Systèmes Avancés - 2019
Groupe “Amorce”
Présentation finale

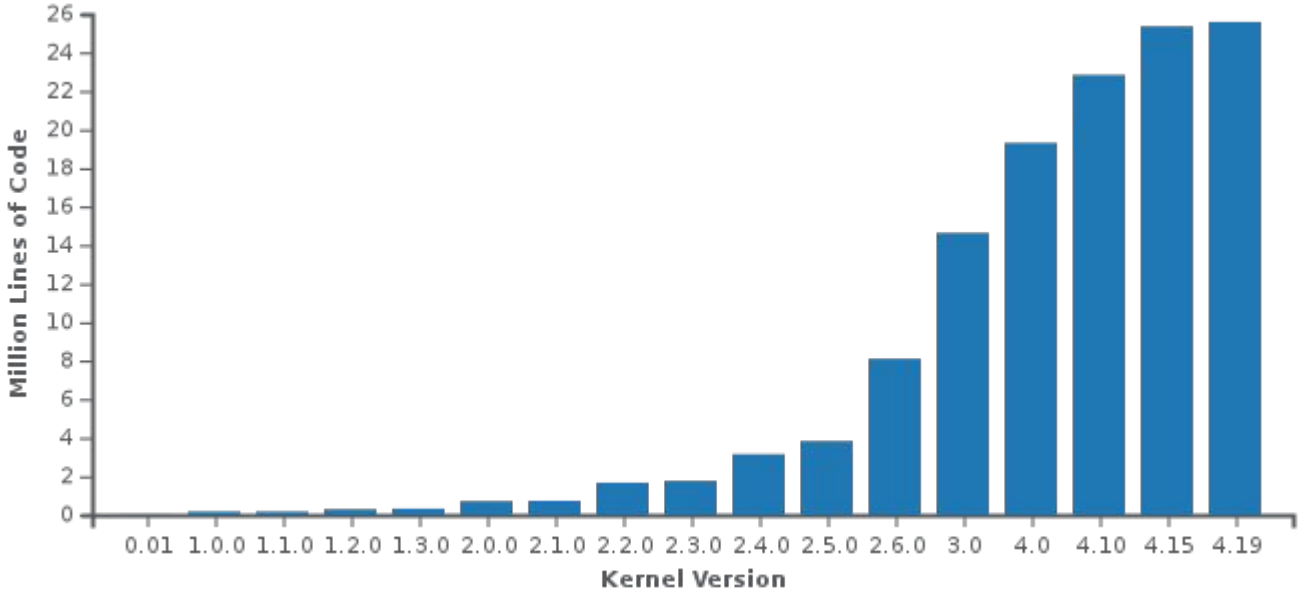
Vincent Bonnevalle, Hugo Pompougnac, Julien Rolland, Slimane Siroukane, Djamel Chekroun

21/03/2019

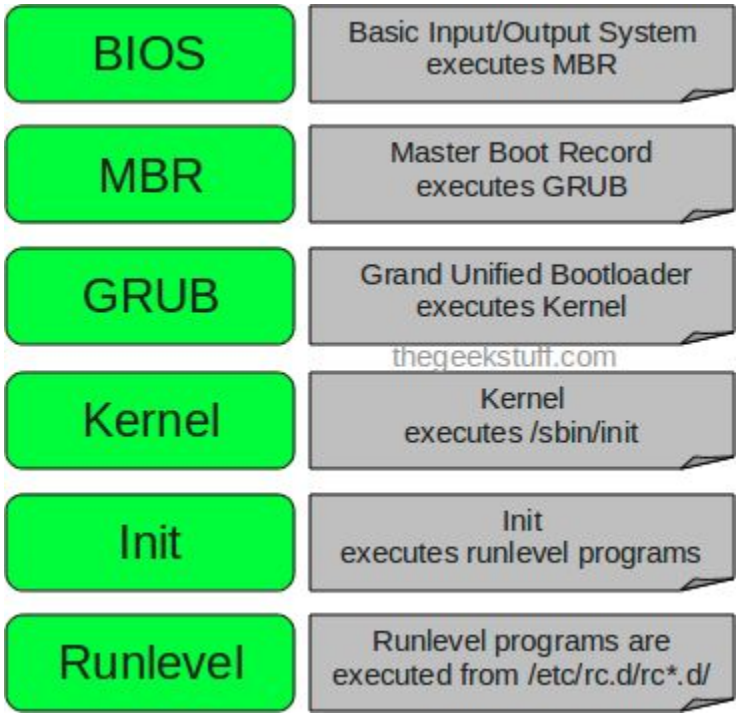
Notre sujet: le démarrage de Linux

Le noyau: monstre ou géant?

dernière version stable: 5.0.2 sur www.kernel.org



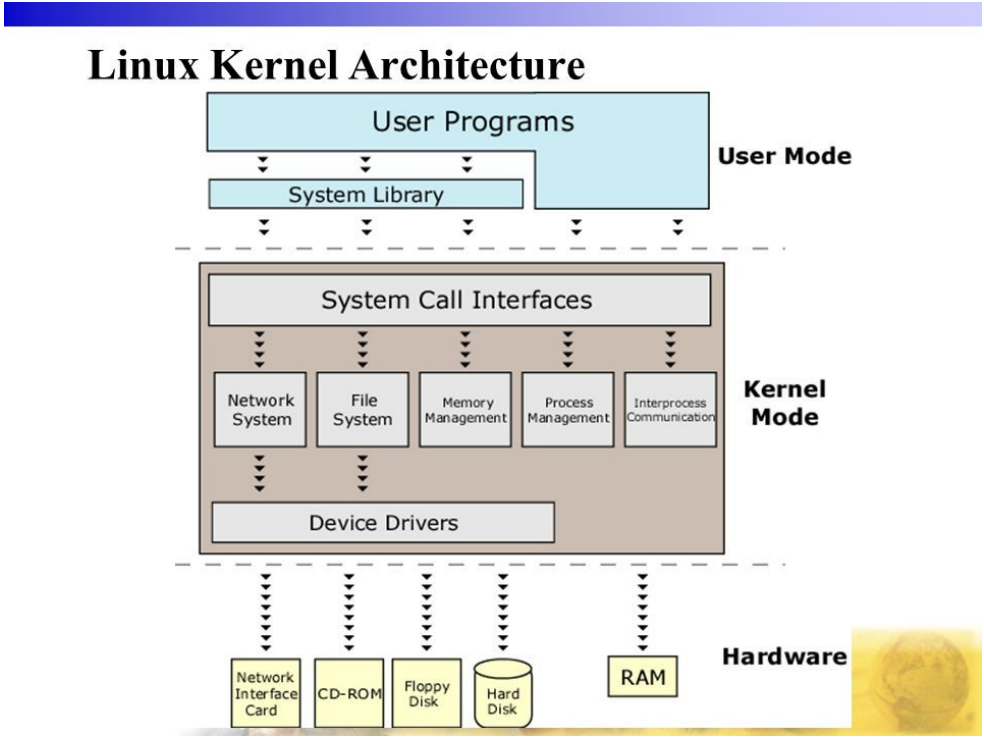
Démarrage de Linux



Les notions essentielles

- ❏ l’architecture du noyau
- ❏ l’organisation des disques
- ❏ les partitions
- ❏ les systèmes de fichiers
- ❏ ext et les inodes
- ❏ la procédure d’installation

