Introduction, Objectifs et Plan

À chaque fois que vous appuyez sur le bouton d'alimentation de votre ordinateur, vous lancez une série d'étapes complexes. Ce geste, qui semble simple, fait passer votre PC d'un état éteint à une machine prête à fonctionner. Dans ce chapitre, nous allons examiner en détail ces étapes pour mieux comprendre comment votre ordinateur démarre.

Objectifs

Voici les objectifs que nous visons à atteindre dans ce chapitre :

1. Voici les objectifs que nous visons à atteindre dans ce chapitre :Maîtriser la séquence de démarrage des systèmes d'exploitation Linux.
2. Configurer le gestionnaire de démarrage.
3. Comprendre en détail le processus d’initialisation.
4. Apprendre à personnaliser le processus d’initialisation selon les besoins spécifiques.

Plan

Ainsi, nous allons structurer ce chapitre en deux parties distinctes :

- Partie 1 : Séquence de démarrage
- Partie 2 : Processus d’initialisation

Introduction

Qu'est-ce que le démarrage ?

Le démarrage d'un ordinateur est le processus par lequel il passe de l'état éteint à un état pleinement opérationnel, prêt à exécuter des applications et à fournir des services, dès que l'utilisateur appuie sur le bouton d'alimentation. Ce processus implique plusieurs étapes cruciales, chacune ayant un rôle spécifique pour préparer le système à fonctionner correctement. Ces étapes comprennent:

- L'initialisation du matériel,
- La localisation et le lancement du chargeur de démarrage,
- L'amorçage du noyau du système d'exploitation,
- L'initialisation des services et des processus nécessaires

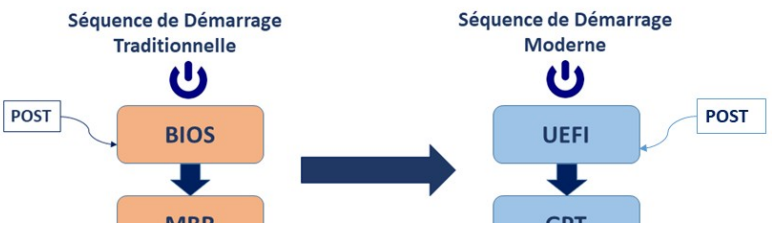
Chaque phase est essentielle pour garantir que l'ordinateur est entièrement prêt à être utilisé.

Vue d'ensemble des étapes du démarrage

Le processus de démarrage d'un ordinateur sous Linux commence lorsque l'utilisateur appuie sur le bouton d'alimentation. L'UEFI (Unified Extensible Firmware Interface), remplaçant moderne du BIOS, initialise le matériel et configure les composants essentiels. Ensuite, le système utilise la table de partition GUID (GPT), qui a remplacé le MBR (Master Boot Record), pour gérer les partitions des disques durs et localiser la partition contenant le chargeur de démarrage.

Le chargeur de démarrage GRUB (Grand Unified Bootloader) est ensuite chargé, permettant la sélection et le chargement du noyau Linux. Une fois le noyau chargé, il prend en charge la gestion des ressources matérielles et prépare le système pour l'exécution de logiciels supplémentaires.

À ce stade, le noyau passe le contrôle à un programme d'initialisation, tel que systemd (ou, dans les systèmes plus anciens, init), qui démarre et gère les services et processus nécessaires pour rendre l'ordinateur entièrement opérationnel. Dans les cours suivants, nous détaillerons chaque étape de la séquence de démarrage et analyserons en profondeur chaque étape du processus.

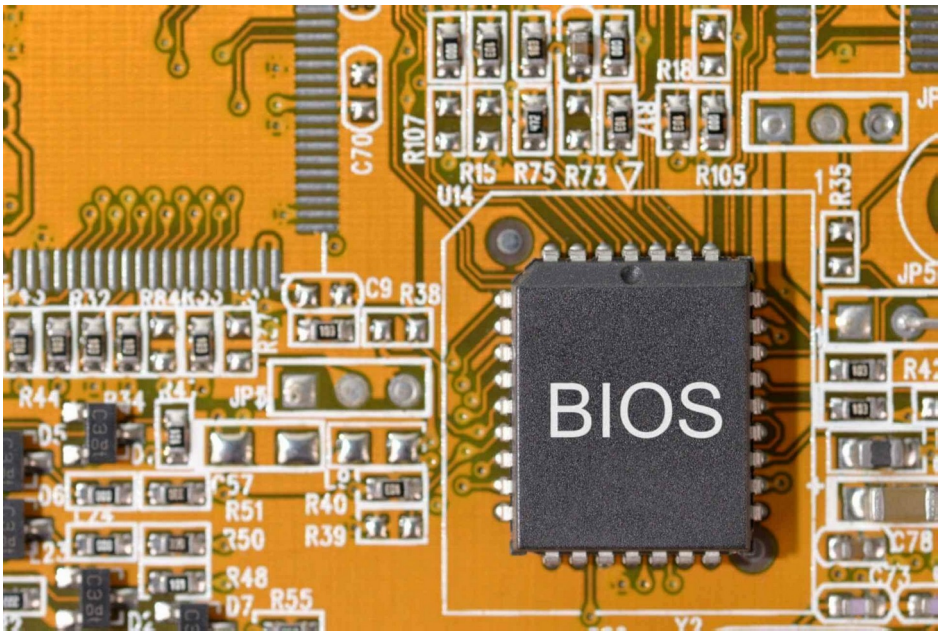



MBR
GRUB2
KERNEL
INIT
GPT
GRUB2
KERNEL
SYSTEMD

Première Etape: BIOS / UEFI

La première étape cruciale de la séquence de démarrage d'un PC commence avec le BIOS, un logiciel intégré essentiel chargé d'initialiser le matériel. Nous introduirons ensuite son successeur, l'UEFI, une amélioration significative du BIOS qui offre des fonctionnalités étendues. En explorant ces deux systèmes, nous examinerons également les différences clés qui les distinguent.

1. Qu’est-ce qu’un BIOS ?



Basic Input-Output system (https://www.howtogeek.com/179789/htg-explains-what-is-bios-and-when-should-i-use-it/)(BIOS); Il s’agit d’un logiciel de bas niveau qui réside dans une puce sur la carte mère de votre ordinateur. Le BIOS se charge au démarrage de votre ordinateur, et le BIOS est chargé de réveiller les composants matériels de votre ordinateur et de s’assurer qu’ils fonctionnent correctement.

Vous pouvez configurer différents paramètres dans l’écran de configuration du BIOS. Des paramètres tels que la configuration matérielle de votre ordinateur ainsi que l’heure système. Vous pouvez accéder à cet écran en appuyant sur une touche spécifique (différente sur différents ordinateurs, mais souvent Échap, F2, F10 ou Supprimer) pendant le démarrage de l’ordinateur.

Power On Self Test(POST): Le BIOS passe par un auto-test matériel nommé POST, ce test vérifie la présence et le bon fonctionnement des périphériques matériels tels que la mémoire RAM, le processeur et les périphériques connectés. En cas de problème matériel détecté, le POST signale généralement les erreurs par des bips sonores ou des messages à l'écran.

CMOS: Le BIOS fonctionne de paire avec une puce CMOS (Complementary Metal Oxide Semiconductor) dans laquelle les paramètres personnalisés du BIOS sont enregistrés : date et heure du système, ordre d’amorçage, configuration des disques, overclocking du CPU et la RAM, mot de passe d’accès à l’ordinateur… Une pile 3 V est nécessaire pour que les informations stockées dans la puce CMOS soient conservées lorsque l’ordinateur est éteint

ou débranché de sa source d'alimentation. Si la pile CMOS de votre ordinateur est hors service, tous les paramètres du BIOS seront réinitialisés à leurs valeurs par défaut.

2. Pourquoi le BIOS est-il obsolète ?

Le BIOS traditionnel présente de sérieuses limitations. Il ne peut démarrer qu'à partir de disques de 2,1 To ou moins. Les disques de 8 To sont désormais courants, et un ordinateur doté d'un BIOS ne peut pas démarrer à partir de ceux-ci. Cette limitation est due au fonctionnement du système Master Boot Record du BIOS.

Le BIOS doit fonctionner en mode processeur 16 bits et ne dispose que de 1 Mo d'espace pour s'exécuter. Il a du mal à initialiser plusieurs périphériques matériels à la fois, ce qui entraîne un processus de démarrage plus lent lors de l'initialisation de toutes les interfaces et périphériques matériels sur un PC moderne.

Le BIOS a besoin d'être remplacé depuis longtemps. Intel a commencé à travailler sur la spécification EFI (Extensible Firmware Interface) en 1998. Apple a choisi EFI lorsqu'il est passé à l'architecture Intel sur ses Mac en 2006, mais les autres fabricants de PC n'ont pas suivi.

En 2007, Intel, AMD, Microsoft et les fabricants de PC se sont mis d'accord sur une nouvelle spécification UEFI (Unified Extensible Firmware Interface). Il s'agit d'une norme à l'échelle de l'industrie gérée par , et qui n'est pas uniquement pilotée par Intel. La prise en charge UEFI a été introduite dans Windows avec Windows Vista Service Pack 1 et Windows 7. La grande majorité des ordinateurs que vous pouvez acheter aujourd'hui utilisent désormais UEFI plutôt qu'un BIOS traditionnel.

3. Comment UEFI remplace et améliore le BIOS?

3. Comment UEFI remplace et améliore le BIOS?

L'UEFI (Unified Extensible Firmware Interface) est un micrologiciel stocké dans une mémoire flash située sur la carte mère de l'ordinateur. Installé lors de la fabrication, il constitue le premier programme à s'exécuter au démarrage de l'ordinateur. À l'instar du BIOS traditionnel, l'UEFI joue un rôle crucial en effectuant un test POST (Power-On Self-Test) pour vérifier la présence et le bon fonctionnement des périphériques matériels. Ce test garantit que les composants nécessaires au démarrage sont correctement identifiés et opérationnels avant de charger le système d'exploitation. L'UEFI assure ainsi une transition fluide et fiable vers le chargement du système d'exploitation, tout en offrant des fonctionnalités avancées par rapport au BIOS classique.

a. Les avantages offerts par UEFI:

UEFI offre plusieurs avantages tels que:

- Le mécanisme de rétro-compatibilité: UEFI a remplacé le BIOS traditionnel sur les PC. Il n'est pas possible de mettre à jour un BIOS existant vers UEFI, il est nécessaire d'acheter un nouveau matériel compatible avec UEFI, comme ceux des ordinateurs récents. Toutefois, la plupart des systèmes UEFI offrent une émulation du BIOS, permettant ainsi l'installation et le démarrage de systèmes d'exploitation plus anciens conçus pour le BIOS, assurant ainsi une rétrocompatibilité.

- La prise en charge des disques supérieurs: Cette nouvelle norme permet de surmonter les limitations du BIOS. Le micrologiciel UEFI peut démarrer à partir de disques de 2,2 To ou plus, jusqu'à 9,4 zettaoctets. En effet, UEFI utilise le schéma de partitionnement GPT au lieu de MBR (nous examinerons en détail MBR et GPT dans le reste du cours).

