      |-kpanel
      |                      |-krootwm
      |                      `--kwmsound
      |-kflushd
      |-kikbd
      |-klogd
      |-kpiod
      |-kswapd
```



# Rôle du processus *init*

Il **vérifie** le système de fichier et **le monte**

Il **démarre les démons** qui enregistrent les messages système (syslog, cron, ...), **gèrent le réseau**, écoutent les **signaux** de la souris et clavier.

Démarre les **processus getty** qui vous donnent l'invite de connexion sur vos terminaux.

Ainsi que d'autres fonctions pour lesquelles il a été configuré.

# Les RunLevel

Linux Offre plusieurs **modes d'exploitation** appelés **runlevel**.

A chaque runlevel correspond a un **état** dans lequel se trouve le système.

Il y'a en tous **7 états** possibles (de 0 à 6).

Chaque état est caractérisé par sa propre configuration.

# Les RunLevel

Un niveau d'exécution : correspond à **une configuration** logicielle qui permet de **lancer** un certain nombre de processus.

- **Mode d'arrêt (init 0)**
- **Mode simple utilisateur (init 1)**
- **Mode multi-utilisateurs (init 2)**
- **Mode multi-utilisateurs + réseau (init 3)**
- **Mode multi-utilisateurs + réseau + Graphique (init 5)**
- **Mode redémarrage (init 6)**

# Le processus init

## *Les différents niveaux d'exécution*

0	Arrêter le système.
1	mode mono-utilisateur (console).
2	Mode multi-utilisateurs. Les systèmes de fichiers sont montés. Le service réseau est démarré.
3	C'est un sur ensemble du niveau 2. Il est associé au démarrage des services de partage à distance.
4	Mode multi-utilisateurs spécifique au site informatique.

# Le processus init

## *Les différents niveaux d'exécution*

5	C'est un sur ensemble du niveau 3. Interface X-Window (graphique).
6	Redémarrer le système.
s,S	Mode mono-utilisateur (linux single). Les systèmes de fichiers sont montés. Seuls les processus fondamentaux pour le bon fonctionnement du système sont activés. Un shell en mode 'root' est activé sur une console. Le répertoire "etc" n'est pas indispensable.

# Le processus init

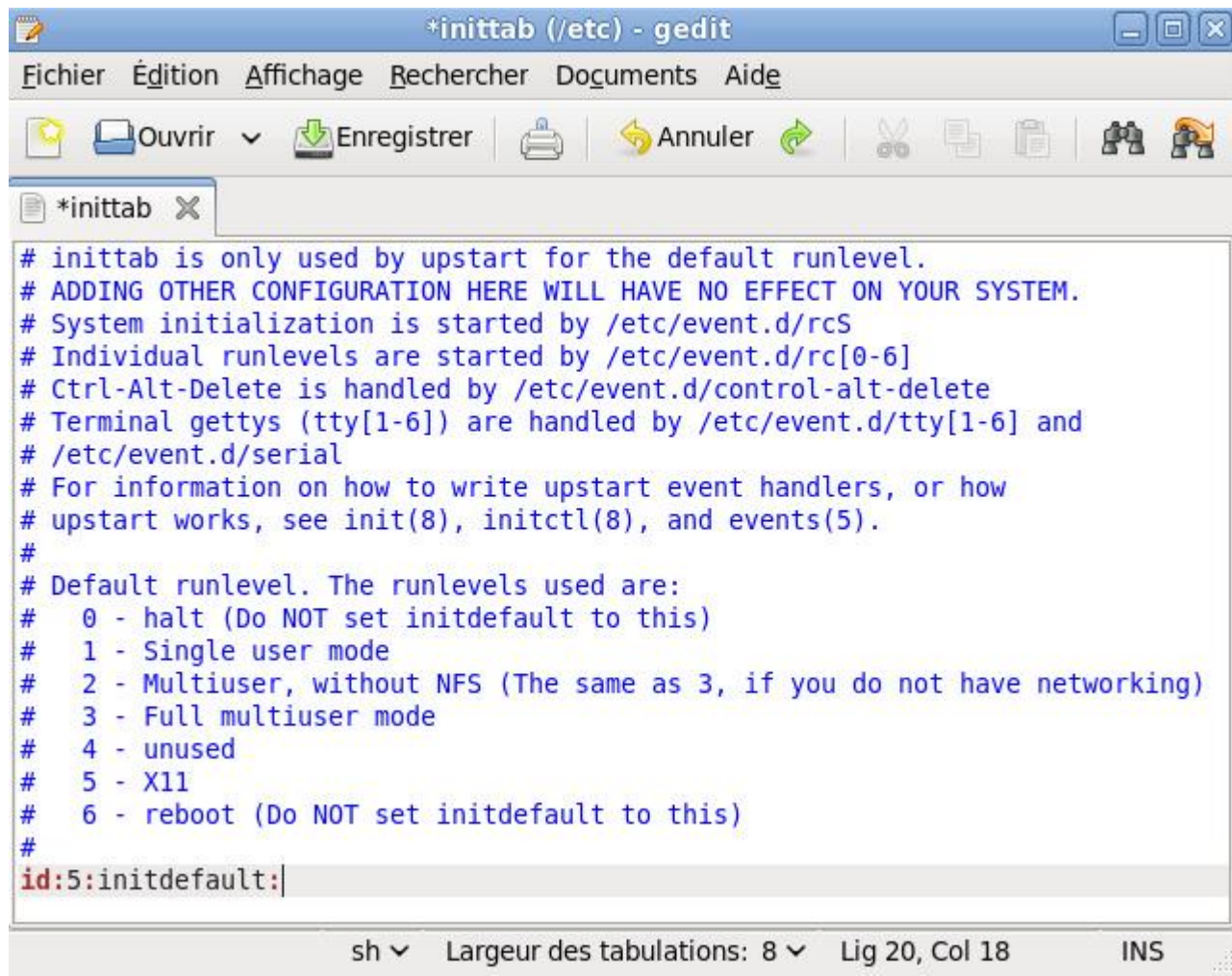
## *Le processus 'init' et le fichier /etc/inittab*

- Quand "init" démarre, il lit le fichier de configuration *'/etc/inittab'* pour y trouver ses instructions.
- Ce fichier définit ce qui doit s'exécuter dans les différents niveaux d'exécution (run-level) offerts par un système d'exploitation Unix/Linux.
- Cela donne à l'administrateur système une méthode simple de gestion dans laquelle chaque niveau d'exécution est associé à un ensemble de services devant s'exécuter.



# Le fichier inittab

Le niveau d'exécution par default est positionné dans le fichier **/etc/inittab** sur la ligne **initdefault**