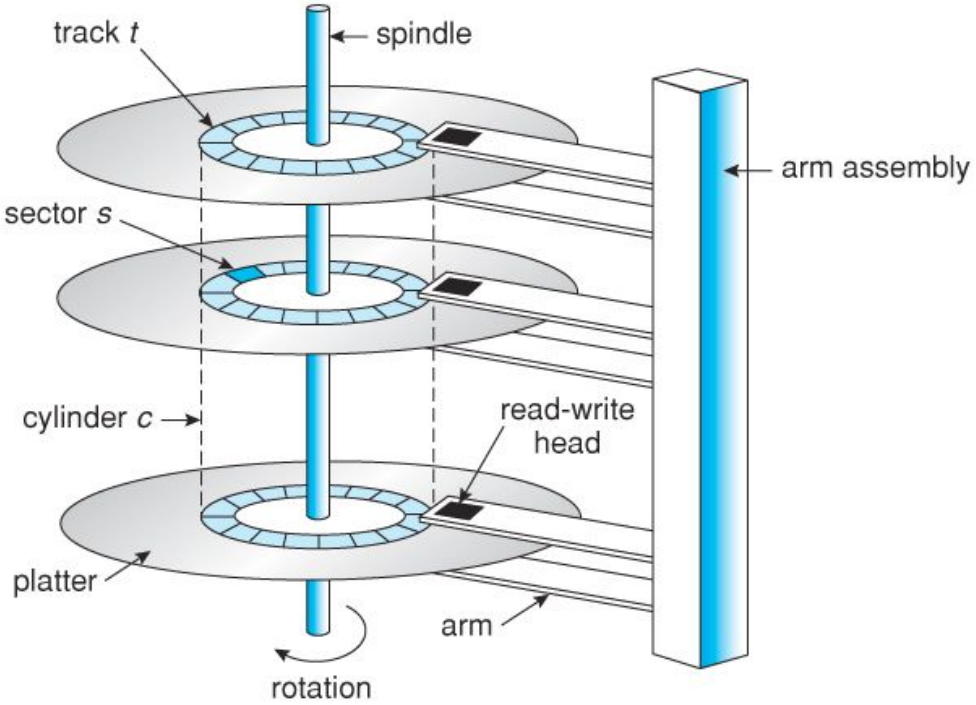
Architecture du noyau



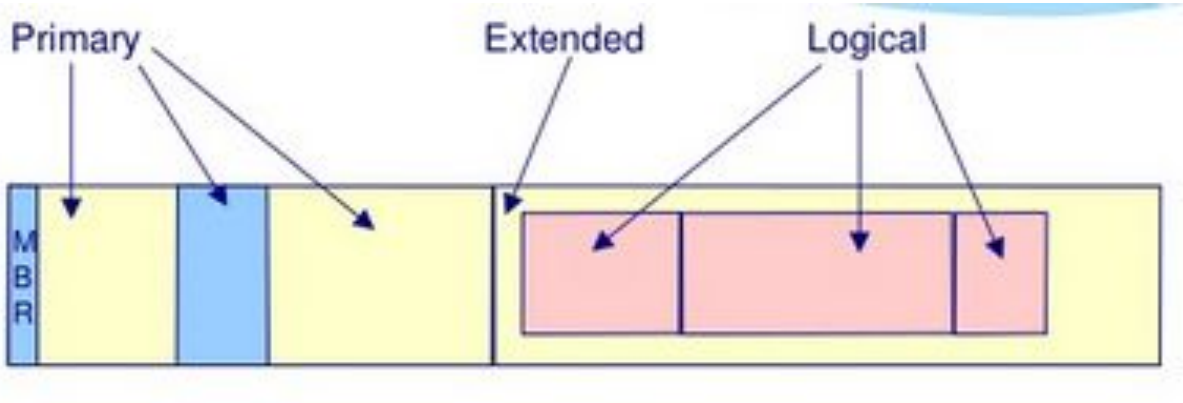
UN CONCEPT CLÉ:

