Systèmes Avancés - 2019
Groupe “Amorce”
Présentation initiale

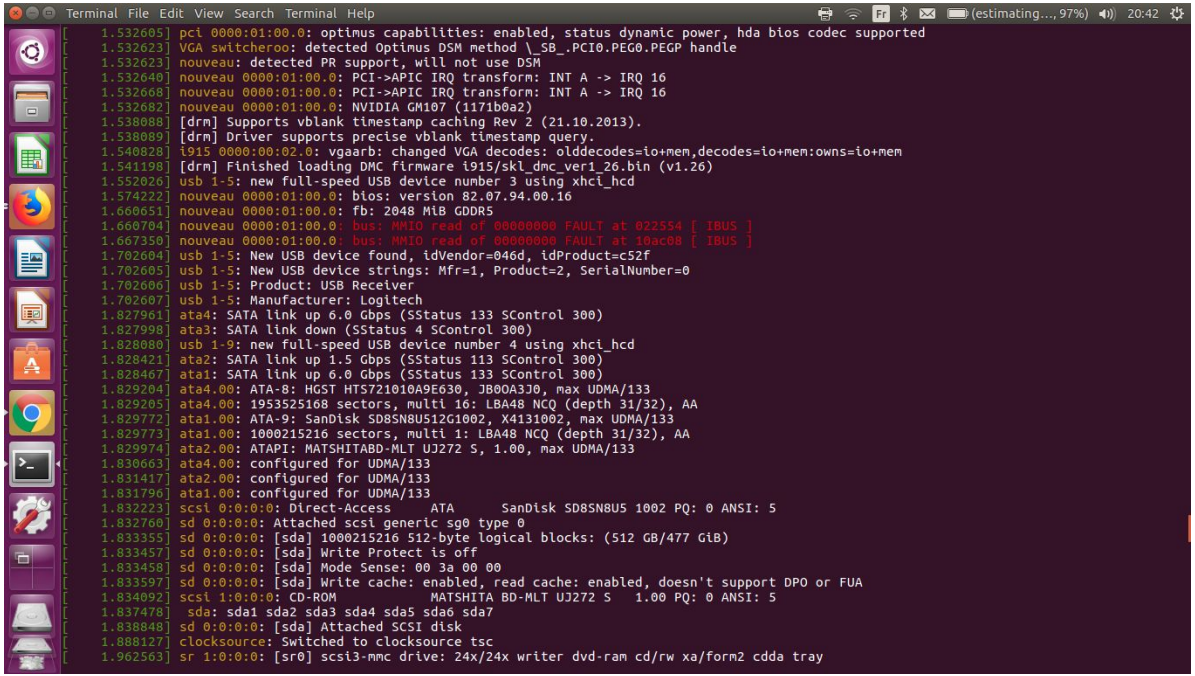
Vincent Bonnevalle, Hugo Pompougnac, Julien Rolland, Slimane Siroukane, Djamel Chekroun