- Un démarrage plus rapide: UEFI peut s'exécuter en mode 32 bits ou 64 bits et dispose d'un espace d'adressage plus important que le BIOS, ce qui signifie que votre processus de démarrage est plus rapide. Cela signifie également que les écrans de configuration UEFI peuvent être plus lisses que les écrans de paramètres du BIOS, y compris les graphiques et la prise en

écrans de paramètres du BIOS, y compris les graphiques et la prise en charge du curseur de la souris.

- La prise en charge du Secure boot: UEFI regorge d'autres fonctionnalités. Il prend en charge le démarrage sécurisé (Secure Boot), une fonctionnalité de sécurité qui vérifie la signature numérique des logiciels et des systèmes d'exploitation pendant le processus de démarrage. Lorsqu'elle est activée, cette vérification garantit que seul un système d'exploitation et des logiciels approuvés peuvent s'exécuter au démarrage. Cela empêche les logiciels malveillants de s'introduire avant que le système d'exploitation ne commence à fonctionner, car ces logiciels ne passeraient pas la vérification de signature.

- La prise en charge des fonctionnalités réseaux: UEFI peut prendre en charge les fonctionnalités de mise en réseau directement dans le micrologiciel UEFI lui-même, ce qui peut faciliter le dépannage et la configuration à distance. En revanche, avec un BIOS traditionnel, toutes les configurations doivent être effectuées directement sur l'ordinateur. Il n'est pas possible de gérer le BIOS à distance, ce qui nécessite une présence physique pour effectuer des ajustements ou des réparations.

b. La mémoire NVRAM

UEFI utilise la NVRAM (Non-Volatile Random-Access Memory), une mémoire vive non volatile, pour stocker les variables essentielles au démarrage de l'ordinateur, qui doivent persister entre les redémarrages. Contrairement à la RAM volatile, qui perd ses données lorsque l'ordinateur est éteint, la NVRAM conserve les données de configuration, les entrées de démarrage, et d'autres informations cruciales même lorsque l'alimentation est coupée. Cette capacité permet à UEFI d'assurer une gestion rapide et fiable des paramètres système, offrant ainsi une flexibilité et une stabilité accrues par rapport au BIOS traditionnel, qui utilisait la mémoire CMOS.

Parmi les informations stockées dans la NVRAM figurent des éléments tels que les options de démarrage suivantes:

.....✔ Quel système est actuellement démarré ?

..... ✔ L'ordre des entrées de démarrage

.....✔ Le délai avant de démarrer sur l'entrée par défaut (suivant l'ordre précédent)

..... ✔ La liste des entrées de démarrage

Pour gérer les options de démarrage, UEFI utilise des variables spécifiques appelées EFI_LOAD_OPTION. Ces variables sont composées d'un GUID (Globally Unique Identifier) qui identifie le propriétaire de la variable, d'un nom qui désigne l'option de démarrage, et d'une valeur qui contient les détails nécessaires au processus de démarrage. Vous trouverez ci-

valeur qui contient les détails nécessaires au processus de démarrage. Vous trouverez ci-dessous quelques exemples de variables stockées dans la NVRAM:

Nom de la variable	Description générale
BootOrder	Cette variable détermine l'ordre dans lequel les options de démarrage sont considérées. Le système tentera de démarrer depuis chaque entrée Boot#### dans l'ordre dans lequel elles sont listées dans cette variable. Une fois qu'une option de démarrage est chargée avec succès, le système cesse d'essayer les autres options listées.
Boot####	Cette variable représente une entrée de démarrage spécifique qui peut être amorcée. #### est un nombre hexadécimal à quatre chiffres qui identifie de manière unique cette option de démarrage.
BootCurrent	Indique l'option de démarrage Boot#### qui a été sélectionnée pour le démarrage en cours.
Timeout	Cela indique la durée en seconde après laquelle une entrée est automatiquement sélectionnée dans le menu de démarrage.
BootNext	Définir l'option de démarrage Boot#### à essayer en priorité lors du prochain démarrage. Une fois que l'option de démarrage spécifiée par BootNext a été essayée, le système reprend l'utilisation de la liste normale définie par BootOrder.

==>En raison de la diversité des fabricants et des spécificités des configurations, il est impossible de dresser une liste exhaustive des GUID utilisés pour les variables NVRAM, chaque modèle de matériel ou version de firmware pouvant introduire ses propres GUID pour les variables de configuration, de démarrage ou de sécurité. Toutefois, un exemple de GUID typique pour la variable de configuration BootOrder est {8BE4DF61-93CA-11D2-AA0D-00E098032B8C}.

c. EFIBootmgr

efibootmgr est un utilitaire en ligne de commande utilisé sous Linux pour gérer le gestionnaire de démarrage UEFI . Il permet de :

1. Répertorier les entrées du gestionnaire de démarrage EFI

1. Répertorier les entrées du gestionnaire de démarrage EFI

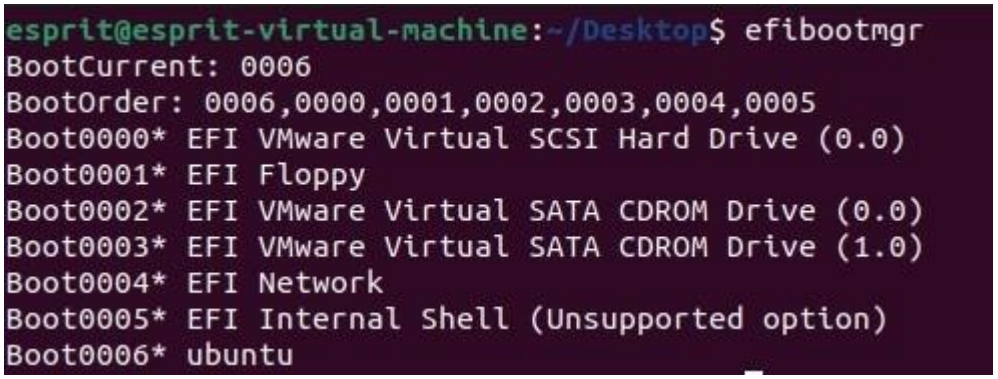
   - La commande efibootmgr, comme le montre la capture ci-dessous, affiche les entrées de démarrage disponibles, leur ordre de priorité et l'entrée actuellement utilisée pour le démarrage.

BootCurrent: 0006 : Indique que l'entrée actuellement utilisée pour démarrer est 0006, correspondant à Ubuntu.

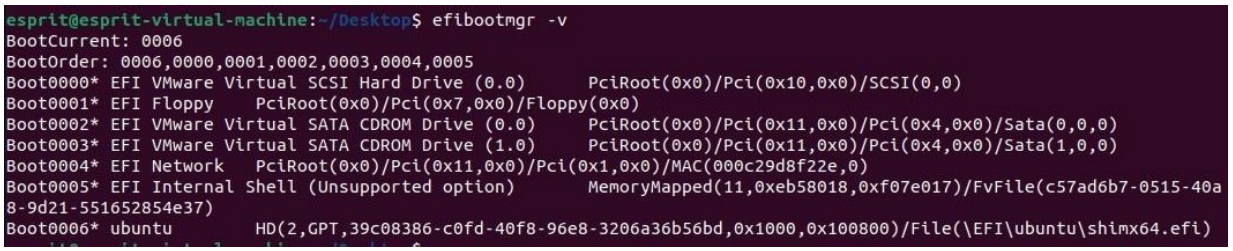
BootOrder: 0006,0000,0001,0002,0003,0004,0005 : Spécifie l'ordre dans lequel les périphériques de démarrage sont essayés,L'ordre de démarrage est :

1. Boot0006* ubuntu
2. Boot0000* EFI VMware Virtual SCSI Hard Drive (0.0): Disque dur virtuel SCSI dans une machine virtuelle VMware.
3. Boot0001* EFI Floppy: Lecteur de disquettes EFI.
4. Boot0002* EFI VMware Virtual SATA CDROM Drive (0.0): Lecteur de CD-ROM SATA virtuel dans une machine virtuelle VMware.
5. Boot0003* EFI VMware Virtual SATA CDROM Drive (1.0): Autre lecteur de CD-ROM SATA virtuel dans une machine virtuelle VMware.
6. Boot0004* EFI Network: Option de démarrage réseau.
7. Boot0005* EFI Internal Shell (Unsupported option): Shell EFI interne, souvent utilisé pour le dépannage ou le diagnostic.

Le système tentera d'abord de démarrer à partir de l'entrée 0006 (Ubuntu), puis passera aux autres périphériques selon l'ordre spécifié si l'entrée actuelle échoue ou n'est pas disponible.

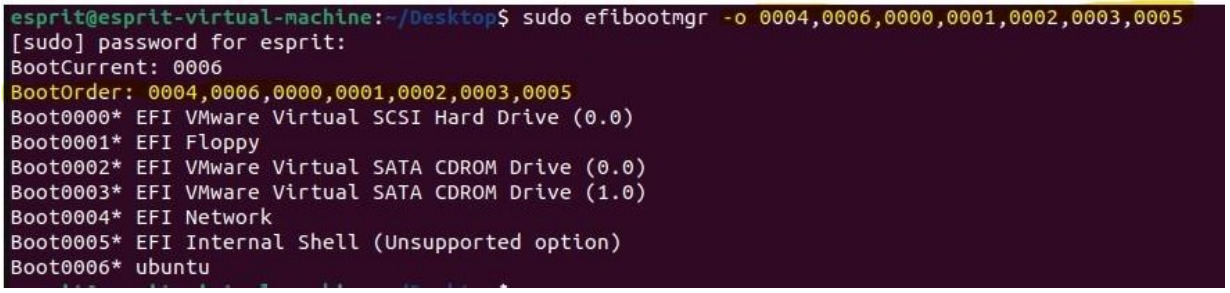


   - Pour lister les entrées de démarrage avec les chemins, utilisez l'option -v :



2. Changer l'ordre des entrées

La commande efibootmgr -o permet de modifier l'ordre de démarrage des périphériques. Comme illustré dans la capture d'écran ci-dessous., la commande sudo efibootmgr 0004,0006,0000,0001,0002,0003,0005 ajuste l'ordre de démarrage dans le firmware UEFI. Après l'exécution de cette commande, le système mettra à jour l'ordre de démarrage en essayant d'abord le périphérique 0004 (démarrage réseau), suivi de 0006 (Ubuntu), et ainsi de suite selon l'ordre spécifié.