L'ABSTRACTION

Disque HDD


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

Partitionnement

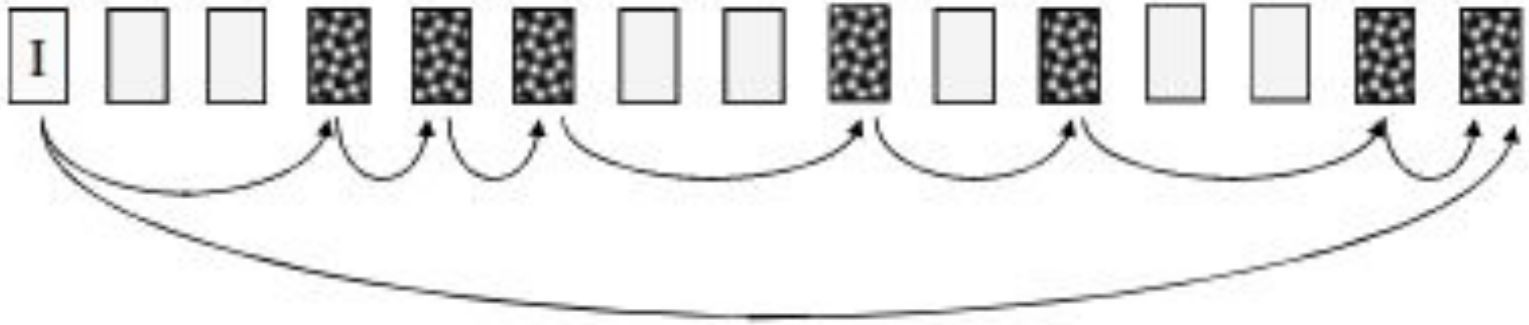


Systèmes de fichiers

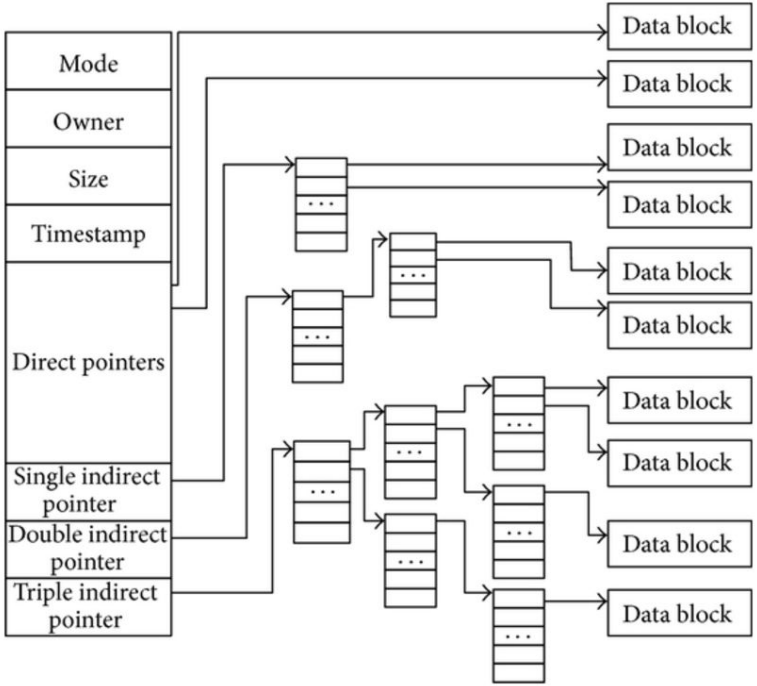
fat32, ntfs, ext2,ext3,ext4...

Deux caractéristiques essentielles:

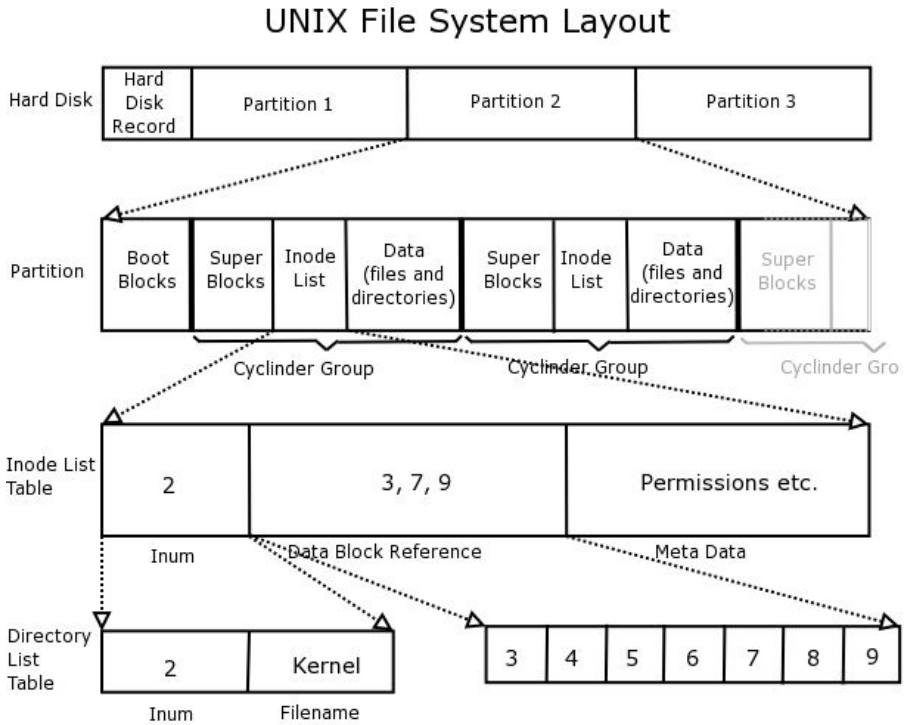
- L'organisation des catalogues
- La désignation des blocs



