

Systemes Avancés - 2019

Groupe “Amorce”

Présentation initiale

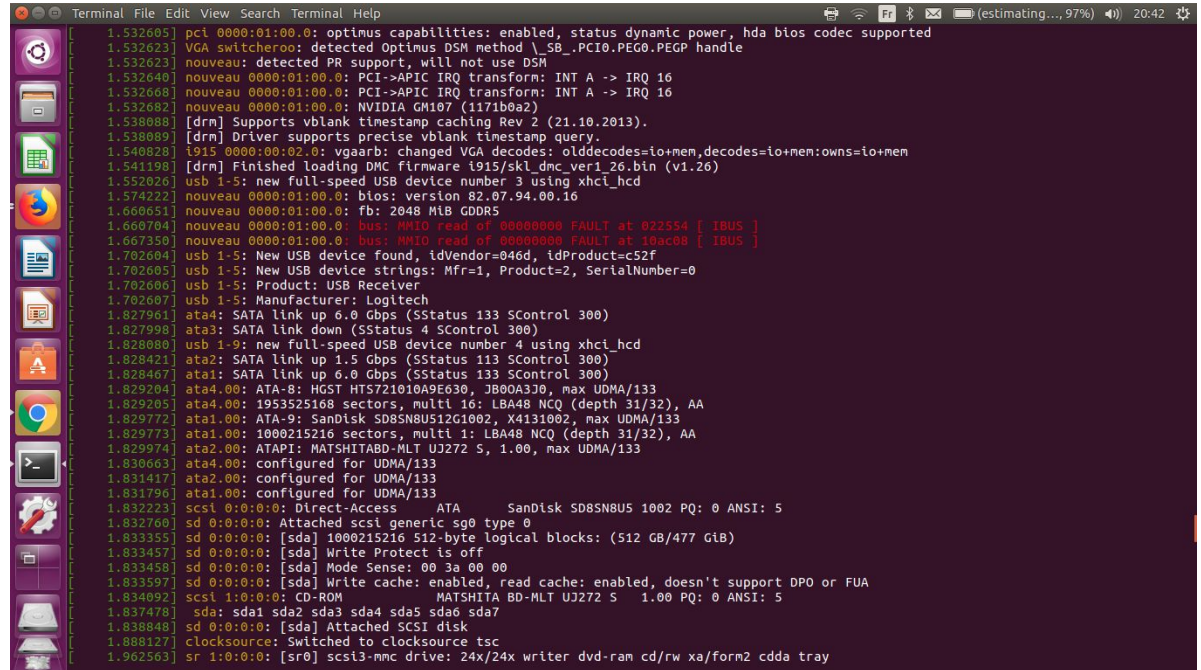
Vincent Bonneville, Hugo Pompougnac, Julien Rolland, Slimane Siroukane, Djamel Chekroun

01/02/2019

Démarrage de Linux

Que se passe-t-il après le bouton ON? Beaucoup de choses!!!