01/02/2019

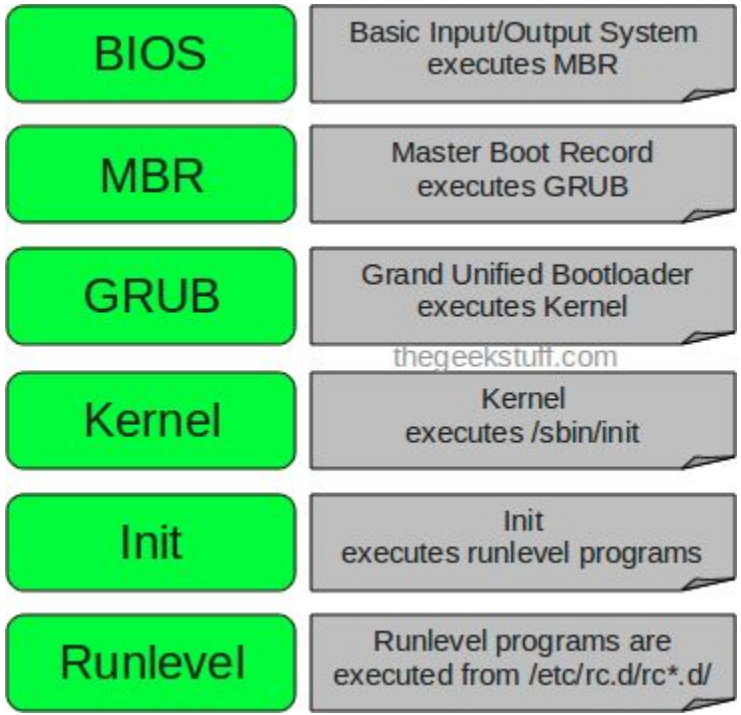
Démarrage de Linux

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

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



Les grandes étapes du démarrage



La procédure d'installation

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

Procédure d'installation :

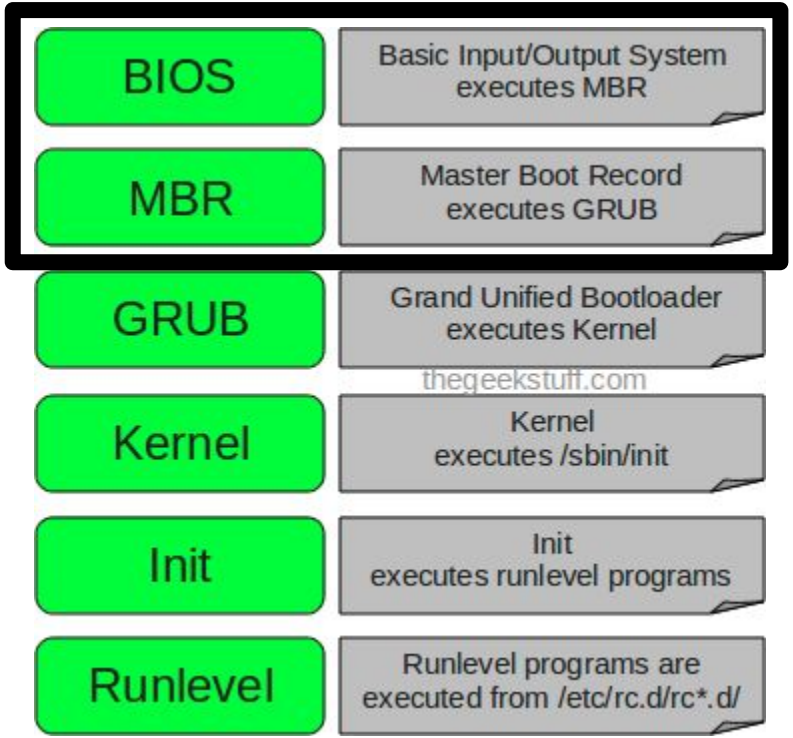
- ❏ créer un média de boot
- ❏ configurer le BIOS pour le choix du média de boot (CD, USB, réseau)
- ❏ créer les partitions (pour file systems Linux, swap, muti-boot…)
- ❏ affecter les partitions aux file systems
- ❏ installation du système (et écriture des scripts de configuration pour le démarrage)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