```
*inittab (/etc) - gedit
Fichier Édition Affichage Rechercher Documents Aide
Ouvrir Enregistrer Annuler
*inittab x
# inittab is only used by upstart for the default runlevel.
# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
# System initialization is started by /etc/event.d/rcS
# Individual runlevels are started by /etc/event.d/rc[0-6]
# Ctrl-Alt-Delete is handled by /etc/event.d/control-alt-delete
# Terminal gettys (tty[1-6]) are handled by /etc/event.d/tty[1-6] and
# /etc/event.d/serial
# For information on how to write upstart event handlers, or how
# upstart works, see init(8), initctl(8), and events(5).
#
# Default runlevel. The runlevels used are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
sh Largeur des tabulations: 8 Lig 20, Col 18 INS
```

Ce fichier est  
remplacé par  
**upstart** dans  
quelque  
nouvelles  
distributions

# Le processus init

## *Démarrage et changement de niveau*

- Au démarrage du système, lorsque le noyau crée le processus 'init', il ne spécifie pas de niveau.
- C'est le niveau défini par défaut dans 'inittab' qui est choisi.
- Si on change de niveau de fonctionnement 'init' envoie le signal 'SIGTERM' (le signal 15) à tous les processus qui sont pas concernés par le nouveau niveau demandé.

Au bout de 5 secondes, les processus qui ne sont pas terminés reçoivent le signal 'SIGKILL' (signal 9).

# Le processus init

## *La commande init*

- permet changer le niveau d'exécution courant

- Syntaxe

init [-options] [0123456Ss]

- Exemple

```
[root]# init 5
```

## *La commande runlevel*

- Permet de connaître le niveau d'exécution courant

- Syntaxe

runlevel [-options]

- Exemple