Extrait du log (commande ***dmesg***): on voit le système reconnaître la carte graphique NVIDIA et la souris LOGITECH....



```
1.532605] pci 0000:01:00.0: optimus capabilities: enabled, status dynamic power, hda bios codec supported
1.532623] vga_switcheroo: detected Optimus DSM method \_SB\_PCI0.PEG0.PEGP handle
nouveau: detected PR support, will not use DSM
1.532640] nouveau 0000:01:00.0: PCI->APIC IRQ transform: INT A -> IRQ 16
1.532668] nouveau 0000:01:00.0: PCI->APIC IRQ transform: INT A -> IRQ 16
1.532682] nouveau 0000:01:00.0: NVIDIA GM107 (1171b0a2)
1.538088] [drm] Supports vblank timestamp caching Rev 2 (21.10.2013).
1.538089] [drm] Driver supports precise vblank timestamp query.
1.540828] i915 0000:00:02.0: vgaarb: changed VGA decodes: olddecodes=io+mem,decodes=io+mem:owns=io+mem
1.541198] [drm] Finished loading DMC firmware i915/skl_dmc_ver1_26.bin (v1.26)
1.552026] usb 1-5: new full-speed USB device number 3 using xhci_hcd
1.574222] nouveau 0000:01:00.0: bios: version 82.07.94.00.16
1.660651] nouveau 0000:01:00.0: fb: 2048 MLB GDDR5
1.660704] nouveau 0000:01:00.0: bus: MMIO read of 00000000 FAULT at 022554 [ IBUS ]
1.667350] nouveau 0000:01:00.0: bus: MMIO read of 00000000 FAULT at 10ac08 [ IBUS ]
1.702604] usb 1-5: New USB device found, idVendor=046d, idProduct=c52f
1.702605] usb 1-5: New USB device strings: Mfr=1, Product=2, SerialNumber=0
1.702606] usb 1-5: Product: USB Receiver
1.702607] usb 1-5: Manufacturer: Logitech
1.827961] ata4: SATA link up 6.0 Gbps (SStatus 133 SControl 300)
1.827998] ata3: SATA link down (SStatus 4 SControl 300)
1.828080] usb 1-9: new full-speed USB device number 4 using xhci_hcd
1.828421] ata2: SATA link up 1.5 Gbps (SStatus 113 SControl 300)
1.828467] ata1: SATA link up 6.0 Gbps (SStatus 133 SControl 300)
1.829204] ata4.00: ATA-8: HGST HTS721010A9E630, JB00A3J0, max UDMA/133
1.829205] ata4.00: 1953525168 sectors, multi 16: LBA48 NCQ (depth 31/32), AA
1.829772] ata1.00: ATA-9: SanDisk SDBS8N8U512G1002, X4131002, max UDMA/133
1.829773] ata1.00: 1000215216 sectors, multi 1: LBA48 NCQ (depth 31/32), AA
1.829974] ata2.00: ATAPI: MATSHITABD-MLT UJ272 S, 1.00, max UDMA/133
1.830663] ata4.00: configured for UDMA/133
1.831417] ata2.00: configured for UDMA/133
1.831796] ata1.00: configured for UDMA/133
1.832223] scsi 0:0:0:0: Direct-Access ATA SanDisk SDBS8N8U5 1002 PQ: 0 ANSI: 5
1.832760] sd 0:0:0:0: Attached scsi generic sg0 type 0
1.833355] sd 0:0:0:0: [sda] 1000215216 512-byte logical blocks: (512 GB/477 GiB)
1.833457] sd 0:0:0:0: [sda] Write Protect is off
1.833458] sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
1.833597] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
1.834092] scsi 1:0:0:0: CD-ROM MATSHITA BD-MLT UJ272 S 1.00 PQ: 0 ANSI: 5
1.837478] sda: sda1 sda2 sda3 sda4 sda5 sda6 sda7
1.838848] sd 0:0:0:0: [sda] Attached SCSI disk
1.888127] clocksource: switched to clocksource tsc
1.962563] sr 1:0:0:0: [sr0] scsi3-mmc drive: 24x/24x writer dvd-ram cd/rw xa/form2 cdda tray
```

Les grandes étapes du démarrage

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/

La procédure d'installation

Linux démarre selon la façon dont il a été installé => il faut s'intéresser à l'installation

Procédure d'installation :

- 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 des scripts de configuration pour le démarrage)
- création d'un média de démarrage de secours

Les notions liées à l'installation

Ce qu'il faut comprendre :

- Organisation des disques physiques HDD
- disques SSD
- disques RAID
- Logical Volumes
- Systèmes de fichiers (NTFS,ext4..)
- Partitions
- File Systems (Arborescences des répertoires)
- BIOS, MBR

BIOS/MBR vs UEFI/GPT

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/

Le rôle de BIOS

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

BIOS / UEFI

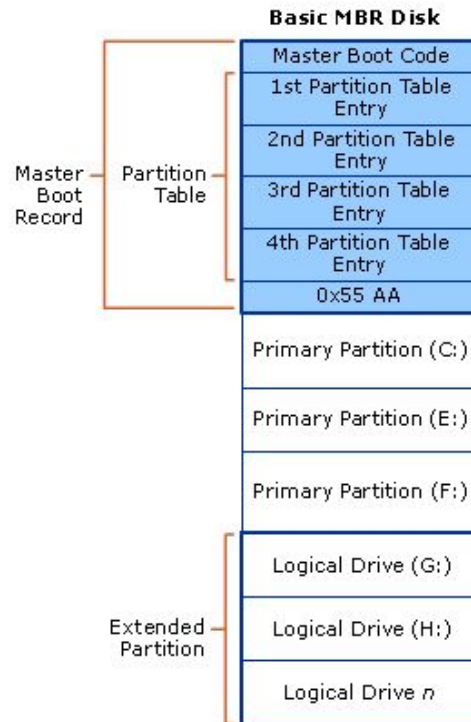
UEFI est un BIOS plus avantageux (gestion de partitions GPT, sécurité, interface graphique, ...etc), le grand changement qui nous intéresse est le mécanisme de gestion des partitions:

❑ BIOS / MBR

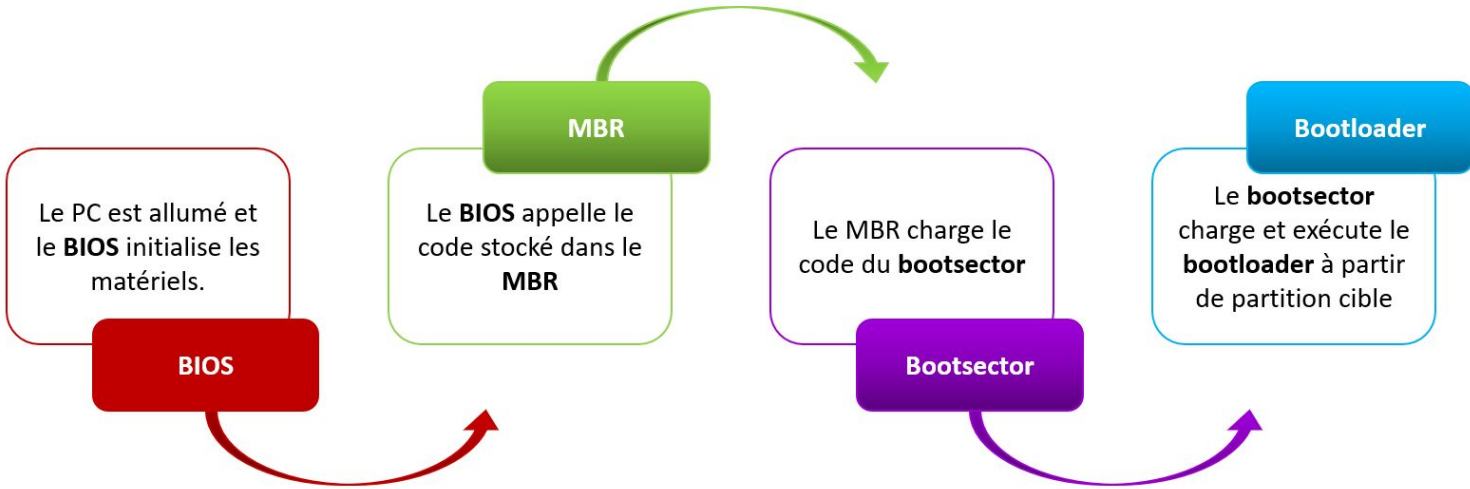
- ❑ Seulement 4 partitions (2.2 To par partition maximum)
- ❑ MBR (seulement 512 octets de long)

❑ UEFI / GPT

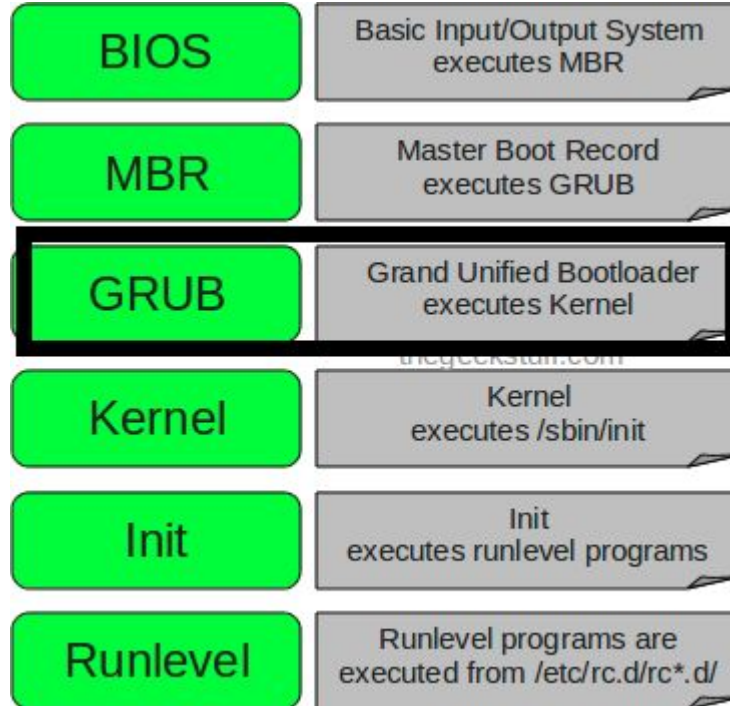
- ❑ Jusqu'à 128 partitions (256 To par partition maximum)
- ❑ Il suffit de créer une partition de type EFI (ESP).



BIOS / MBR



Le Bootloader



Le Bootloader

Fonctionnement de GRand Unified Bootloader :

- ❑ Programme en deux parties
 - ❑ 1^{er} étape : programme dans le *MBR*, charge 2^{ème} étape
 - ❑ 2^{ème} étape : kernel de *GRUB*
- ❑ Configuré par des fichiers créés lors de l'installation de *GRUB*
- ❑ Possède un mini-kernel pour les *I/O*
- ❑ Passe la main à l'étape suivante en fonction des entrées utilisateur

Le Bootloader

Ce qu'il faut comprendre :

- ❑ Comment le bootloader est lancé
- ❑ Comment le bootloader lance l'étape suivante du boot
- ❑ Comment le bootloader détecte les systèmes d'exploitations
- ❑ Comment le bootloader fait les *I/O* en l'absence d'OS