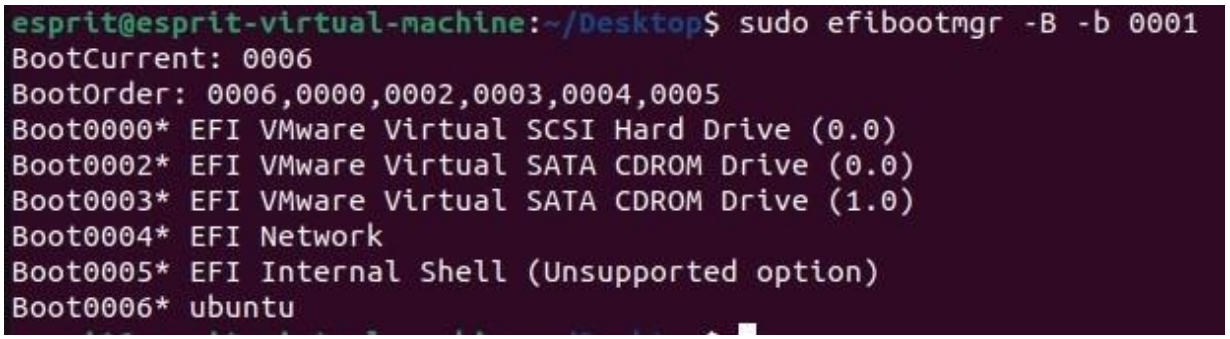
3. Supprimer une entrée de démarrage

La commande sudo efibootmgr -B -b <numéro_entrée> est utilisée pour supprimer une entrée de démarrage.

Comme illustré dans la capture ci-dessous, si l'on souhaite supprimer l'entrée "EFI Floppy" (Boot0001), on utilise la commande suivante : sudo efibootmgr -B -b 0001

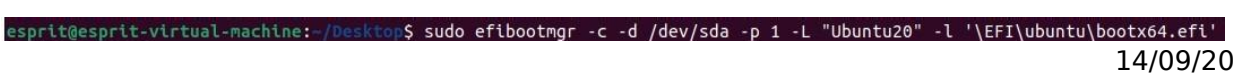
- -b 0001 : spécifie l'entrée de démarrage à supprimer.
- -B : Pour exécuter la suppression.

Après l'exécution de cette commande, l'entrée "EFI Floppy" (Boot0001) sera supprimée.



4. Créer une entrée de démarrage

Voici un exemple pour créer une nouvelle entrée de démarrage UEFI pour un système d'exploitation spécifique en utilisant la commande sudo efibootmgr -c -d /dev/sda -p 1 -L "Ubuntu20" -l '\EFI\ubuntu\bootx64.ef' : l'option -c permet de créer l'entrée, -d /dev/sda spécifie le disque, -p 1 indique la partition, -L "Ubuntu20" attribue un nom à l'entrée, et -l '\EFI\ubuntu\bootx64.efi' définit le chemin du fichier de démarrage.



```
BootNext: 0004
BootCurrent: 0006
Timeout: 10 seconds
BootOrder: 0001,0006,0000,0002,0003,0004,0005
Boot0000* EFI VMware Virtual SCSI Hard Drive (0.0)
Boot0002* EFI VMware Virtual SATA CDROM Drive (0.0)
Boot0003* EFI VMware Virtual SATA CDROM Drive (1.0)
Boot0004* EFI Network
Boot0005* EFI Internal Shell (Unsupported option)
Boot0006* ubuntu
Boot0001* Ubuntu20
```

5. Amorcer temporairement sur une autre entrée

Pour démarrer exceptionnellement sur une autre entrée UEFI lors du prochain démarrage sans modifier l'ordre de démarrage permanent(BootOrder), utilisez l'option BootNext avec la commande sudo efibootmgr -n <numeroEntrée>.

Comme illustré dans la capture ci-dessous, pour amorcer sur l'entrée "EFI Network" (numéro 0004), exécutez sudo efibootmgr -n 0004. Cette commande configure BootNext sur 0004, permettant au système de démarrer sur cette entrée lors du prochain démarrage, tout en conservant l'ordre de démarrage permanent inchangé pour les redémarrages futurs.

```
esprit@esprit-virtual-machine:~/Desktop$ sudo efibootmgr -n 0004
BootNext: 0004
BootCurrent: 0006
Timeout: 10 seconds
BootOrder: 0006,0000,0002,0003,0004,0005
Boot0000* EFI VMware Virtual SCSI Hard Drive (0.0)
Boot0002* EFI VMware Virtual SATA CDROM Drive (0.0)
Boot0003* EFI VMware Virtual SATA CDROM Drive (1.0)
Boot0004* EFI Network
Boot0005* EFI Internal Shell (Unsupported option)
Boot0006* ubuntu
```

6. Définir l'intervalle de temporisation du gestionnaire de démarrage EFI

La commande efibootmgr -t <délai> est utilisée pour configurer le délai d'attente gestionnaire de démarrage EFI. Comme le montre la capture d'écran ci-dessous, l'exécution de sudo efibootmgr -t 10 règle le délai d'attente à 10 secondes. Cela signifie q gestionnaire de démarrage EFI attendra 10 secondes avant de passer automatiquement à l'entrée de démarrage par défaut, ou à la première entrée dans l'ordre de démarrage spécifié si aucune autre entrée n'est sélectionnée par l'utilisateur.

```
esprit@esprit-virtual-machine:~/Desktop$ sudo efibootmgr -t 10
BootCurrent: 0006
Timeout: 10 seconds
BootOrder: 0006,0000,0002,0003,0004,0005
Boot0000* EFI VMware Virtual SCSI Hard Drive (0.0)
Boot0002* EFI VMware Virtual SATA CDROM Drive (0.0)
Boot0003* EFI VMware Virtual SATA CDROM Drive (1.0)
Boot0004* EFI Network
Boot0005* EFI Internal Shell (Unsupported option)
Boot0006* ubuntu
```

Deuxième Etape: MBR / GPT

Nous avons déjà découvert la première étape du démarrage d'un ordinateur avec le BIOS et l'UEFI, qui préparent le terrain pour l'initialisation du système. À présent, nous nous penchons sur: le Master Boot Record (MBR) et sa version avancée, moderne et améliorée, la GUID Partition Table (GPT). Ces deux mécanismes sont essentiels pour le partitionnement des disques et le chargement efficace du système d'exploitation. Dans cette partie, nous explorerons le fonctionnement du MBR et du GPT, en mettant en lumière leurs principales différences ainsi que leurs avantages et inconvénients respectifs.

1. Qu'est-ce que le MBR ?

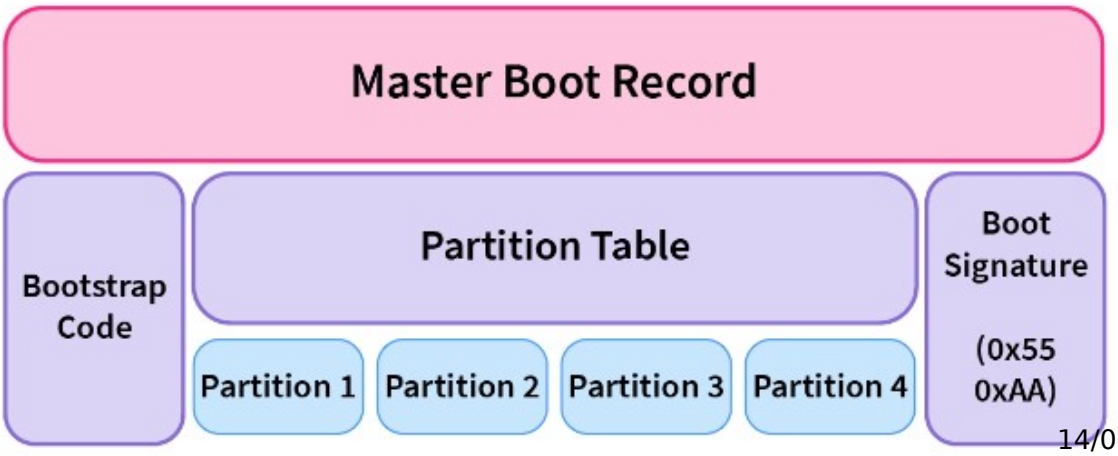
Le Master Boot Record (MBR) est une petite zone de données située dans le premier secteur d'un disque dur, souvent désignée comme le « secteur zéro ». Cette zone est cruciale car elle contient des informations essentielles nécessaires au démarrage de l'ordinateur. On l'appelle également « bloc de démarrage principal ».

Le MBR joue un rôle clé dans le processus de démarrage d'un ordinateur. Lorsqu'on allume l'ordinateur, le BIOS (Basic Input/Output System) effectue un certain nombre de vérifications. L'une de ses premières actions consiste à examiner le MBR pour y trouver le chargeur de démarrage (bootloader). Ce chargeur est un programme qui permet de charger le système d'exploitation dans la mémoire principale de l'ordinateur, ce qui initie le processus de démarrage. Nous examinerons plus en profondeur le fonctionnement du bootloader dans la prochaine section du cours.

Le MBR contient également la table des partitions, qui définit comment le disque est structuré. Une partition de disque est une partie d'un disque dur qui peut être utilisée pour stocker des fichiers ou installer un système d'exploitation. Cette table de partition inclut des informations sur les partitions présentes sur le disque, comme leur taille, leurs secteurs de début et de fin, et le type de système de fichiers utilisé.

Composants de MBR (Master boot record)

Le Master Boot Record (MBR) se compose de trois éléments significatifs nécessaires au processus de démarrage :



1. Boot Code : C'est un petit programme exécuté lorsque l'ordinateur démarre après que le BIOS a chargé le MBR. Sa principale fonction est de trouver la partition active, qui est marquée pour charger le système d'exploitation, et de transférer le contrôle au chargeur d'amorçage (bootloader) de cette partition.
2. Partition Table : Cette table contient des informations sur les partitions présentes sur le disque. Elle indique la position et la taille de chaque partition ,ainsi qu'un indicateur qui marque la partition active, celle à partir de laquelle le système d'exploitation sera chargé.
3. Boot Signature :La boot signature se réfère aux deux derniers octets du MBR, toujours fixés à 0x55AA sur tous les ordinateurs modernes. Ce code est utilisé par le BIOS pour vérifier si le périphérique de démarrage sélectionné peut réellement démarrer. Lors du chargement du MBR, le BIOS examine ces octets. S'ils ne correspondent pas à 0x55AA, le BIOS passe au périphérique suivant dans la liste de démarrage configurée (comme une clé USB ou un disque dur). En l'absence de toute signature correcte, le BIOS affichera un message d'erreur comme "No boot device is available" ou "Reboot and select proper boot device".

Caractéristiques de MBR

- Capacité de chargement du démarrage : Dans le MBR, le code du chargeur de démarrage est présent. Le chargeur de démarrage, qui est utile pour charger le système d'exploitation (https://www.geeksforgeeks.org/what-is-an-operating-system/) dans la mémoire principale, appelé processus de démarrage.
- Facilité d'utilisation : Il est très facile à utiliser, de sorte que les utilisateurs peuvent facilement gérer et entretenir leurs partitions de disque avec MBR.
- Identification du disque : Pour reconnaître le disque, le MBR utilise une signature de disque.
- Compatibilité : La majorité des systèmes d'exploitation, y compris Windows, Linux et macOS, ainsi qu'une grande variété d'appareils utilisent le Master Boot Record (MBR).