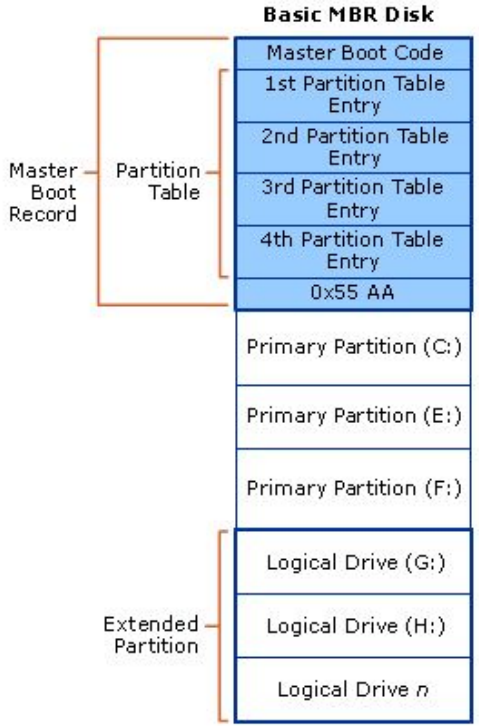
Le rôle de BIOS

- **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**

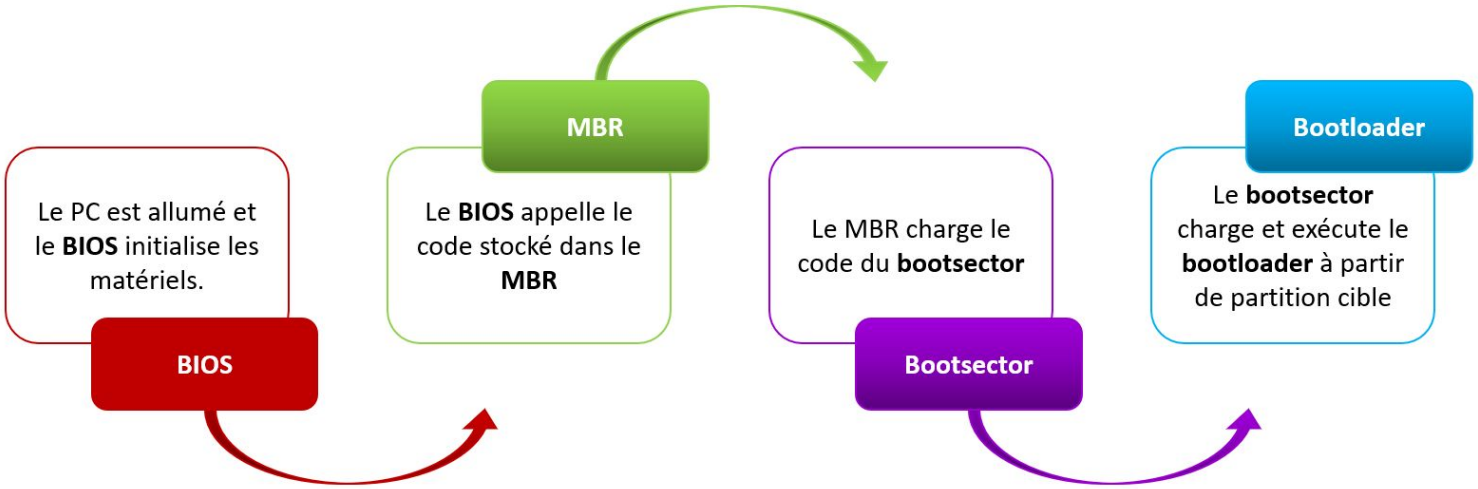
BIOS / UEFI

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:

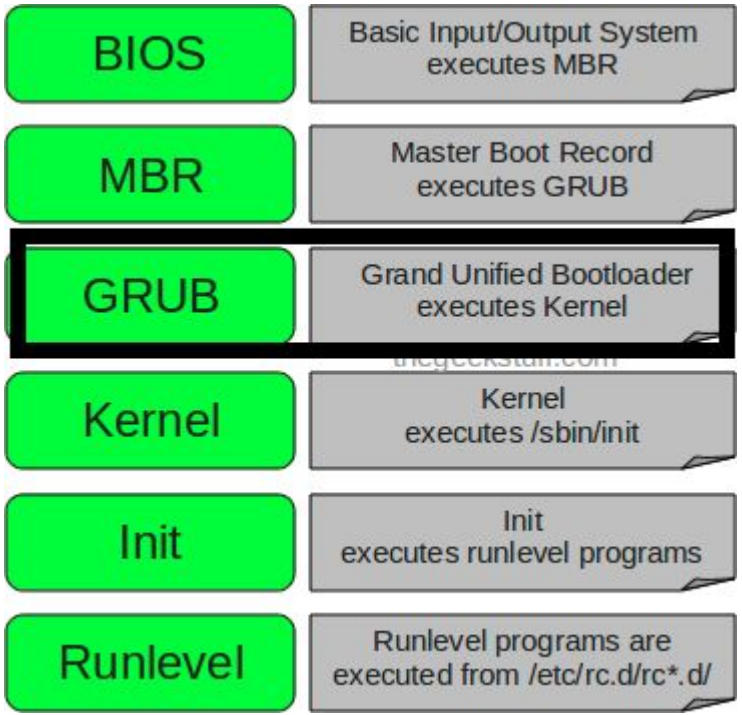
- **BIOS / MBR**
  - Seulement 4 partitions (2.2 To par partition maximum)
  - MBR (seulement 512 octets de long)
- **UEFI / GPT**
  - Jusqu'à 128 partitions (256 To par partition maximum)
  - Il suffit de créer une partition de type EFI (ESP).



BIOS / MBR


Le PC est allumé et le **BIOS** initialise les matériels.
BIOS
MBR
Le **BIOS** appelle le code stocké dans le **MBR**
Le MBR charge le code du **bootsector**
Bootsector
Bootloader
Le **bootsector** charge et exécute le **bootloader** à partir de partition cible

Le *Bootloader*



Le *Bootloader*

Fonctionnement de GRand Unified Bootloader :

- Programme en deux parties
  - 1er étape : programme dans le *MBR*, charge 2ème étape
  - 2ème 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*

<u>Ce qu'il faut comprendre</u> :

- ❏ 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

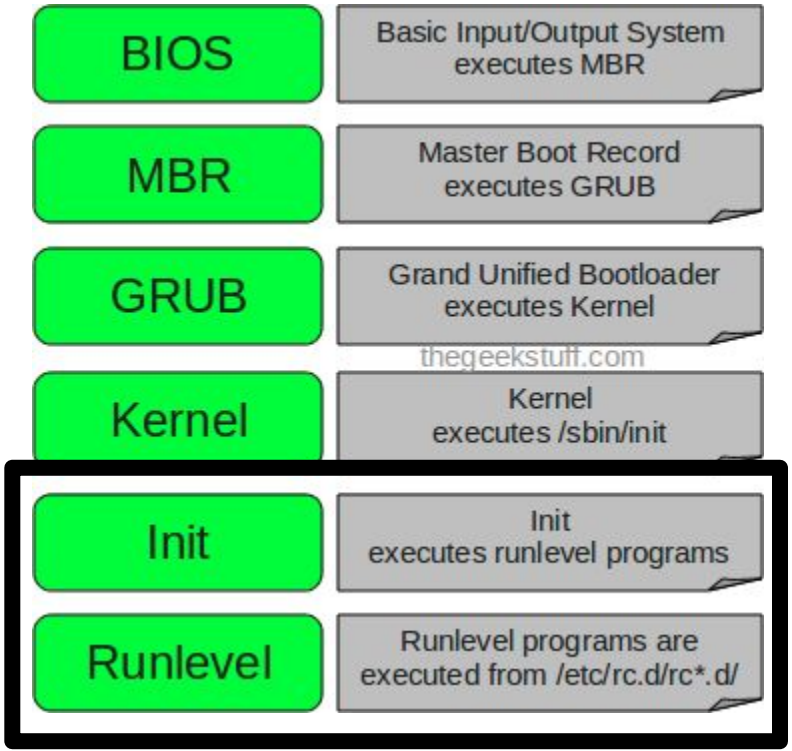
La *setup part* du noyau Linux

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 ?

<u>Ce qu'il faut comprendre</u> :

- ❏ 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.

<u>Ce qu'il faut comprendre</u> :

- **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