```
[root]# runlevel
```

# Le processus init

## *La commande init avec l'option 'q'*

- permet prendre en compte immédiatement les modifications apportées à ce fichier

- Syntaxe

`init [-options] [0123456Ss]`

- Exemple

**`[root]# init q`**

## *Description du fichier '/etc/inittab'*

- Syntaxe

`étiquette:niveau(x):action:commande`

# Le processus init

## *Description du fichier '/etc/inittab'*

étiquette	Chaîne de 0 à 4 caractères qui identifie la ligne.
niveau(x)	Niveau(x) d'exécution du processus associé. Il est possible de préciser plusieurs niveaux. Un champ vide est équivalent à tous les niveaux sauf 's'.
action	mot clé conditionnant le lancement du processus associé.
commande	commande ou script-shell lancé.



# Le processus init

## *Les actions du fichier '/etc/inittab'*

initdefault	Entrée de <i>inittab</i> lue par <i>init</i> par défaut au démarrage
sysinit	Exécuté une seule fois lors du démarrage à froid. Ces lignes sont exécutées pour tous niveaux.
wait	<i>init</i> lance le processus et attend sa terminaison avant de poursuivre la lecture d' <i>inittab</i> .
ctrlaltdel	<i>Exécuté si init reçoit SIGINT.</i>



# Le processus init

## *Les actions du fichier '/etc/inittab'*

powerfail	Commande exécutée quand init reçoit le signal SIGPWR. Un onduleur peut être à l'origine de ce signal qui déclenche un arrêt propre du système en cas de coupure de courant.
powerokwait	Même principe de fonctionnement que la commande powerfail mais init attend que l'exécution de la commande soit terminée pour poursuivre.
respawn	<i>init</i> relance automatiquement le processus si celui-ci meurt.
once	Processus non recréé une fois mort.

# Le processus init

## Le fichier '/etc/inittab'

**id:3:itdefault:**

lecture du niveau de démarrage par défaut.

#System initialization.

si::sysinit:/etc/rc.d/rc.sysinit

Commandes de contrôle et d'initialisation.

10:0:wait:/etc/rc.d/rc 0

11:1:wait:/etc/rc.d/rc 1

12:2:wait:/etc/rc.d/rc 2

13:3:wait:/etc/rc.d/rc 3

14:4:wait:/etc/rc.d/rc 4

15:5:wait:/etc/rc.d/rc 5

16:6:wait:/etc/rc.d/rc 6

Chargement du niveau de démarrage

# Le processus init

## *Le fichier '/etc/inittab'*