Limitations du Master Boot Record (MBR)

- Capacité du disque : MBR ne peut être compatible qu'avec des disques d'une taille maximale de 2 To , il ne peut donc pas être utilisé pour des disques d'une taille supérieure.
- Limitation due à l'ancien BIOS : MBR ne peut pas être utilisé avec des systèmes UEFI (Unified Extensible Firmware Interface) plus récents, car il ne peut être utilisé qu'avec des systèmes BIOS (Basic Input/Output System) antérieurs.
- Manque de redondance : le MBR ne dispose d'aucun mécanisme de redondance intégré, de sorte que des dommages au MBR pourraient empêcher un système de démarrer.
- Partitions limitées : le MBR peut prendre en charge jusqu'à quatre partitions principales, ou trois partitions primaires et une partition étendue.

principales, ou trois partitions primaires et une partition étendue.

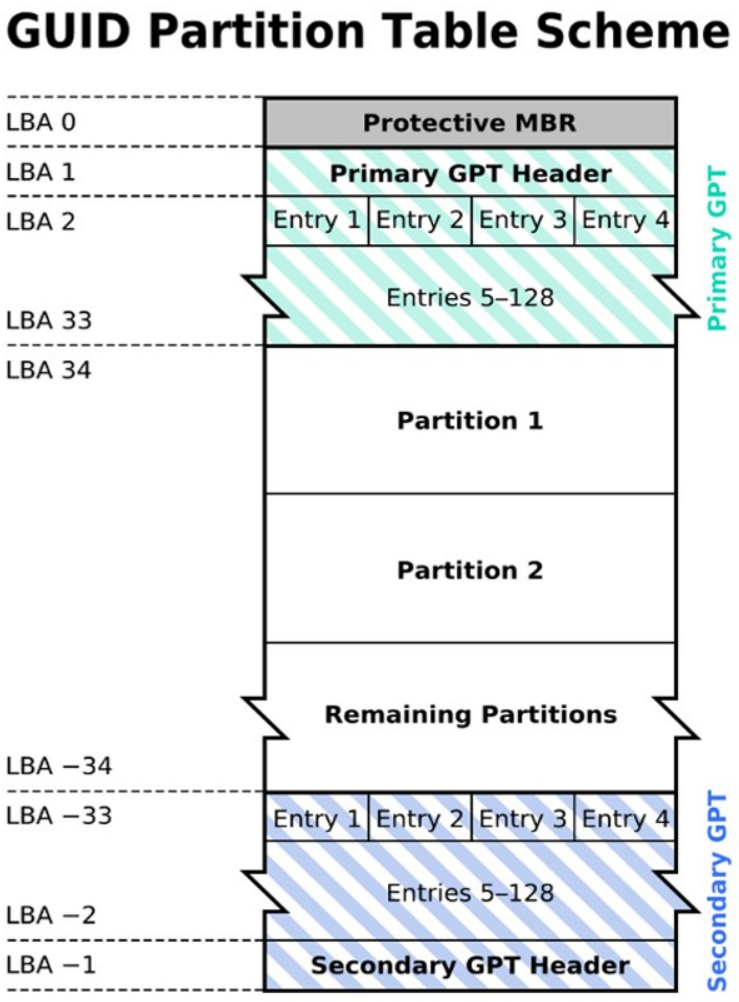
L'évolution des besoins en stockage et les limitations du Master Boot Record (MBR) ont conduit à l'apparition de GPT (GUID Partition Table).

2. Qu'est-ce que le GPT ?

Le GUID Partition Table (GPT) est une structure de données utilisée sur les disques durs et autres périphériques de stockage, principalement dans les systèmes compatibles avec l'EFI (Extensible Firmware Interface). Contrairement au MBR, le GPT utilise des identifiants uniques (GUID) pour chaque partition sur le disque. Cette approche avancée permet une gestion efficace des partitions, même sur des disques de grande capacité.

Le rôle central du GPT réside dans la gestion des partitions et le démarrage des ordinateurs équipés de l'UEFI. Au démarrage, le firmware UEFI utilise les informations du GPT pour localiser le chargeur de démarrage du système d’exploitation ainsi que toutes les données relatives aux partitions. En plus de la table des partitions, le GPT inclut des informations de secours qui renforcent la résilience du système, assurant une intégrité accrue des données et une récupération améliorée en cas de panne.

Structure GPT



La GUID Partition Table (GPT) utilise un système moderne d’adressage par blocs logiques (LBA), remplaçant la méthode obsolète d’adressage par cylindre-tête-secteur (CHS) utilisée par le Master Boot Record (MBR). Par convention, chaque bloc LBA fait généralement 512 octets, correspondant ainsi à un secteur. La structure de la GPT est organisée en plusieurs éléments clés :

Protective MBR (LBA 0) :

Le MBR protecteur est intégré au début d’un disque pour protéger le schéma de partition GPT. Son rôle principal est d'empêcher les utilitaires de disque (les logiciels conçus pour gérer les disques durs), qui ne reconnaissent que le MBR, de mal interpréter et éventuellement d’écraser le disque GPT. L’ensemble du disque GPT est étiqueté comme une seule partition avec l’ID système 0xEE, indiquant qu'il utilise le GPT. Ainsi, même si un système d'exploitation non compatible détecte le disque, il le considère comme inaccessible, évitant ainsi toute écriture accidentelle.

Primary GPT Header (LBA 1) :

L'en-tête GPT primaire est un composant essentiel de la table de partitionnement GUID (GPT), enregistré dans le deuxième bloc du support de stockage (LBA 1), juste après le MBR. Il définit les blocs utilisables du disque, ainsi que le nombre et la taille des entrées de partition dans la table de partitionnement.

Entrées de partition (LBA 2 à LBA 33) :

Cette zone contient les entrées individuelles pour chaque partition définie sur le disque et décrit jusqu'à 128 partitions. Chaque entrée fournit des informations détaillées sur la partition, y compris son type, sa taille, les adresses LBA de début et de fin, ainsi qu son GUID unique.

Partitions (LBA 34 et suivantes) :

C'est là que les données réelles des partitions sont stockées.

Secondary GPT Header (LBA -1) :

Il s'agit d'une copie miroir de l'en-tête GPT primaire, enregistrée dans le GPT secondaire, située à la fin du disque. Il sert de sauvegarde en cas de corruption de l'en-tête primaire.

Entrées de partition secondaires (LBA -2 à LBA -33) :

Il s'agit d'une copie miroir du tableau des partitions primaire, enregistrée dans le GPT secondaire

==> L’adresse LBA d’un secteur de données est simplement un numéro unique pris dans l’intervalle [0 ... N] où N est le nombre total de secteurs du support. Comme 0 est le

l’intervalle [0 ... N] où N est le nombre total de secteurs du support. Comme 0 est le numéro du tout premier secteur de données c’est N - 1 le numéro du dernier (et non pas N).

GPT vs MBR : Pourquoi Choisir la GUID Partition Table ?

Les systèmes de partitionnement MBR et GPT sont conçus pour gérer l’organisation des données sur les disques durs, mais GPT offre des avantages significatifs en matière de capacité, d’intégrité des données et de fiabilité. Voici quelques-uns des principaux atouts de GPT par rapport à MBR :

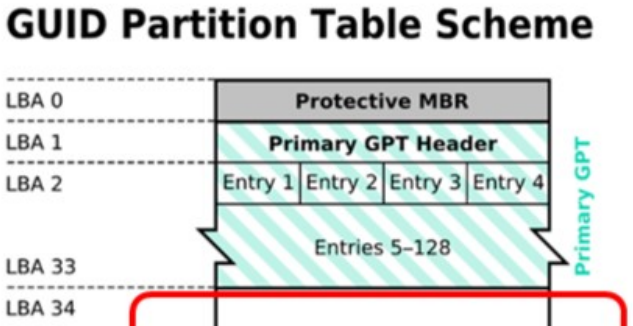
Prise en charge des disques de grande capacité : Contrairement au MBR, qui est limité à des disques de 2 To maximum, le GPT peut gérer des disques de 9,4 ZB (chaque ZB représente 1 milliard de To).

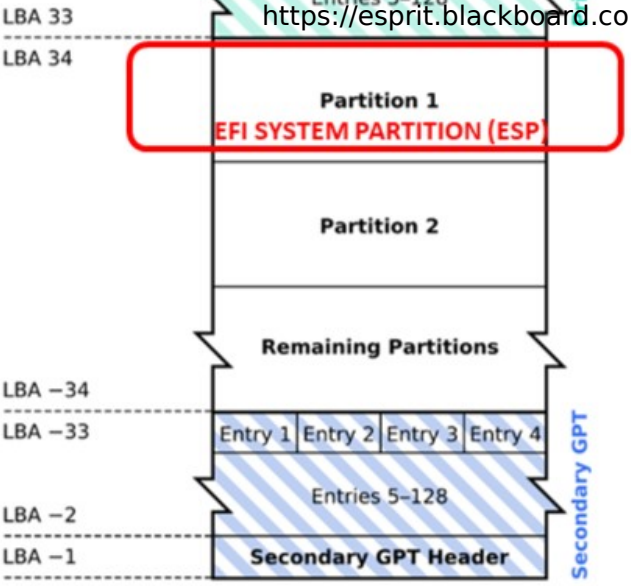
Nombre de partitions : Le GPT permet la création d'un nombre beaucoup plus élevé de partitions par rapport au MBR. Alors que le MBR est limité à 4 partitions primaires , le GPT peut gérer jusqu'à 128 partitions primaires.

Redondance et fiabilité :Les disques partitionnés GPT disposent d'une table de partitions principale ainsi que d'une sauvegarde redondante. GPT conserve les données sur plusieurs zones du disque, avec une copie de sauvegarde en tant que table GPT secondaire pour la récupération (Secondary GPT). En revanche, l'inconvénient du MBR est le stockage des données au même endroit sans sauvegarde, ce qui accroît le risque de corruption et de défaillance du système.

Intégrité des Données: Le GPP utilise le CRC un algorithme de détection d’erreurs qui joue un rôle crucial pour assurer l'intégrité des données. Il est calculé sur différents éléments de la table GPT, notamment l'en-tête primaire et secondaire, ainsi que le tableau des partitions primaire et secondaire.il permet de vérifier si les informations contenues dans ces éléments n'ont pas été corrompues. Si une erreur se produit lors de l'écriture ou de la lecture des données de la table GPT, le CRC calculé ne correspondra plus à la valeur attendue. Le système d'exploitation pourra alors détecter cette erreur et prendre les mesures appropriées, comme lancer une procédure de récupération des données. MBR n’avait aucun moyen de savoir si ses données étaient corrompues - vous ne voyiez qu’il y avait un problème que lorsque le processus de démarrage échouait ou que les partitions de votre disque disparaissaient.

EFI SYSTEM PARTITION





Une partition système EFI (ESP) est une petite partition essentielle située sur les disques durs formatés avec le standard GPT. Elle est cruciale pour le démarrage des ordinateurs utilisant le firmware UEFI. Généralement comprise entre 100 et 250 Mo, l'ESP est souvent positionnée en première partition du disque (par exemple, /dev/sda1). Elle contient les fichiers nécessaires au démarrage des systèmes d'exploitation, tels que les chargeurs de démarrage (bootloaders) et les fichiers de configuration requis par UEFI pour initier le système d'exploitation.