La setup part du noyau Linux

Lorsqu'un bootloader donne la main à Linux, la setup part du noyau doit d'abord préparer le système

Préparation du système :

- ❑ Mise en place d'un environnement d'exécution pour du C primitif (pile, tas, e/s minimales, segments de la mémoire, mémoire virtuelle)
- ❑ Détection du matériel indispensable (barrettes de RAM, clavier, carte vidéo) et des fonctions correspondantes (affichage...)
- ❑ Transition du processeur du *real-mode* vers le *protected-mode* (où le noyau segmente la mémoire), et éventuellement vers le *64-bits mode*
- ❑ Décompression du noyau
- ❑ Exécution du noyau lui-même

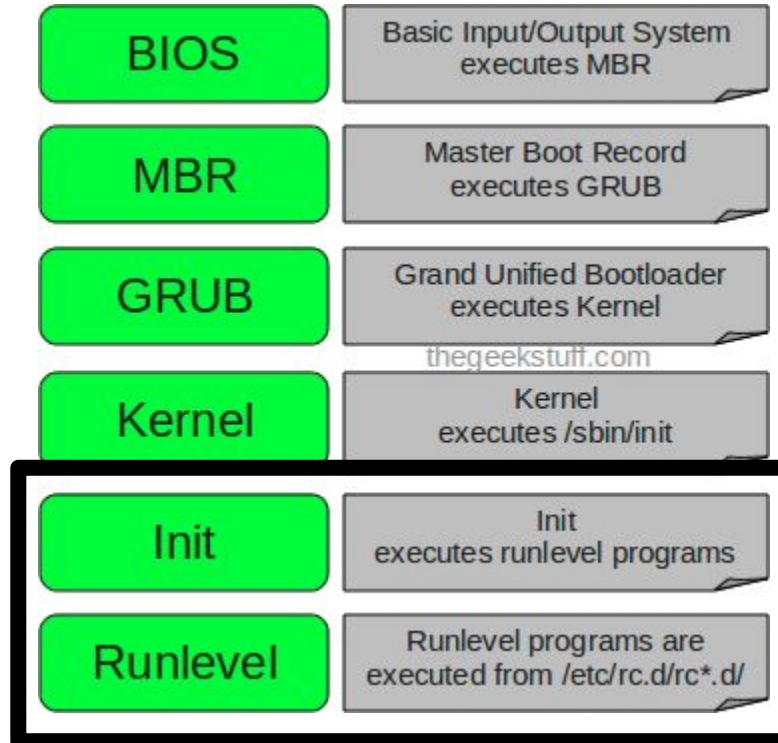
La *setup part* du noyau Linux

La *setup part* est mystérieuse : c'est l'endroit où, en quelque sorte, le noyau se génère lui-même, le big bang du système. Comment est-ce possible ?

Ce qu'il faut comprendre :

- ❑ Comment crée-t-on le C... Alors qu'on ne peut pas utiliser le C ?
- ❑ Quel sous-ensemble du C est ensuite utilisable dans la *setup part* ?
- ❑ Comment offre-t-elle les fonctions matérielles élémentaires (affichage...) alors que les drivers sont dans le noyau compressé ?
- ❑ De quoi a-t-on besoin pour mettre le processeur en *protected-mode* ?
- ❑ Comment la *setup part* protège-t-elle le noyau d'une prise de contrôle malveillante durant sa décompression ?

Processus d'initialisation du système



Processus d'initialisation du système

Une fois le noyau Linux opérationnel, celui-ci lance la phase d'initialisation des différents services indispensables.

Init SystemV :

- ❑ Le noyau exécute sbin/init qui devient le processus racine du système avec pour PID 1
- ❑ Ce programme lit les informations concernant les différentes partitions en fonction du "run level" par défaut (contrôle le choix des services démarrés automatiquement par le système)
- ❑ Lancement séquentiel de ces services, localisés dans le répertoire /etc/init.d, en tâche de fond
- ❑ Le programme init gère ces services jusqu'à l'arrêt du système

Processus d'initialisation du système

Systemd, un gestionnaire d'initialisation plus perfectionné, a pris le pas sur SystemV dans les machines modernes mais est d'autant plus complexe.

Ce qu'il faut comprendre :

❑ **Systemd**

- ❑ Quelques scripts de service communs aux deux gestionnaires

Conclusion

Méthode de travail (et d'exposition) pour les prochaines échéances de travail.

- ❑ Approfondir la compréhension fonctionnelle de chacune de ces phases (qu'est-ce que ça fait, quelle interface ça expose)
- ❑ Exhiber des fragments de code du noyau sur lesquels ces fonctionnements reposent (et les expliquer)
- ❑ Relier ce que voit l'utilisateur (les sorties sur l'écran ou dans les fichiers de log) avec ce que fait le système en arrière-plan