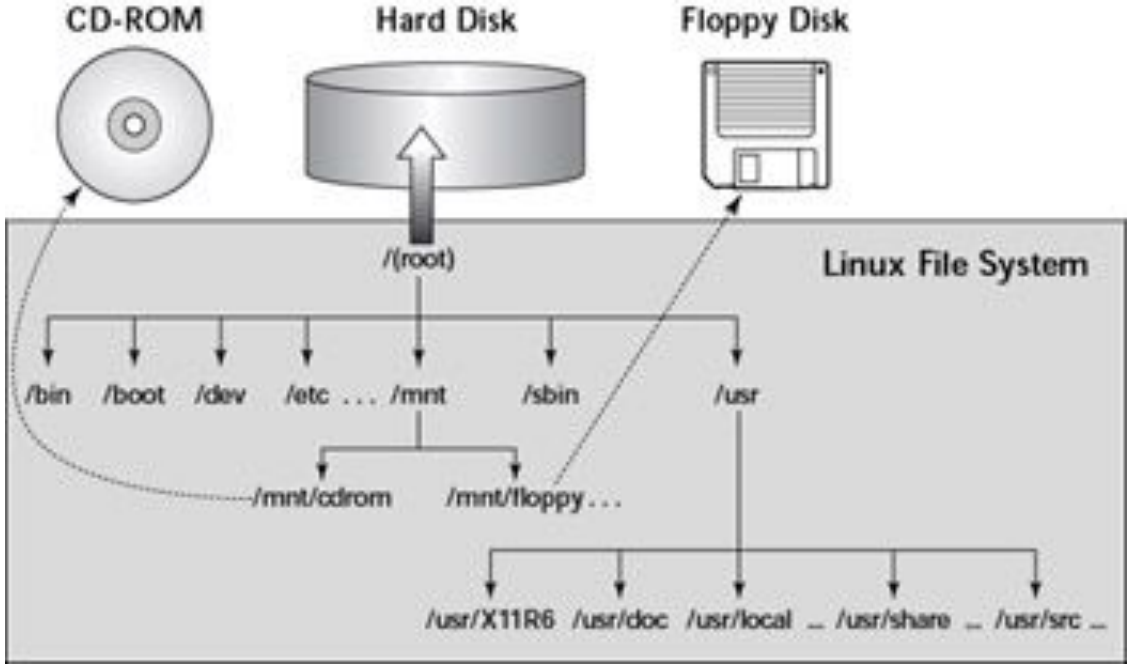
Structure d’inode



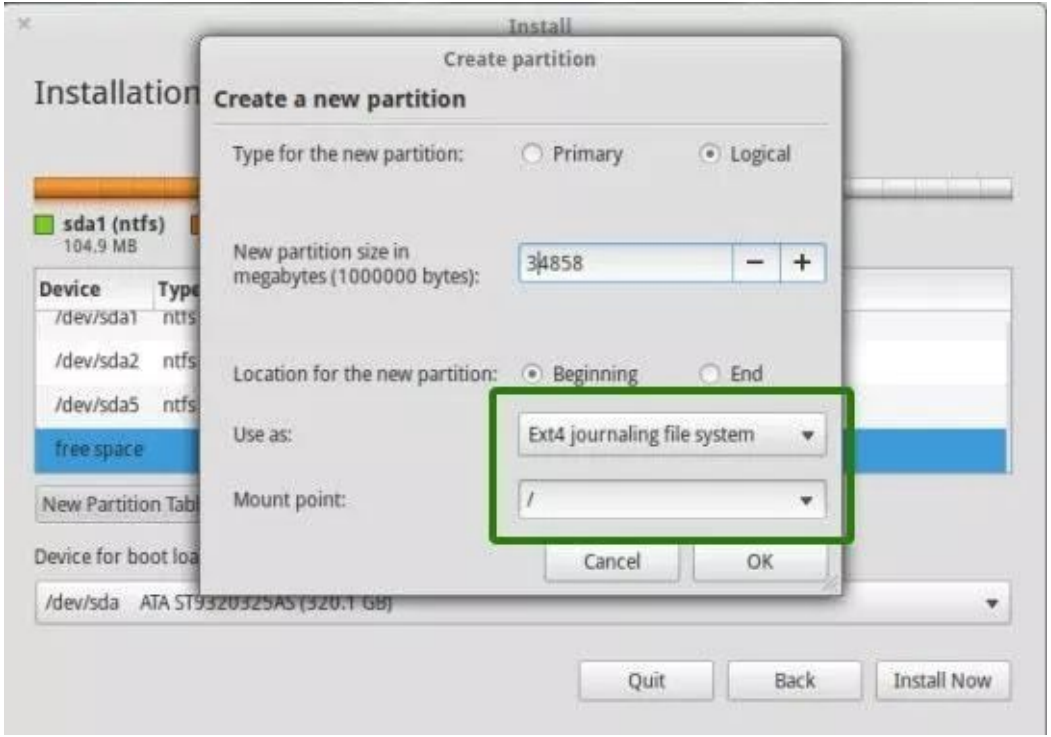
Système de fichier Linux



File system de Linux



Partitionnement à l'installation



Installation

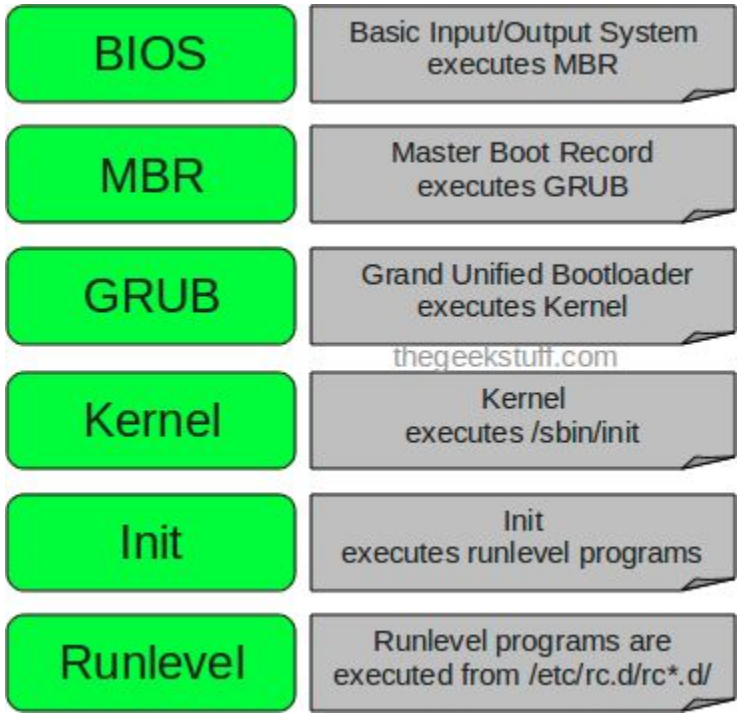
Linux démarre selon la façon dont il a été installé

- ❏ créer un média de boot
- ❏ configurer le BIOS pour le choix du média de boot (CD, USB, réseau)
- ❏ créer les partitions (pour file systems Linux, swap, muti-boot…)
- ❏ affecter les partitions aux file systems
- ❏ installation du système (et écriture du MBR et des scripts de configuration pour le démarrage)
- ❏ création d'un média de démarrage de secours

Quelques commandes

- cat /proc/version => connaître son noyau et sa distrib
- du => structure des disques
- df => examen d'un système de fichier
- fdisk => créer une partition
- outil parted/Gparted => création/modification de partitions
- mkfs => créer un système de fichier
- fsck => vérifier/réparer un système de fichier
- mount => monter un système de fichier

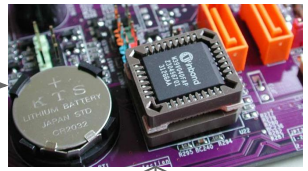
BIOS


BIOS
Basic Input/Output System
executes MBR
MBR
Master Boot Record
executes GRUB
GRUB
Grand Unified Bootloader
executes Kernel
thegeekstuff.com
Kernel
Kernel
executes /sbin/init
Init
Init
executes runlevel programs
Runlevel
Runlevel programs are
executed from /etc/rc.d/rc*.d/

Démarrage de l'ordinateur

Puce BIOS

Puce CMOS et
La pile du CMOS



Lancer l'ordinateur

Phoenix
American MegaTrends

Lecteur CD

Flash Disque

Disque Dure

Le rôle de BIOS

- **Initialisation et test des matériels (RAM, Clavier, ...)**
- **Lancement de système d’exploitation / bootloader (multi OSs)**
  - **Lancement du Boot sector**
- **Création d'une couche entre l'OS et l'E/S**

Partitions et loaders

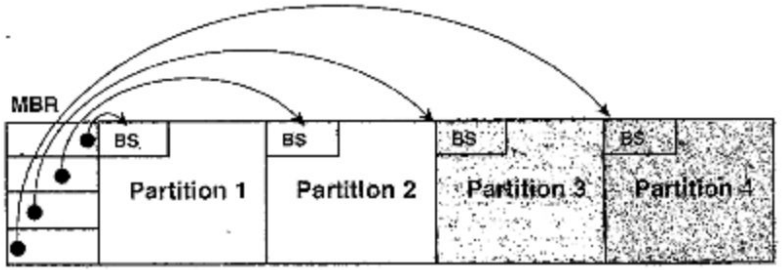


Figure 12.3 • MBR et boot sectors.