```
#Run gettys in standard runlevel
```

```
1:2345:respawn:/sbin/mingetty tty1
```

```
2:2345:respawn:/sbin/mingetty tty2
```

```
3:2345:respawn:/sbin/mingetty tty3
```

```
4:2345:respawn:/sbin/mingetty tty4
```

```
5:2345:respawn:/sbin/mingetty tty5
```

```
6:2345:respawn:/sbin/mingetty tty6
```

```
#Run xdm runlevel 5
```

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

- **Run gettys** est la partie qui permet de déclarer le terminaux accessibles avec les combinaisons de touches "alt+F1" ... "alt+F2".
- **Run xdm** est une ligne indispensable pour pouvoir travailler avec l'interface graphique.

# Le processus init

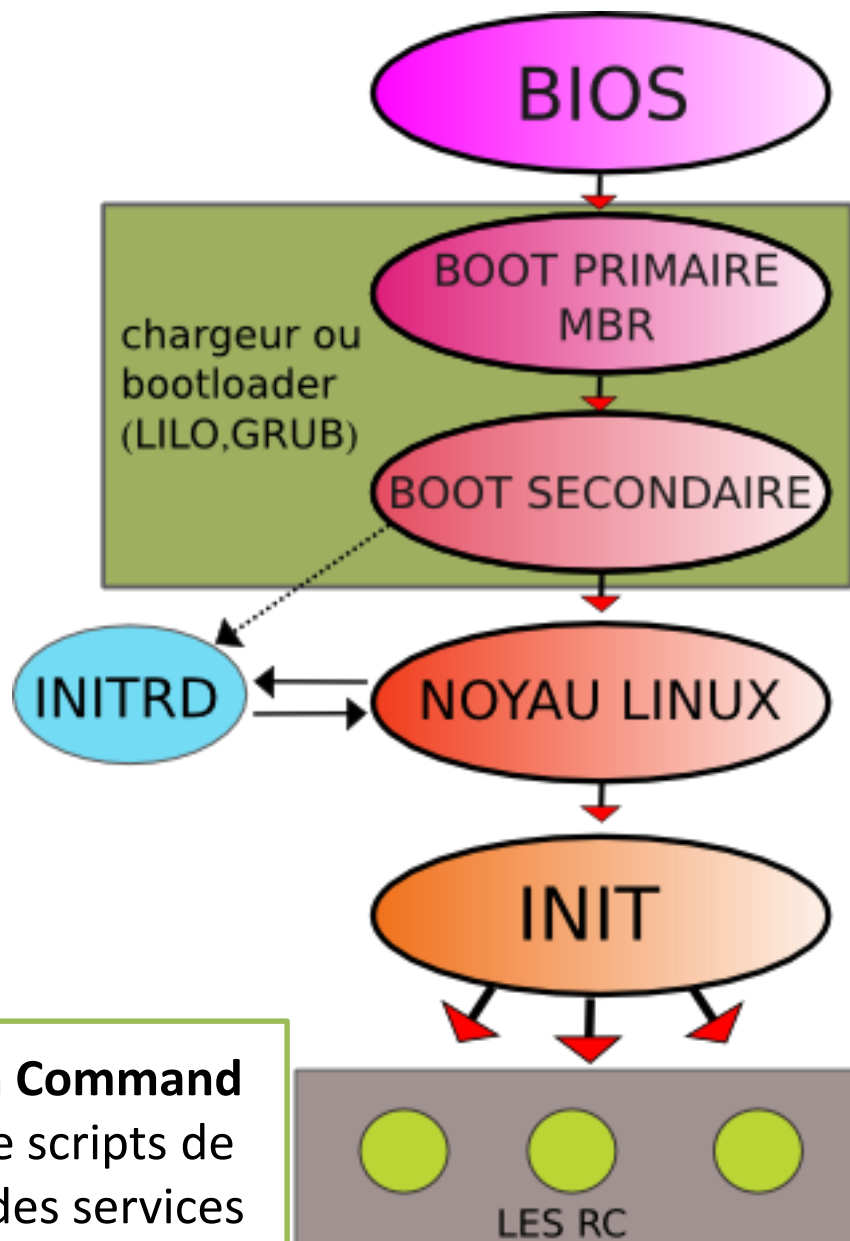
## *Le démarrage des démons associés aux services*

- Le processus 'init' lance en premier lieu le script '/etc/rc.d/rc.sysinit