Si votre disque dur est initialisé avec le schéma de partitionnement GUID (GPT), une partition système EFI (ESP) est automatiquement générée lors de l'installation du système d'exploitation. Les systèmes d'exploitation modernes, comme Windows et Linux, incluent cette étape dans leur processus d'installation

Sous une machine Linux, les fichiers courants que l'on trouve dans une partition ESP, généralement située sous le chemin /boot/efi/EFI, incluent :

```
root@esprit-virtual-machine:~# cd /boot/efi/EFI/
root@esprit-virtual-machine:/boot/efi/EFI# ls
BOOT  ubuntu
root@esprit-virtual-machine:/boot/efi/EFI# ls BOOT/
BOOTX64.EFI  fbx64.efi  mmx64.efi
root@esprit-virtual-machine:/boot/efi/EFI# ls ubuntu/
BOOTX64.CSV  grub.cfg  grubx64.efi  mmx64.efi  shimx64.efi
```

- BOOTX64.EFI : Il est souvent sélectionné automatiquement par le firmware UEFI comme chargeur de démarrage principal si aucun autre n'est configuré. Il est également fréquemment utilisé comme chargeur de démarrage de secours lorsque d'autres options ne sont pas disponibles.
- fbx64.efi : Ce fichier est utilisé pour gérer les entrées NVRAM (Non-Volatile Random Access Memory). L'NVRAM stocke des informations de configuration importantes, comme l'ordre de démarrage. Si certaines entrées sont manquantes, fbx64.efi peut tenter de les restaurer.
- shimx64.efi : Il permet le démarrage de GRUB2, un chargeur de démarrage populaire, ou d'autres chargeurs de démarrage tout en respectant les exigences de Secure Boot, si ce dernier est activé. Il sert d'intermédiaire entre le firmware UEFI et le chargeur de démarrage principal.

le firmware UEFI et le chargeur de démarrage principal.

- mmx64.efi : le MOK (Machine Owner Key) est une clé cryptographique qui est utilisée pour vérifier la signature numérique des logiciels. Le fichier mmx64.efi gère le MOK et permet de sécuriser davantage le processus de démarrage en autorisant uniquement l'exécution des composants et pilotes du système d'exploitation approuvés.
- grubx64.efi : Il s'agit du fichier exécutable du chargeur de démarrage (bootloader) GRUB, utilisé dans les systèmes EFI. Ce fichier est responsable du chargement et de la présentation du menu GRUB, à partir duquel l'utilisateur peut sélectionner le système d'exploitation à démarrer.
- BOOTX64.CSV : C'est un fichier lu lors de l'installation pour ajouter une entrée de démarrage dans la NVRAM du système.

Troisième Etape: Bootloader GRUB

Après avoir exploré les premières étapes du démarrage d'un PC Linux, du BIOS/UEFI aux schémas de partitionnement MBR et GPT, nous allons maintenant aborder une phase cruciale du processus : le bootloader. Le bootloader joue un rôle essentiel en servant de pont entre le BIOS/UEFI et le système d'exploitation. Il prend le relais après le démarrage initial pour préparer le système à charger le noyau Linux.

Dans cette section, nous allons approfondir la fonction et le rôle du bootloader, avec un focus particulier sur GRUB, l'un des bootloaders les plus couramment utilisés dans les environnements Linux. En examinant le fonctionnement de GRUB, sa configuration et ses fonctionnalités, vous serez mieux préparés à gérer et personnaliser le processus de démarrage de vos systèmes Linux.

1. Le Chargeur de démarrage(Bootloader) : Définition, Emplacement, Fonctionnement et Exemples

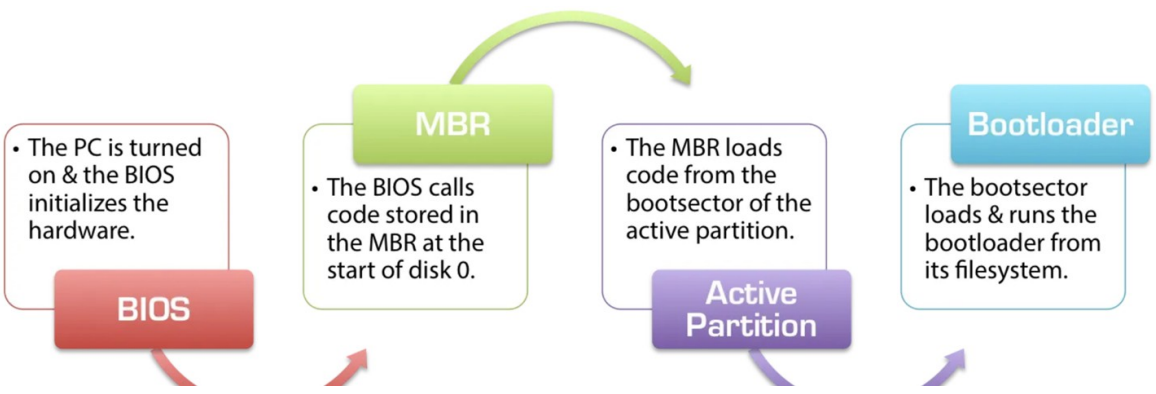
Définition

Pour que le système d'exploitation puisse fonctionner, ses composants doivent être chargés dans la mémoire vive (RAM) lors du démarrage de l'appareil. Cette opération est rendue possible par un programme appelé le chargeur de démarrage, ou bootloader. Le bootloader est un logiciel crucial qui joue le rôle de « chargeur réel » du système d'exploitation. Plus précisément, il est responsable du chargement du noyau du système d'exploitation en mémoire, ce qui permet au système de démarrer et de fonctionner correctement.

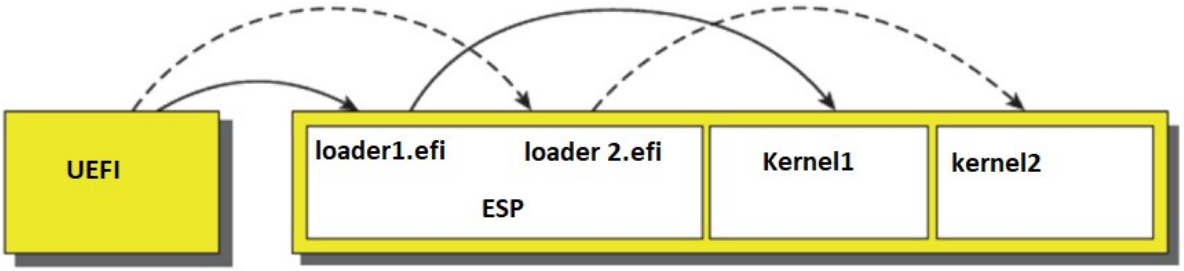
Emplacement du Bootloader

Le bootloader se trouve dans des emplacements spécifiques en fonction du schéma de partitionnement du disque :

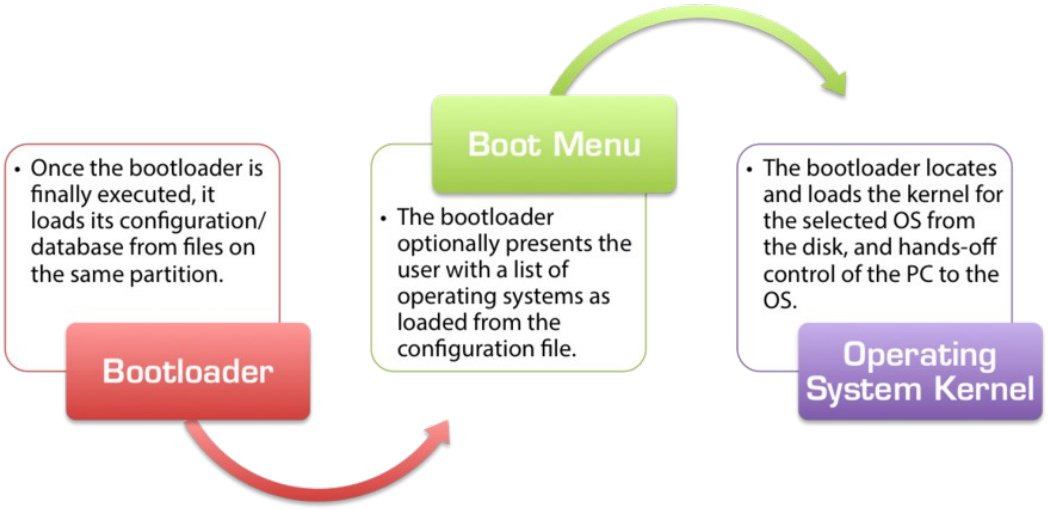
- MBR (Master Boot Record) : Dans les systèmes utilisant le MBR, le bootloader est situé dans le secteur de démarrage principal (boot sector) du disque,



- GPT (GUID Partition Table) : Dans les systèmes utilisant GPT, le bootloader est localisé dans la partition dédiée à l'amorçage EFI (ESP - EFI System Partition). L'ESP est une partition spécialement conçue pour stocker les fichiers nécessaires au démarrage du système, y compris le bootloader. Les bootloaders sont enregistrés sous forme de fichiers avec l'extension .efi, situés dans des sous-répertoires nommés d'après le système d'exploitation sous le répertoire EFI de l'ESP. Par exemple, un bootloader peut se trouver à /boot/efi/EFI/ubuntu/grub.efi. L'UEFI et GPT fonctionnent de manière intégrée : l'UEFI charge directement le bootloader du système d'exploitation, enregistré sous la forme d'un fichier .efi sur la partition EFI du disque GPT.



Fonctionnement du Bootloader



Examinons maintenant le fonctionnement du bootloader pour comprendre comment il garantit le chargement correct du système d'exploitation:

1. Initialisation et Configuration : Après que le BIOS/UEFI ait effectué les tests de démarrage et initialisé le matériel, il recherche le secteur d’amorçage (le MBR pour les anciens systèmes, ou ESP dans le GPT pour les plus récents). C'est à cet endroit que se trouve le bootloader. Celui-ci prend le relais, configure les paramètres nécessaires pour le chargement du système d'exploitation et prépare l’environnement mémoire pour le

noyau.

2. Menu de Boot : Dans les systèmes multi-boot, le bootloader propose une interface permettant à l'utilisateur de sélectionner parmi différents systèmes d’exploitation ou noyaux avant de poursuivre le processus de démarrage.
3. Localisation et Chargement du Noyau : Le bootloader localise le noyau Linux (ainsi que d'autres composants essentiels) du système d'exploitation sélectionné sur le disque ou un autre support de stockage. Il le charge en mémoire vive et passe les paramètres de démarrage nécessaires.
4. Transfert de Contrôle : Une fois le noyau correctement chargé, le bootloader transfère le contrôle au noyau du système d'exploitation, permettant au système de continuer le processus de démarrage et d’initier les services nécessaires pour une opération normale.