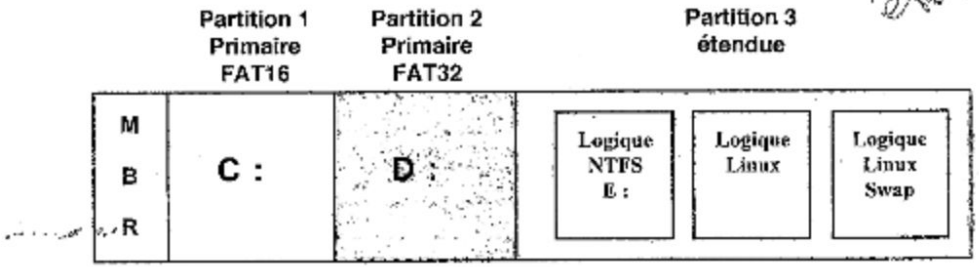
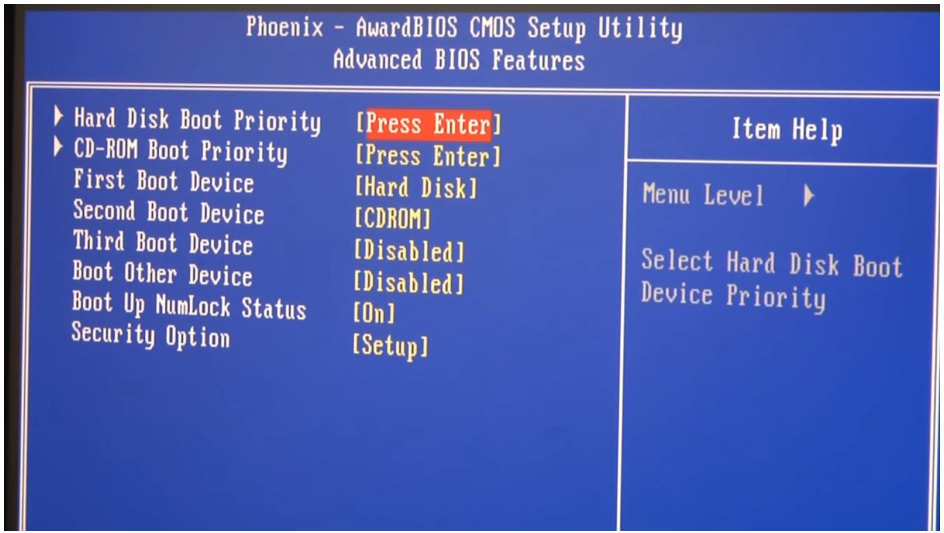


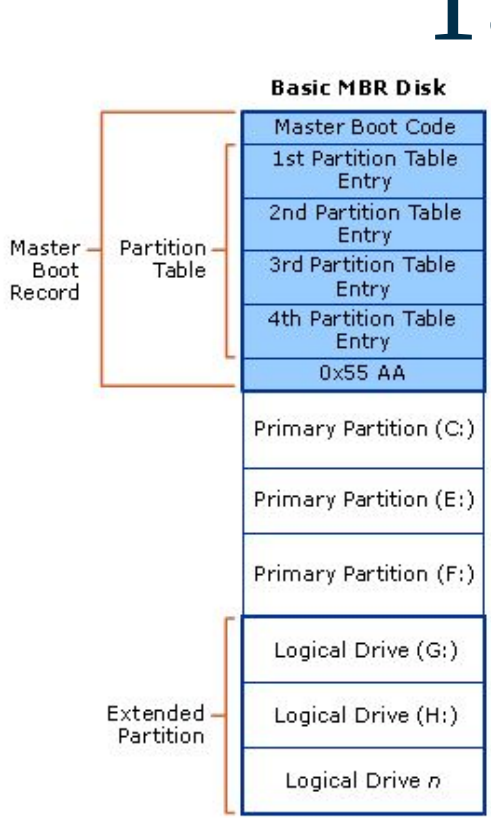
Figure 12.4 • Partitions primaires, étendues et logiques.

BIOS / UEFI


Phoenix - AwardBIOS CMOS Setup Utility
Advanced BIOS Features
Hard Disk Boot Priority [Press Enter]
CD-ROM Boot Priority [Press Enter]
First Boot Device [Hard Disk]
Second Boot Device [CDROM]
Third Boot Device [Disabled]
Boot Other Device [Disabled]
Boot Up NumLock Status [On]
Security Option [Setup]
Item Help
Menu Level
Select Hard Disk Boot Device Priority

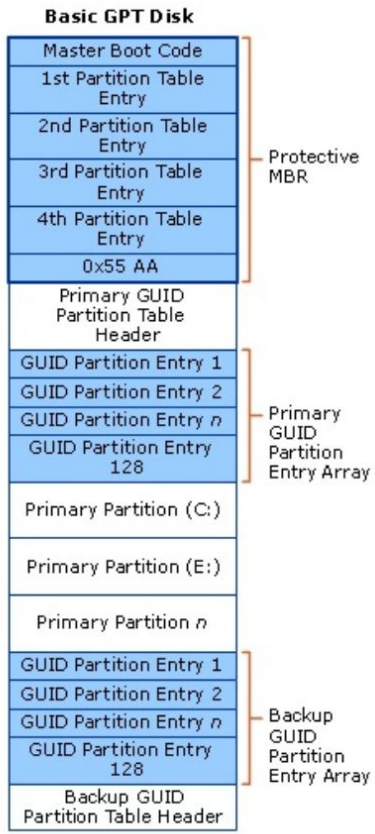

EFI BIOS Utility - EZ Mode
Exit/Advanced Mode
01:42
Saturday[4/9/2011]
SABERTOOTH P67
BIOS Version : 1305
CPU Type : Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
Total Memory : 8192 MB (DDR3 1600MHz)
English
Build Date : 02/11/2011
Speed : 3431 MHz
Temperature
CPU +118.4°F/+48.0°C
MB +98.6°F/+37.0°C
Voltage
CPU 1.232V
5V 5.000V
3.3V 3.328V
12V 12.096V
Fan Speed
CPU_FAN 1339RPM
PWR_FAN N/A
CHA_FAN1 N/A
CHA_FAN2 N/A
System Performance
Quiet
Performance
Energy Saving
ASUS Optimal
Boot Priority
UEFI
Use the mouse to drag or keyboard to navigate to decide the boot priority.
Boot Menu(F8)
Default(F5)