- Ensuite il exécute le script '/etc/rc.d/rc' en lui passant en paramètre le niveau d'exécution demandé

## *Scripts de démarrage des services*

- Pour chaque service géré, il y a un script de démarrage stocké dans un répertoire spécifique (/etc/rc.d/init.d dans la majorité des versions de Linux).

# Amorçage de Linux (plus de détails)



**Les RC : Run Command**  
ensemble de scripts de  
démarrage des services



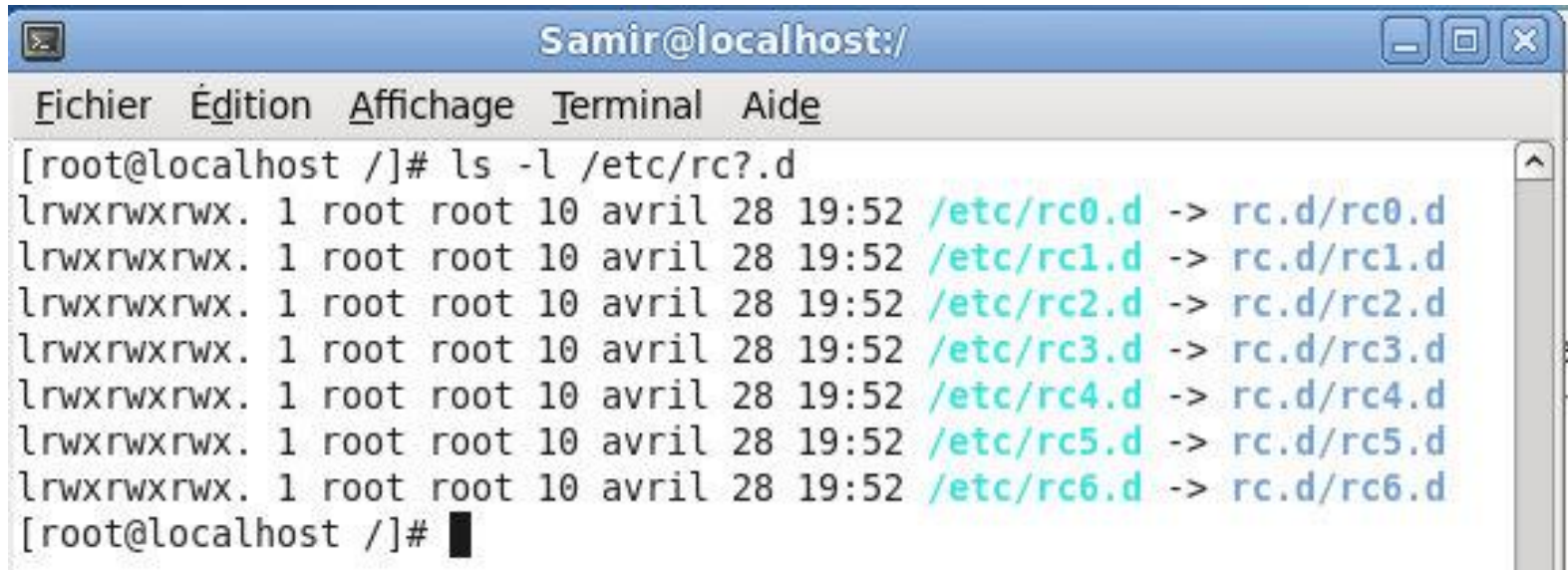
# Les rc

Dans le répertoire **/etc** se trouve une liste de répertoire commençant par **rc**.

**/etc/rc[0-6].d** ou **/etc/rc.d/rc[0-6].d** selon les distributions

## Exemple :

Le répertoire rc0.d contient tous les services du runlevel 0