Les meilleurs chargeurs de démarrage Linux avec lesquels travailler

En ce qui concerne les chargeurs de démarrage dans l'écosystème Linux, plusieurs options sont disponibles, chacune offrant ses propres fonctionnalités et avantages. Le choix d'un chargeur de démarrage dépend de divers facteurs, tels que la compatibilité du système, les fonctionnalités requises et les préférences personnelles. Parmi les options les plus populaires, GRUB et LILO se distinguent :

- GRUB (Grand Unified Bootloader) : GRUB est l'un des chargeurs de démarrage les plus utilisés et les plus polyvalents sous Linux. Il offre une prise en charge étendue des systèmes d'exploitation, une interface conviviale et des fonctionnalités puissantes telles que le double démarrage, les menus de démarrage personnalisables et la compatibilité avec divers systèmes de fichiers. Sa flexibilité et ses capacités de configuration en font le chargeur de démarrage(Bootloader) par défaut pour de nombreuses distributions Linux.
- LILO (Linux Loader) : LILO est un autre chargeur de démarrage bien établi, connu pour sa simplicité et sa fiabilité. Bien qu'il ne possède pas autant de fonctionnalités avancées que GRUB, il peut gérer diverses configurations de disque et plusieurs options de démarrage.

Nous nous concentrerons principalement sur GRUB pour le reste du cours, car il est le bootloader par défaut dans de nombreuses distributions Linux et offre une gamme étendue de fonctionnalités.

2. Comprendre GRUB comme Chargeur de Démarrage sous Linux

Qu'est-ce que GRUB ?

GRUB (également connu sous le nom de GNU GRUB ou GNU Grand Unified Bootloader) est un chargeur de démarrage et un gestionnaire de démarrage largement utilisé pour Linux et d'autres systèmes d'exploitation basés sur Unix. Selon l'architecture du système, GRUB démarre après que le BIOS ou l'UEFI a terminé les vérifications matérielles. Sur les systèmes BIOS, GRUB est chargé à partir du secteur de démarrage du Master Boot Record (MBR). En revanche, sur les systèmes UEFI, GRUB est chargé à partir de la partition EFI (ESP), identifiée par le GUID unique « C12A7328-F81F-11D2-BA4B-00A0C93EC93B ». Une fois en mémoire, GRUB prend le contrôle du système et procède au chargement du noyau Linux.

GRUB prend le contrôle du système et procède au chargement du noyau Linux.

Il existe actuellement deux versions de GRUB disponibles pour Linux :

- GRUB Legacy (GRUB 1) : Cette version obsolète de GRUB est encore présente sur certaines anciennes distributions Linux, mais elle n'est plus activement maintenue.
- GRUB 2 : La version actuelle de GRUB, qui prend en charge les nouvelles architectures PC (comme ARM), ainsi que des systèmes de fichiers modernes (comme ReiserFS), les environnements RAID et LVM. La plupart des distributions Linux récentes utilisent GRUB 2 en raison de ses fonctionnalités améliorées et de son support étendu.

GRUB: Chargeur de démarrage et gestionnaire de démarrage

GRUB joue à la fois le rôle de chargeur de démarrage(Bootloader) et de gestionnaire de démarrage(Bootmanager), chacun ayant des fonctions spécifiques au démarrage du système:

- Le chargeur de démarrage est responsable du transfert du noyau du système depuis un périphérique de stockage (tel qu'un disque dur ou un SSD) vers la RAM, ce qui correspond au chargement du noyau du système d'exploitation.
- Le gestionnaire de démarrage offre une interface utilisateur permettant de gérer les options de démarrage et de sélectionner le système à charger. Cette interface peut se présenter sous la forme d'un menu GRUB par défaut ou être personnalisée pour des distributions spécifiques, Voici un exemple de menu GRUB2 d'Ubuntu :

Une fois que le BIOS ou UEFI a chargé GRUB en mémoire, GRUB présente à l'utilisateur un menu avec des options de démarrage. Ces options incluent les systèmes d'exploitation installés et des options de démarrage spéciales, telles que le mode sans échec(Recovery Mode). Après que l'utilisateur a sélectionné une option ou que le délai pour faire un choix a expiré, GRUB charge le noyau correspondant.

Fonctionnalités de GRUB

GRUB est un chargeur de démarrage polyvalent et robuste, conçu pour répondre aux besoins des systèmes d'exploitation modernes. Voici les fonctionnalités essentielles de GRUB :

- Configuration flexible : Les utilisateurs peuvent personnaliser divers aspects du démarrage, tels que l'entrée par défaut, les paramètres du noyau et les délais, en modifiant le fichier de configuration de GRUB.
- Prise en charge du démarrage multiple . GRUB fournit un menu de démarrage et permet aux utilisateurs de systèmes multi-OS de choisir facilement le système d'exploitation à exécuter.

facilement le système d'exploitation à exécuter.

- Détection dynamique : GRUB détecte automatiquement les systèmes d'exploitation installés et génère les entrées de démarrage correspondantes.
- Langage de script : GRUB permet de créer des configurations de démarrage personnalisées et d'automatiser des tâches via son langage de script.
- Protection par mot de passe : Pour renforcer la sécurité, GRUB offre la possibilité de protéger le menu du chargeur de démarrage par mot de passe.
- Mode de secours (Recovery Mode) : Le mode de secours aide les utilisateurs à dépanner et à résoudre les problèmes liés au démarrage.
- Compatibilité BIOS et UEFI : GRUB fonctionne avec les systèmes BIOS traditionnels ainsi qu'avec le micrologiciel UEFI moderne.

Comment accéder et modifier GRUB2 ?

Pour accéder et modifier GRUB2, il vous suffit d'éditer ses fichiers de configuration via le terminal avec des privilèges d'administrateur. Ces ajustements permettent de personnaliser l'expérience de démarrage, d'adapter les options de démarrage aux besoins spécifiques et d'améliorer l'interaction utilisateur avec le menu de démarrage. Voici quelques exemples de modifications courantes que vous pouvez apporter à GRUB2 pour personnaliser son fonctionnement et son apparence selon vos préférences:

- Changer le Délai d’Attente du Menu : Ajuster le temps pendant lequel le menu GRUB2 est affiché avant de démarrer automatiquement le système d’exploitation par défaut.
- Définir le Système d’Exploitation par Défaut : Choisir quel système d’exploitation ou noyau sera démarré par défaut lorsque le temps d’attente expire.
- Ajouter des Options de Démarrage au Noyau : Passer des options supplémentaires au noyau Linux lors du démarrage pour personnaliser le comportement du système.
- Changer le Fond d’Écran de GRUB : Personnaliser l'apparence de l'écran de démarrage en ajoutant une image de fond.
- Ajouter un Nouveau Système d’Exploitation au Menu : Inclure une entrée pour un autre système d’exploitation qui n'est pas automatiquement détecté.
- Désactiver l’Affichage du Menu GRUB : Passer directement au système d’exploitation par défaut sans afficher le menu GRUB2.
- Activer le Mode de Débogage de GRUB : Activer des messages de débogage pour aider à diagnostiquer des problèmes lors du démarrage.

GRUB utilise plusieurs fichiers de configuration qui peuvent être modifiés pour personnaliser le comportement et l’apparence du chargeur de démarrage. Voici les 3 principaux fichiers de GRUB que vous pouvez éditer :

1. /etc/default/grub :

C'est le fichier principal de GRUB2. Comme le montre la capture ci-dessous, il contient les paramètres par défaut qui influencent le comportement global du chargeur de démarrage.

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US"

# 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
```

Voici quelques paramètres fréquemment modifiés :

- GRUB_DEFAULT détermine l'entrée (système d'exploitation) sélectionnée par défaut dans le menu de GRUB. Une valeur de 0 démarre la première entrée de menu, 1 démarre la deuxième, et ainsi de suite. En utilisant la valeur "saved", GRUB charge le dernier système d'exploitation qui a été démarré avec succès.
- GRUB_TIMEOUT_STYLE définit le style d'affichage du menu GRUB pendant le délai d'attente. Il peut prendre les valeurs suivantes : menu affiche le menu avec le délai visible, hidden masque le menu et montre uniquement un écran de démarrage, tandis que hidden_timeout masque le menu mais affiche un délai d'attente avec la possibilité de révéler le menu en appuyant sur une touche.
- GRUB_TIMEOUT définit le nombre de secondes pendant lesquelles GRUB affiche le menu de démarrage avant de démarrer le système d'exploitation par défaut. Définissez sur 0 pour démarrer immédiatement sans afficher le menu, ou sur -1 pour attendre indéfiniment
- GRUB_DISTRIBUTOR définit le titre affiché dans le menu de GRUB. Par défaut, il est automatiquement basé sur la distribution installée (par exemple Ubuntu), mais peut être modifié pour personnaliser l'affichage si vous souhaitez afficher un nom différent pour votre système.
- GRUB_CMDLINE_LINUX_DEFAULT Ce paramètre définit les options de démarrage par défaut du noyau utilisées lors du démarrage normal du système. Par exemple, vous pouvez ajouter des paramètres comme splash pour afficher un écran de démarrage graphique, ou quiet pour réduire les messages affichés à l'écran pendant le démarrage.
- GRUB_CMDLINE_LINUX Ce paramètre est utilisé pour spécifier des options supplémentaires qui seront appliquées en mode récupération. Par exemple, vous pouvez ajouter fsck.mode=force pour forcer la vérification du système de fichiers, ou fsck.repair=yes pour réparer automatiquement les erreurs détectées.

2. /etc/grub.d/

Ce répertoire contient plusieurs scripts qui génèrent les fichiers de configuration de GRUB. Vous pouvez ajouter ou modifier ces scripts pour personnaliser le menu de démarrage et les options disponibles.