Table de partition



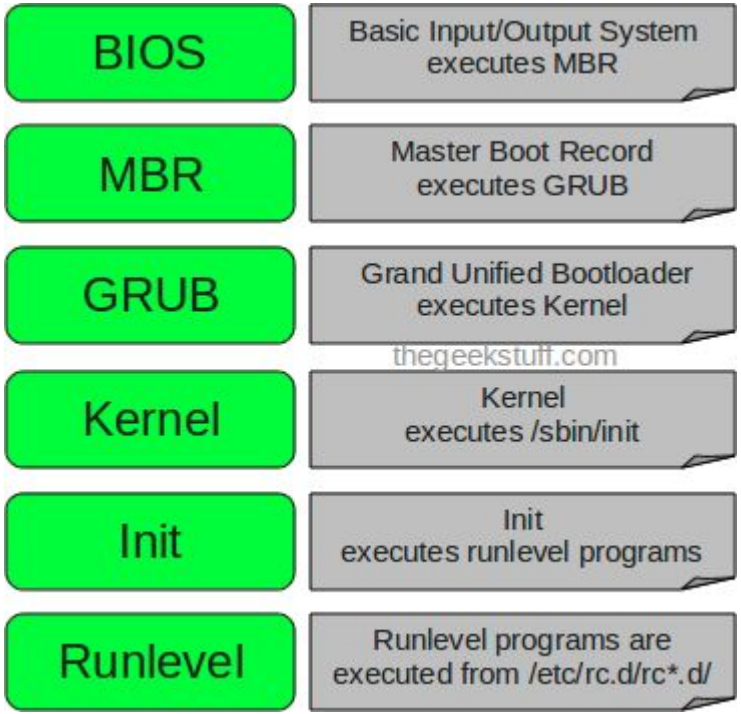
MBR

GPT



Les 512 octets (dont 442) octets du MBR

GRUB


BIOS
Basic Input/Output System
executes MBR
MBR
Master Boot Record
executes GRUB
GRUB
Grand Unified Bootloader
executes Kernel
thegeekstuff.com
Kernel
Kernel
executes /sbin/init
Init
Init
executes runlevel programs
Runlevel
Runlevel programs are
executed from /etc/rc.d/rc*.d/

GRUB

GRand Unified Bootloader :

- prépare la phase de démarrage de Linux ou, en cas de multi-boot, propose les différents OS disponibles et passe la main au bootloader du système choisi
- peut donner des paramètres entrés par l'utilisateur pour le démarrage du noyau

GRUB

Deux parties stage 1 et stage 2:

- Code de chargement dans le *MBR*
- Loader avec :
    - E/S
    - Shell
    - Modules (gestion de système de fichier, ...)
    - Code du passage de la main au noyau

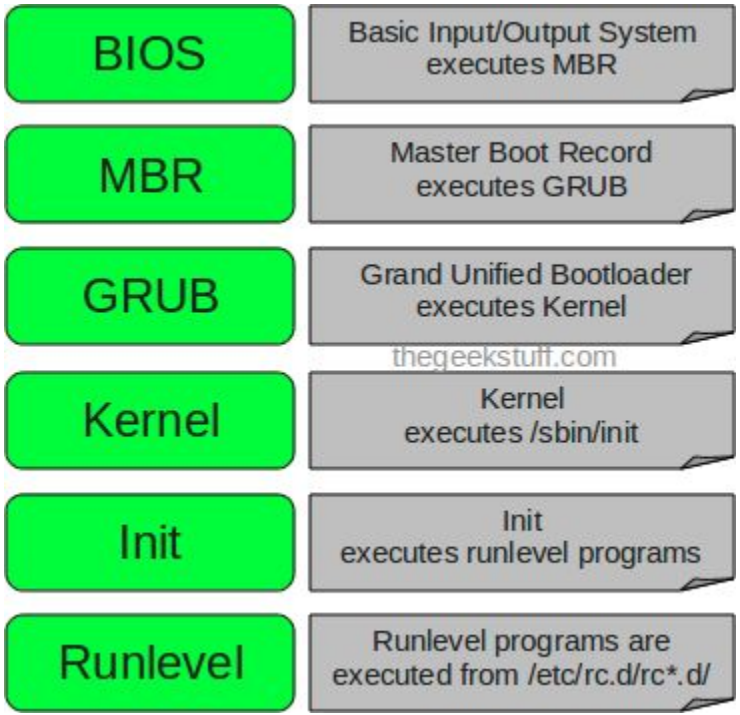
GRUB

Configuration par script bash dans /etc/grub.d/

```
-rwxr-xr-x 1 root root  8871 23 nov.  21:42 00_header
-rwxr-xr-x 1 root root 10400 23 nov.  21:42 10_linux
-rwxr-xr-x 1 root root 10455 23 nov.  21:42 20_linux_xen
-rwxr-xr-x 1 root root 11301 23 nov.  21:42 30_os-prober
-rwxr-xr-x 1 root root   214 23 nov.  21:42 40_custom
-rwxr-xr-x 1 root root   216 23 nov.  21:42 41_custom
```