```
Samir@localhost:/  
Fichier  Édition  Affichage  Terminal  Aide  
[root@localhost /]# ls -l /etc/rc?.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc0.d -> rc.d/rc0.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc1.d -> rc.d/rc1.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc2.d -> rc.d/rc2.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc3.d -> rc.d/rc3.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc4.d -> rc.d/rc4.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc5.d -> rc.d/rc5.d  
lrwxrwxrwx. 1 root root 10 avril 28 19:52 /etc/rc6.d -> rc.d/rc6.d  
[root@localhost /]#
```



# Nom des Scripts

[ S | K ] XX nom\_du\_script.

**S** indique au système que le script doit être lancé avec l'opérande start (démarrage du service).

**K** indique au système que le script doit être lancé avec l'opérande stop (arrêt du service).

**XX** est un numéro qui détermine l'ordre de lancement (ou d'arrêt) du script par rapport aux autres,

## Exemple :

– **S18sound** et **S24messagebus** => impliquent que le script **sound** sera lancé avant le script **messagebus**,

– **K08lisa** et **K09dm** => impliquent que le script **lisa** sera arrêté avant que le script **dm** ne soit lui-même arrêté.

Fichier Édition Affichage Terminal Aide

[root@localhost ~]# ls /etc/init.d

abrttd	bttrack	haldaemon	killall	netfs	ntpd	restorecond	sendmail	wpa_supplicant
acpid	cpuspeed	halt	lvm2-monitor	netplugd	ntpdate	rpcbind	smartd	ypbind
atd	crond	httpd	mdmonitor	network	openvpn	rpcgssd	smolt	
auditd	cups	ip6tables	messagebus	NetworkManager	pcscd	rpcidmapd	snmpd	
avahi-daemon	dnsmasq	iptables	microcode_ctl	nfs	portreserve	rpcsvcgssd	snmptrapd	
bluetooth	firstboot	irda	multipathd	nfslock	psacct	rsyslog	sshd	
btseed	functions	irqbalance	netconsole	nscd	rdisc	saslauthd	udev-post	

[root@localhost ~]# ls /etc/rc0.d

K01smartd	K16abrttd	K60crond	K74pcscd	K83rpcidmapd	K87multipathd	K90network	S01halt
K01smolt	K24irda	K60nfs	K75netfs	K84btseed	K87restorecond	K92ip6tables	
K02avahi-daemon	K25sshd	K69rpcsvcgssd	K75ntpdate	K84bttrack	K87rpcbind	K92iptables	
K05atd	K30sendmail	K73ypbind	K75udev-post	K84NetworkManager	K88auditd	K95firstboot	
K10cups	K50dnsmasq	K74acpid	K76openvpn	K84wpa_supplicant	K88rsyslog	K99cpuspeed	
K10psacct	K50netconsole	K74haldaemon	K83bluetooth	K85mdmonitor	K89netplugd	K99lvm2-monitor	
K10saslauthd	K50snmpd	K74nscd	K83nfslock	K85messagebus	K89portreserve	K99microcode_ctl	
K15httpd	K50snmptrapd	K74ntpd	K83rpcgssd	K87irqbalance	K89rdisc	S00killall	

[root@localhost ~]# ls /etc/rc1.d

K01smartd	K16abrttd	K60crond	K74pcscd	K84btseed	K87restorecond	K92ip6tables
K01smolt	K24irda	K60nfs	K75netfs	K84bttrack	K87rpcbind	K92iptables
K02avahi-daemon	K25sshd	K69rpcsvcgssd	K75ntpdate	K84NetworkManager	K88auditd	K95firstboot
K05atd	K30sendmail	K73ypbind	K76openvpn	K84wpa_supplicant	K88rsyslog	K99microcode_ctl
K10cups	K50dnsmasq	K74acpid	K83bluetooth	K85mdmonitor	K89netplugd	S02lvm2-monitor
K10psacct	K50netconsole	K74haldaemon	K83nfslock	K85messagebus	K89portreserve	S06cpuspeed
K10saslauthd	K50snmpd	K74nscd	K83rpcgssd	K87irqbalance	K89rdisc	S26udev-post
K15httpd	K50snmptrapd	K74ntpd	K83rpcidmapd	K87multipathd	K90network	

[root@localhost ~]# ls /etc/rc2.d

K01smartd	K24irda	K74haldaemon	K83rpcgssd	K89netplugd	S08iptables	S26acpid
K01smolt	K50dnsmasq	K74nscd	K83rpcidmapd	K89rdisc	S11auditd	S26pcscd
K02avahi-daemon	K50netconsole	K74ntpd	K84btseed	K90network	S11portreserve	S26udev-post
K05atd	K50snmpd	K75netfs	K84bttrack	K95firstboot	S12rsyslog	S27NetworkManager
K10psacct	K50snmptrapd	K75ntpdate	K84wpa_supplicant	S00microcode_ctl	S13rpcbind	S55sshd
K10saslauthd	K60nfs	K76openvpn	K87irqbalance	S02lvm2-monitor	S15mdmonitor	S80sendmail
K15httpd	K69rpcsvcgssd	K83bluetooth	K87multipathd	S06cpuspeed	S22messagebus	S90crond
K16abrttd	K73ypbind	K83nfslock	K87restorecond	S08ip6tables	S25cups	S99local

[root@localhost ~]#

# Chkconfig

La commande **chkconfig** affiche l'état d'un service pour chaque un des runlevel

## Exemple

# Chkconfig --list NetworkManager

Affiche l'état du service NetworkManager pour chaque Runlevel (marche ou arrêt)



A terminal window titled 'Samir@localhost:/' showing the command 'chkconfig --list NetworkManager' and its output. The output is a table with 7 columns representing runlevels 0 through 6. Runlevel 0 is 'arrêt', 1 is 'arrêt', 2 is 'marche', 3 is 'marche', 4 is 'marche', 5 is 'marche', and 6 is 'arrêt'.

```
Samir@localhost:/  
Fichier Édition Affichage Terminal Aide  
[root@localhost /]# chkconfig --list NetworkManager  
NetworkManager 0:arrêt 1:arrêt 2:marche      3:marche      4:marche      5:marche      6:arrêt  
[root@localhost /]#
```



# Chkconfig

**Pour démarrer le service « httpd » dans le niveau d'exécution 3 il faut lancer la cmd suivante :**

## **Exemple**

```
# Chkconfig - - level 3 httpd on
```

# Les commandes Start,Stop,Status

**#/etc/init.d/prog1 start**

⇒ pour démarrer le programme

**#/etc/init.d/prog1 stop**

⇒ pour arrêter le programme

**#/etc/init.d/prog1 status**

⇒ pour arrêter le programme

## **Exemple:**

/etc/init.d/syslog status

/etc/init.d/syslog stop

/etc/init.d/syslog start