```
esprit@esprit-virtual-machine:~$ cd /etc/grub.d
esprit@esprit-virtual-machine:/etc/grub.d$ ls
00_header        10_linux      20_linux_xen   30_os-prober      35_fwupd   41_custom
05_debian_theme  10_linux_zfs  20_memtest86+  30_uefi-firmware  40_custom  README
```

Voici les scripts situés dans le répertoire /etc/grub.d/, exécutés dans l'ordre numérique par la commande update-grub pour générer le fichier grub.cfg. Les scripts sont traités dans l'ordre de leur numérotation, par exemple, 10_linux est exécuté avant 20_memtest86+. Cet ordre structuré permet de construire progressivement le menu de démarrage :

- 00_header : est le script qui charge les paramètres GRUB à partir de / etc / default / grub, y compris le délai d'expiration, l'entrée de démarrage par défaut et autres.
- 05_debian_theme : script pour configurer le thème, y compris l'arrière-plan , les couleurs et les polices, pour personnaliser l'interface du menu de démarrage.
- 10_linux : ce script identifie les distributions Linux installées sur votre système et génère les entrées de menu correspondantes dans GRUB. Il s'assure que les noyaux disponibles sont listés, ainsi que les options de mode de récupération, si elles sont activées;
- 20_linux_xen : ce Script gère les entrées de menu pour les noyaux Linux utilisant le hyperviseur Xen. Il détecte les noyaux Xen installés, ajoute les options de démarrage nécessaires pour le support de la virtualisation Xen, et les configure dans le menu de GRUB. ;
- 20_memtest86+ : ce Script ajoute une entrée pour Memtest86+ dans le menu de GRUB. Memtest86+ est un outil de test de la mémoire vive (RAM) qui permet de vérifier la présence d'erreurs dans la mémoire du système. Ce script détecte Memtest86+ installé sur le système et configure l'entrée correspondante pour qu'il soit accessible depuis le menu de démarrage de GRUB.
- 30_os-prober : si vous avez un système d'exploitation non basé sur Linux installé sur votre disque dur, ce fichier de script trouvera ce système d'exploitation et créera l'entrée appropriée pour celui-ci. En d'autres termes, si Windows est installé sur votre système, il le trouvera automatiquement et créera une entrée appropriée pour celui-ci;
- 30_uefi-firmware : C'est un script qui permet d'ajouter des entrées au menu de démarrage afin d'accéder directement aux paramètres du firmware UEFI de votre système. ;
- 35_fwupd : Le script permet d'ajouter une entrée afin de vérifier et, si nécessaire, de mettre à jour le firmware de différents composants matériels (comme les cartes graphiques, les réseaux, etc.) lors du démarrage avec l'outil fwupd.
- 40_custom : Permet d'ajouter des entrées personnalisées au menu GRUB, comme des systèmes d'exploitation supplémentaires ou des options de démarrage spécifiques. Les entrées définies dans ce fichier apparaissent après les entrées générées automatiquement par les autres scripts.
- 41_custom : Fonctionne de manière similaire à 40_custom, mais est exécuté après ce dernier. Il est souvent utilisé pour des personnalisations supplémentaires ou des ajustements spécifiques qui doivent être appliqués après les entrées définies dans 40_custom.

3. /boot/grub/grub.cfg

C'est le fichier de configuration final et exécutable de GRUB2. Il est généré automatiquement par le programme grub-mkconfig et mis à jour par la commande update-grub à partir des paramètres définis dans les fichiers de configuration principaux (/etc/default/grub) et des scripts présents dans le répertoire /etc/grub.d/. Il est non recommandé de modifier ce fichier directement, car vos changements seront écrasés lors de la prochaine mise à jour de GRUB.

Remarque: Après chaque modification dans /etc/default/grub ou dans les scripts de /etc/grub.d/, n'oubliez pas de lancer la commande sudo update-grub pour enregistrer les changements dans le fichier /boot/grub/grub.cfg. Sinon, les modifications apportées ne seront pas prises en compte et le fichier /boot/grub/grub.cfg restera inchangé.

Système d'initialisation: Définition et Rôles et Type

Système d'initialisation: Définition

Après le chargement du noyau Linux, le système n'est pas encore entièrement opérationnel. Pour que l'ordinateur devienne pleinement fonctionnel, il est nécessaire de démarrer et de configurer les services requis ainsi que l'interface utilisateur. C'est à ce stade que le système d'initialisation entre en jeu.

Le noyau délègue au système d'initialisation la tâche de compléter le processus de démarrage, en suivant la configuration établie par l'administrateur. Le système d'initialisation est responsable du lancement des services en arrière-plan, tels que le montage des périphériques de stockage, la configuration du réseau, la gestion des services de messagerie, ainsi que la mise en place de l'interface utilisateur, qu'elle soit graphique ou en ligne de commande.

Le premier processus utilisateur lancé par le noyau joue un rôle crucial, car il gère le démarrage et la supervision de tous les autres processus. Ce processus, portant le « PID 1 » (Processus n°1), est le premier à être lancé lors du démarrage de la machine et le dernier à s'arrêter, car il supervise tous les autres processus.

Traditionnellement, ce premier processus était init, le système d'initialisation traditionnel utilisé dans les premières versions de Linux. Plus récemment, SysVinit, une implémentation spécifique de init, était largement utilisé, mais il est désormais souvent remplacé par systemd dans les distributions modernes. Systemd offre des fonctionnalités avancées pour la gestion des services et des processus, telles que le démarrage parallèle des services et une gestion améliorée des dépendances.

Ainsi, le système d'initialisation est un composant essentiel du système ; s'il ne fonctionne pas correctement, la machine ne pourra pas démarrer. Cela souligne l'importance de sa robustesse et de sa fiabilité dans le bon fonctionnement du système.

Quelques Rôles principaux du système d'initialisation :

1. Démarrage des services essentiels :

- Montage des systèmes de fichiers: Il s'assure que tous les systèmes de fichiers nécessaires sont montés, rendant ainsi accessibles les données et les applications.
- Lancement des démons: Les démons sont des processus qui s'exécutent en arrière-plan et fournissent des services comme le réseau, les processus d'impression, les bases de données, etc.
- Initialisation de la console: Il prépare la console pour l'interaction utilisateur, en lançant un shell (comme Bash) par exemple.

2. Gestion des niveaux d'exécution(runlevels):

- Différents niveaux: Linux définit différents niveaux d'exécution, chacun correspondant à un état du système (mode mono-utilisateur, multi-utilisateur, etc.).
- Transition entre niveaux: Le système d'initialisation permet de passer d'un niveau à un autre, par exemple lors d'un arrêt ou d'un redémarrage.

3. Arrêt du système:

- Arrêt des services: Il ordonne l'arrêt de tous les services en cours d'exécution.
- Fermeture des fichiers: Il s'assure que tous les fichiers ouverts sont correctement fermés.
- Extinction du système: Il envoie le signal d'arrêt au noyau.

4. Gestion des processus orphelins:

- Adoption des processus: Chaque processus doit avoir au moins un processus père, à l'exception du tout premier processus lancé, identifié par le PID 1. Si un processus se détache de son père et devient orphelin, le processus PID 1 adopte automatiquement ces processus orphelins

5. Intégration avec d'autres services :

- Coordination : Le système d'initialisation gère le démarrage et l'arrêt des services système, tels que les gestionnaires de réseau, les serveurs d'affichage et les bases de données. Il assure la coordination nécessaire pour que ces services soient correctement activés ou désactivés en fonction des besoins du système.
- Dépendances : Il prend en charge les dépendances entre les différents services, garantissant que chaque service est démarré dans le bon ordre. Cela signifie que les services critiques sont lancés avant ceux qui en dépendent, assurant ainsi un démarrage fluide et fonctionnel du système.

Les deux types de systèmes d'initialisation les plus connus

1. SysVinit : Un système d'initialisation traditionnel utilisé pour démarrer et gérer les services du système Linux à l'aide de scripts d'initialisation et de niveaux d'exécution.

2. systemd : Un système d'initialisation moderne qui remplace SysVinit, offrant une gestion avancée des services, des dépendances, et un démarrage plus rapide avec des fichiers d'unité.

Résumé de la Séquence de Démarrage d'un PC Sou

Après avoir analysé en profondeur chaque étape du processus de démarrage d'un ordinateur sous Linux, nous allons désormais présenter un récapitulatif complet. Ce résumé met en évidence les étapes essentielles, depuis la mise sous tension jusqu'à l'initialisation des services et la connexion de l'utilisateur. Retrouvez ci-dessous un aperçu détaillé sous forme de vidéo et de texte.

1. Mise sous tension et POST :

- Dès que la machine est mise sous tension le BIOS (Basic Input/Output System) ou UEFI (Unified Extensible Firmware Interface) effectue un autotest du système (POST - Power-On Self-Test). Ce processus vérifie les composants matériels essentiels (mémoire vive, processeur, périphériques de stockage, etc.) pour détecter les défaillances potentielles avant de passer à l'étape suivante.

2. Initialisation des composants :

- UEFI (ou BIOS) active les composants essentiels nécessaires au démarrage, tels que l'affichage vidéo, le clavier et les médias de stockage.

3. Recherche de la partition de démarrage :

- Sous UEFI : L'UEFI recherche une partition GPT (GUID Partition Table) avec un GUID spécifique qui identifie cette partition comme la partition système EFI (ESP - EFI System Partition). Si plusieurs périphériques de démarrage sont disponibles, le gestionnaire de démarrage UEFI sélectionne l'ESP approprié en fonction de l'ordre de démarrage défini dans le firmware UEFI.
- Sous BIOS : La recherche se fait dans le MBR (Master Boot Record) où le BIOS cherche la table de partition pour localiser la partition active ou de démarrage.

4. Chargement du chargeur de démarrage :

- Sous UEFI : Si le démarrage sécurisé n'est pas activé, UEFI exécute directement le chargeur de démarrage GRUB (grubx64.efi) depuis l'ESP. Si le démarrage sécurisé est activé, UEFI charge d'abord l'application shim.efi, qui est responsable de la certification du chargeur de démarrage. Si le certificat est valide, shim.efi charge GRUB. GRUB à son tour valide le noyau à charger.
- Sous BIOS : GRUB (installé sur le MBR) est exécuté pour charger le noyau. Dans les configurations plus anciennes, ce chargeur de démarrage pourrait également être LILO ou un autre chargeur.

5. Chargement et Initialisation du noyau :

- Le chargeur de démarrage charge le fichier image du noyau, vmlinuz, en mémoire. vmlinuz est une version compressée du noyau Linux, qui constitue le cœur du système d'exploitation. Une fois le noyau chargé en mémoire, il commence son processus d'initialisation. Au cours de cette phase, le noyau décompresse et monte initramfs, un système de fichiers temporaire également stocké dans une image compressée. initramfs contient les outils et modules essentiels pour initialiser le système. Il fournit les pilotes et les scripts nécessaires pour détecter et monter le système de fichiers racine.

6. Démarrage du premier processus :

- Le noyau passe le contrôle au programme d'initialisation, init(PID1) pour les systèmes Linux basés sur SysV ou systemd pour les distributions modernes . Ce programme lance les services système essentiels, comme le réseau et les gestionnaires de connexion, ainsi que les scripts de démarrage. Sur les systèmes utilisant systemd, cela se fait à travers des unités et des cibles.

7. Connexion de l’Utilisateur :

- L’environnement de bureau ou l'écran de connexion est présenté, permettant à l'utilisateur de se connecter et de commencer à utiliser le système.

systemd

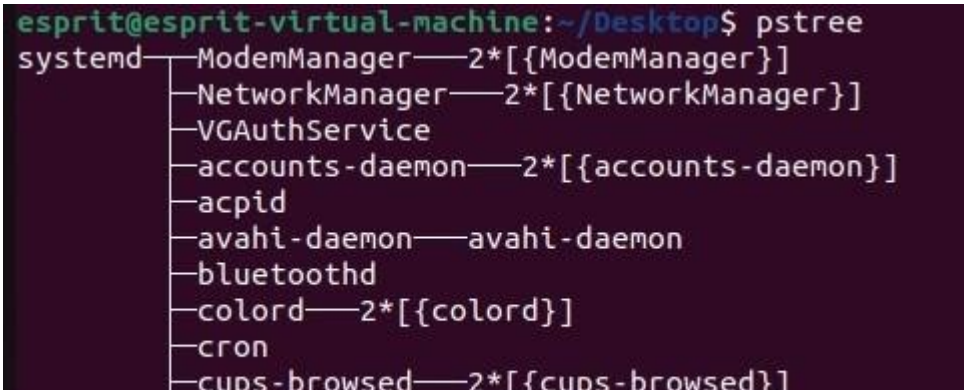
Introduction

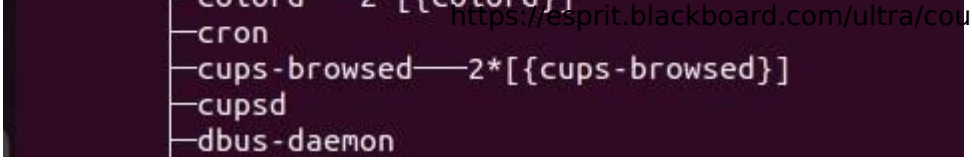
Systemd est un système d'initialisation moderne et complet adopté par de nombreuses distributions Linux. Il s'agit d'un ensemble de programmes conçu pour gérer le démarrage, la surveillance et la gestion des services et des processus sur un système d'exploitation Linux. Systemd vise à améliorer l'efficacité et la performance du système en offrant les fonctionnalités suivantes :

- Démarrage Parallèle des Services : Systemd permet le lancement simultané des services au démarrage du système, réduisant ainsi le temps nécessaire pour l'initialisation.
- Activation à la Demande : Il peut activer des démons et services uniquement lorsqu'ils sont requis, optimisant ainsi les ressources du système.
- Gestion des Dépendances : Systemd gère automatiquement les dépendances entre les services, garantissant que les services sont démarrés et arrêtés dans le bon ordre.
- Journalisation Intégrée : Il inclut des outils tels que journald pour une gestion centralisée et unifiée des journaux système.

Avec l'adoption de systemd, qui a introduit un modèle d'initialisation différent basé sur les unit files plutôt que sur des scripts shell, il aurait pu y avoir une incompatibilité avec les anciens scripts SysV, cependant, les développeurs de systemd ont pris soin de mettre en place un mécanisme de compatibilité. Cela signifie que les systèmes utilisant systemd sont capables de comprendre et d'exécuter les anciens scripts SysV sans problème. cela signifie que les administrateurs système peuvent migrer vers systemd tout en continuant à utiliser leurs anciens scripts d'initialisation SysVinit sans avoir à les réécrire complètement. Cela a contribué à faciliter l'adoption de systemd dans l'écosystème Linux

Exemple : Imaginez systemd comme un chef d'orchestre très organisé. Il dispose de partitions claires pour chaque musicien (service), sait quand chaque musicien doit commencer à jouer (au démarrage) et comment les faire s'arrêter (à l'arrêt). Il surveille également la performance des musiciens en cours de route pour s'assurer qu'ils jouent correctement (supervision). Grâce à sa capacité à gérer les erreurs et les dépendances, systemd apporte une orchestration moderne et efficace, rendant le système Linux plus performant et fiable.





Fonctionnement de systemd

1. Unité

Dans systemd, les services et autres éléments du système sont gérés à l'aide de fichiers appelés unités remplaçant les scripts SysV de SysVinit . Ces fichiers sont utilisés pour configurer et contrôler différents aspects du système. Voici les principaux types d'unités :

- Service (.service) : Gère les services (ou démons) du système, comme le serveur web Apache ou le service de messagerie. Exemple : apache2.service.

Exemple de fichier apache2.service :