mise à jour de la configuration avec **update-grub**

Set up du noyau



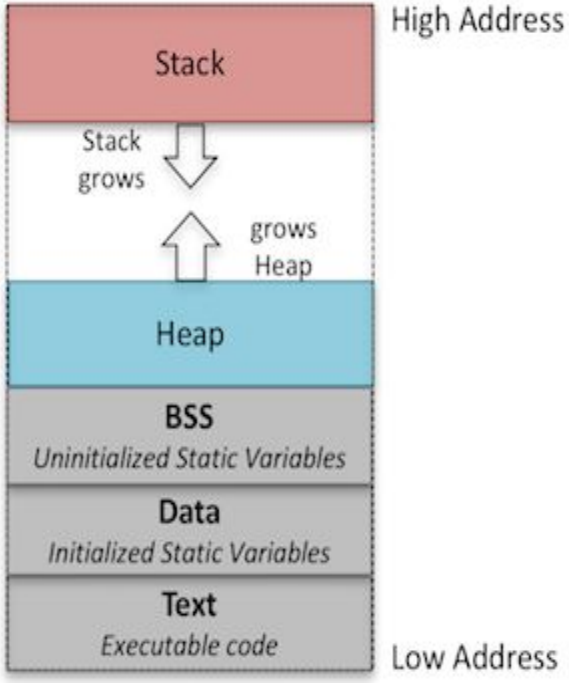
kernel ex machina

C’est le code qui “installe” le noyau à chaque démarrage : le bootloader lui donne la main, et une fois qu’elle s’est exécutée, le noyau est prêt à l’emploi.

Elle a la responsabilité :

- ❏ D’installer un environnement d’exécution opérationnel pour le C
- ❏ De donner accès au matériel indispensable
- ❏ D’installer la mémoire virtuelle et de passer en protected mode
- ❏ De décompresser l’image du noyau

parat memoriae
(préparation de la mémoire)



Fiat C et facta C

“Que le C soit, et le C fut”

La setup part commence par préparer la mémoire.

- ❏ Le bootloader a déjà chargé le setup code dans la section text, et le les fonctions dont il a besoin dans la section init
- ❏ On commence par installer la pile (càd positionner le stack pointer)
- ❏ On initialise ensuite les mots de la section bss à zéro
- ❏ On installe enfin le tas dans l’espace mémoire entre les deux

La setup part fournit et implémente un sous-ensemble de la libC.

- ❏ Les fonctions de manipulation de la mémoire et des strings
- ❏ L’accès explicite depuis le C aux registres et au tas
- ❏ L’accès aux interruptions du bios et les fonctions d’entrées-sorties

Mens sana in corpore sano

“Un esprit sain dans un corps sain”, Juvénal

Le noyau a désormais un cerveau : le C. Il lui manque un corps… c’est à dire qu’il doit accéder au matériel. Il collecte donc dans l’ordre :

- ❏ Les flags du processeur (jeu d’instructions, anneau de privilèges…)
- ❏ La mémoire physique
- ❏ Le clavier
- ❏ La carte graphique, à partir de laquelle il lance le mode d’affichage dans la meilleure résolution possible (pour fournir un feedback visuel aux opérations ultérieures)

Ad astra per aspera

“Vers les étoiles à travers les difficultés”

La phase sans laquelle rien n’est possible : passer le processeur en protected mode (on était jusqu’ici en real mode). Pourquoi faire ?

- ❏ Disposer d’un adressage sur 32/64 bits (et non 16)
- ❏ Supporter les commutations de contexte
- ❏ Supporter la segmentation de la mémoire et la mémoire virtuelle

Il faut créer/charger le GDT (Global Descriptor Table), et donc fournir autant de segment descriptors qu’on a collecté de RAM. Pour chacun d’entre eux, renseigner :

- ❏ Le début de son adresse physique
- ❏ Sa taille
- ❏ Son type (segment/data/code) et ses droits (read/write/access)

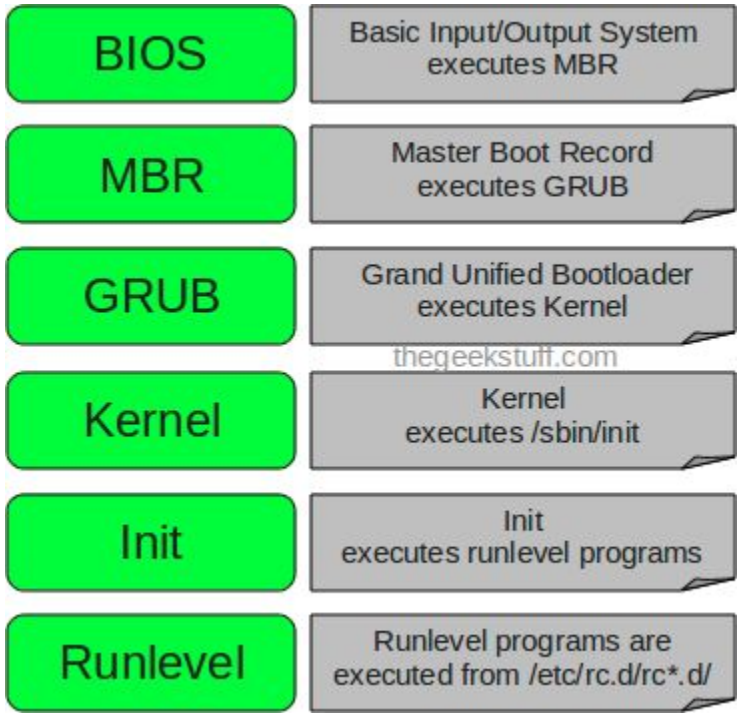
Exegi monumentum aere perennius

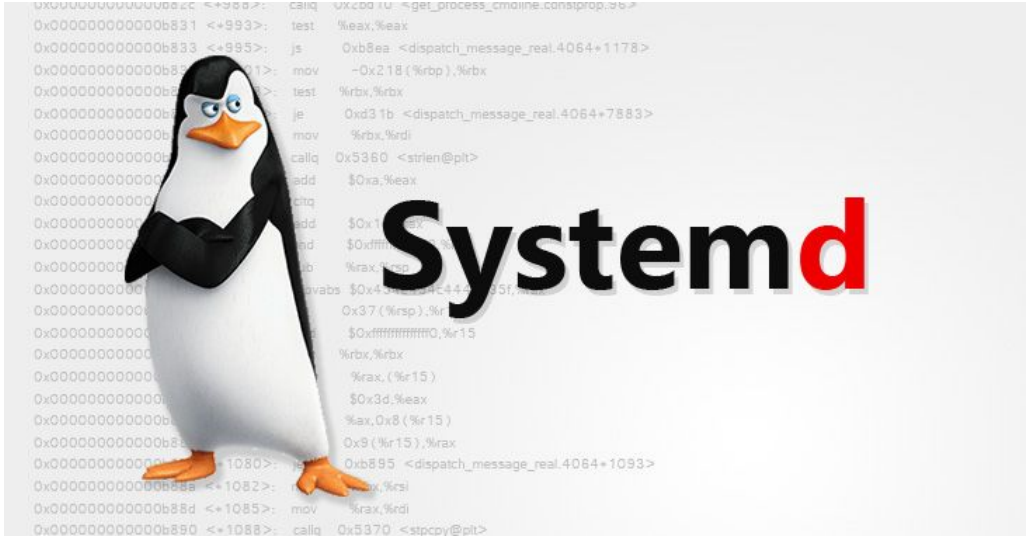
“J’ai achevé un monument plus durable que l’airain”, Horace

Il ne reste plus qu’à décompresser le noyau, qui n’est encore qu’une archive sur le disque. Avant cela, la setup part doit :

- ❏ Réorganiser la mémoire (nouvelle section text où elle charge le code de décompression, etc)
- ❏ Choisir l’adresse où le kernel sera décompressé sans qu’elle puisse être devinée à l’avance (sécurité).
- ❏ Décompresser (plusieurs algorithmes possibles : gzip, bzip2, lzma, xz, lz0, lz4)
- ❏ Déplacer le noyau à son adresse finale… Et sauter à cette adresse.

Systemd


BIOS
Basic Input/Output System
executes MBR
MBR
Master Boot Record
executes GRUB
GRUB
Grand Unified Bootloader
executes Kernel
thegeekstuff.com
Kernel
Kernel
executes /sbin/init
Init
Init
executes runlevel programs
Runlevel
Runlevel programs are
executed from /etc/rc.d/rc*.d/



**C'est le processus qui gère tous les services (on parle de processus init).
Il se charge de lancer les services (daemons) et de mettre en place un environnement pour l'utilisateur (graphique, réseau, etc) lors du démarrage du système. Sa dernière action consiste généralement à présenter un écran d'authentification, laissant la main à l'utilisateur.**

Le processus init

Il existe différents processus init, dont les plus connus sont:

- *SysVinit*: processus init historique, hérité de *UNIX System V*
- *Upstart*: sorti en 2006, à l'origine pour *Ubuntu*
- **Systemd**: sorti en 2010

***SysVinit* et *Upstart* ont une notion de niveau d'exécution (*RunLevel*) tandis que systemd a une notion de cible.**
À un niveau d'exécution ou une cible correspond un mode de fonctionnement du système dans lequel certains services sont arrêtés tandis que d'autres sont lancés.

Les unités de **systemd**

- Les unités sont les objets que **systemd** utilise pour initialiser le système et sont manipulables par la commande *systemctl*.
- D'une certaine manière, les unités peuvent être assimilées à des services ou à des tâches d'autres systèmes d'initialisation.
- Les unités permettent une plus grande souplesse dans la définiton des ressources et des services

Dépendances et ordonnancement

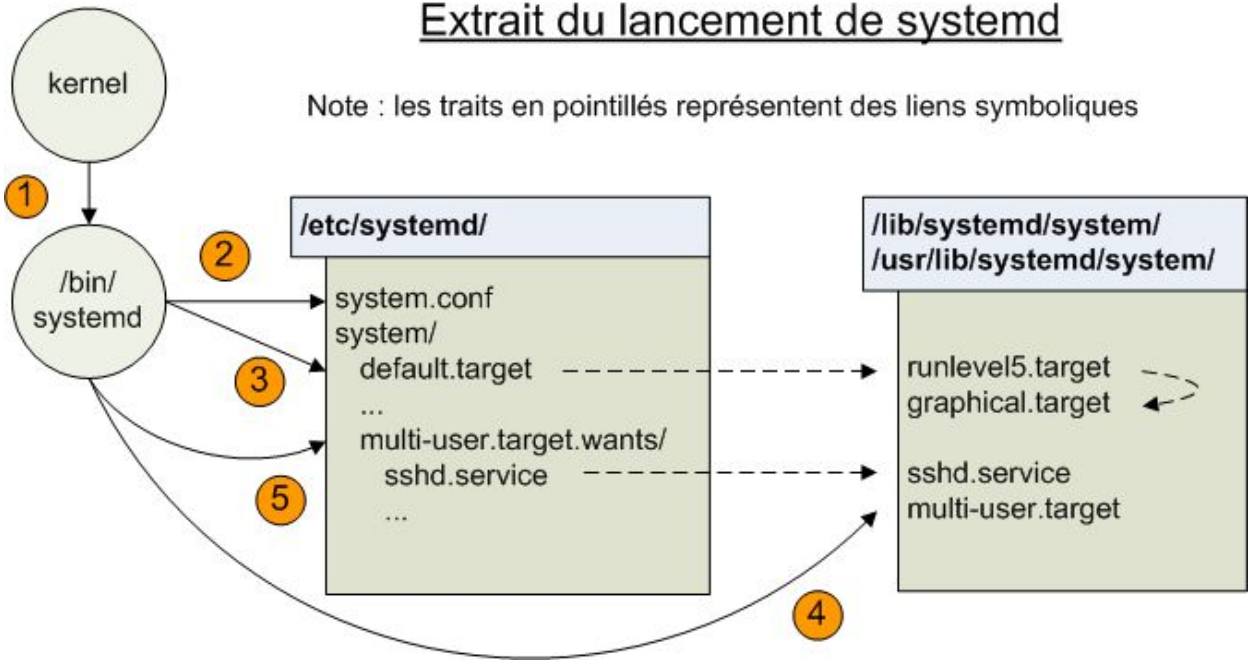
Dépendances des unités:

- Si unit1 contient `*Wants=unit2*` comme dépendance, lorsque unit1 est exécutée, unit2 sera également exécutée. Mais le bon lancement de unit2 ou son échec n'affecte pas unit1 en cours d'exécution.
- Si unit1 contient la directive `*Requires=unit2*`, les deux unités seront exécutées, mais si unit2 n'aboutit pas, unit1 est également désactivée.

Ordonnancement des unités:

- Si unit1 contient la directive `*Before=unit2*` et si les deux unités sont exécutées, unit1 sera exécutée complètement avant le démarrage de unit2.
- Si unit1 a pour directive `*After=unit2*` et si les deux unités sont exécutées, unit2 sera exécuté intégralement avant le démarrage de unit1.

Exemple d'unités dépendantes



Nos contributions au blog

- Découvrir le code source du noyau Linux
- Fiat Lux : le jour 0 du Noyau
- Configuration de GRUB
- Le stage 1 de GRUB