```
[Unit]
Description=The Apache HTTP Server

[Service]
ExecStart=/usr/sbin/apache2 -DFOREGROUND
ExecReload=/bin/kill -HUP $MAINPID
ExecStop=/bin/kill -TERM $MAINPID
Restart=always

[Install]
WantedBy=multi-user.target
```

- Socket (.socket) : Gère les sockets réseau ou inter-processus. Un socket peut être utilisé pour la communication inter-processus, permettant la communication entre deux processus. Exemple : ssh.socket pour SSH.
- Target (.target) : Sert à grouper d'autres unités ensemble pour former un état ou un niveau de fonctionnement du système. Exemple : multi-user.target pour le mode multi-utilisateur.
- Mount (.mount) : Gère les points de montage de systèmes de fichiers. Exemple : home.mount pour monter le répertoire /home.
- Timer (.timer) : Exécute des tâches programmées, comme les cron jobs. Exemple : daily.timer pour des tâches à exécuter quotidiennement.

Type d'unité	Extension de fichier	Description
Unité du service	.service	Service système.
Unité cible	.target	Un groupe d'unités systemd.
		Un point Automount du

Unité Automount	.automount	Un point Automount du système de fichiers.
Unité du périphérique	.device	Fichier du périphérique reconnu par le noyau.
Unité de montage	.mount	Point de montage du système de fichiers.
Unité de chemin	.path	Un fichier ou répertoire dans un système de fichiers.
Unité scope	.scope	Un processus créé de manière externe.
Unité de tranche	.slice	Un groupe d'unités organisées de manière hiérarchique qui gèrent des processus système.
Unité d'instantané	.snapshot	Un état enregistré du gestionnaire systemd.
Unité de socket	.socket	Un socket de communication inter-processus.
Unité swap	.swap	Un périphérique ou fichier swap.
Unité minuteur	.timer	Un minuteur systemd.

Les fichiers d'unité sont situés dans deux répertoires principaux :

- /etc/systemd/system/
- /lib/systemd/system/

2. Ciblage (Targets)

Au lieu des runlevels utilisés par SysVinit, systemd utilise des targets pour définir les états opérationnels du système et regrouper d'autres unités Systemd dans une chaîne de dépendances. Les targets sont similaires aux runlevels, mais offrent plus de flexibilité permettant une gestion dynamique et une personnalisation avancée des cibles.

Par exemple, l'unité graphical.target, qui est utilisée pour lancer une session graphique, lance des services systèmes comme le gestionnaire d'affichage GNOME (display-manager.service) et active également l'unité multi-user.target.

```
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

Voici quelques exemples de targets :

- multi-user.target : Correspond au mode multi-utilisateur avec réseau (équivalent au runlevel 3 dans SysVinit).
- graphical.target : Correspond au mode graphique avec interface utilisateur (équivalent au runlevel 5 dans SysVinit).
- rescue.target : Mode de secours pour la maintenance du système (équivalent au runlevel 1).

Tâches / Système	Systemd
Arrêt	poweroff.target
Mono utilisateur	rescue.target
Multi-utilisateur	multi-user.target
Multi-utilisateur avec réseau	multi-user.target
Personnalisable	runlevel4.target
Multi-utilisateur avec réseau et graphique	graphical.target
Reboot	reboot.target

Voici comment systemd utilise les cibles

1. systemd identifie la cible par défaut du système à l'aide du fichier situé à /usr/lib/systemd/system/default.target. Ce fichier de configuration indique quelle cible e active lors du démarrage du système.
2. Les fichiers de configuration pour les cibles sont situés sous le répertoire /usr/lib/systemd/system/. Chaque fichier de cible a une extension .target et définit les services et unités à activer ou à désactiver lorsque la cible est atteinte.
3. systemd positionne le système dans l'état défini par la cible en démarrant, arrêtant ou redémarrant des unités de type service. Lorsqu'un target est activé, systemd lit les fichiers de configuration des cibles pour déterminer quels services doivent êtr démarrés ou arrêtés.

Voici un exemple de fichier default.target :

Open /usr/lib/systemd/system

```
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

3. Utilitaire de Base : systemctl

systemctl est un outil en ligne de commande utilisé pour interagir avec systemd. Il permet aux administrateurs système de démarrer, arrêter, contrôler, configurer et vérifier l'état des services, ainsi que de manipuler des cibles (targets) et d'autres unités du système, tout en consultant les journaux système. systemctl est une composante essentielle de la gestion des systèmes modernes utilisant systemd, offrant la possibilité de gérer efficacement l'état du système et d'ajuster les services en fonction des besoins.

Exemple :Lister Tous les Services :

```
esprit@esprit-virtual-machine:~$ systemctl list-units --type=service
  UNIT                                      LOAD   ACTIVE SUB     D>
  accounts-daemon.service                   loaded active running A>
  acpid.service                             loaded active running A>
  alsa-restore.service                      loaded active exited  S>
  apparmor.service                          loaded active exited  L>
  apport.service                            loaded active exited  L>
  avahi-daemon.service                      loaded active running A>
  bluetooth.service                         loaded active running B>
  colord.service                            loaded active running M>
  console-setup.service                     loaded active exited  S>
  cron.service                              loaded active running R>
  cups-browsed.service                      loaded active running M>
  cups.service                              loaded active running C>
  dbus.service                              loaded active running D>
  gdm.service                               loaded active running G>
  irqbalance.service                        loaded active running i>
  kerneloops.service                        loaded active running T>
  keyboard-setup.service                    loaded active exited  S>
  kmod-static-nodes.service                 loaded active exited  C>
  ModemManager.service                      loaded active running M>
  networkd-dispatcher.service               loaded active running D>
  NetworkManager-wait-online.service        loaded active exited  N>
  NetworkManager.service                    loaded active running N>
```

Exemple : Arrêter le service bluetooth :

```
esprit@esprit-virtual-machine:~$ sudo systemctl stop  bluetooth.service
```

Tâches / Système	Commande Systemd
Démarrer un service	systemctl start service.service

Démarrer un service	systemctl start service.service –
Stopper un service	systemctl stop service.service
Redémarrer un service	systemctl restart service.service
Recharger un service	systemctl reload service.service
Statut d'un service	systemctl status service.service
Activer un service au démarrage	systemctl enable service.service
Désactiver un service au démarrage	systemctl disable service.service
vérifier si un service est activé au boot	systemctl is-enabled service.service

Arrêter le système	systemctl halt
Eteindre le système	systemctl poweroff
Redémarrer le système	systemctl reboot
Suspendre le système	systemctl suspend
Hiberner le système	systemctl hibernate
Afficher les logs systèmes	journalctl -f
Basculer temporairement de runlevel	systemctl isolate nom.target
Changer de runlevel par défaut	systemctl set-default nom.target
Vérifier le runlevel actif	systemctl get-default

4. Journalisation (journalctl)

systemd intègre un système de journalisation appelé journald, qui centralise et gère les logs du système et des services. Avec journalctl, vous pouvez consulter les journaux de manière unifiée. Voici comment utiliser quelques commandes de base :

Description	Commande Complète
Affiche tous les journaux système.	journalctl
Affiche la fin du journal (derniers messages).	journalctl -e

messages).	journalctl -e
Affiche les nouveaux messages en temps réel (mode suivi).	journalctl -f
Affiche les derniers N messages du journal.	journalctl -n 100
Affiche les journaux filtrés par niveau de priorité (err, warning, etc.).	journalctl -p err
Affiche les journaux depuis une date/heure spécifique.	journalctl --since "2024-08-01 12:00:00"
Affiche les journaux pour un service spécifique.	journalctl -u apache2.service

Résumé de la Séquence de Démarrage d'un PC Sou

Après avoir analysé en profondeur chaque étape du processus de démarrage d'un ordinateur sous Linux, nous allons désormais présenter un récapitulatif complet. Ce résumé met en évidence les étapes essentielles, depuis la mise sous tension jusqu'à l'initialisation des services et la connexion de l'utilisateur. Retrouvez ci-dessous un aperçu détaillé sous forme de vidéo et de texte.

1. Mise sous tension et POST :

- Dès que la machine est mise sous tension le BIOS (Basic Input/Output System) ou UEFI (Unified Extensible Firmware Interface) effectue un autotest du système (POST - Power-On Self-Test). Ce processus vérifie les composants matériels essentiels (mémoire vive, processeur, périphériques de stockage, etc.) pour détecter les défaillances potentielles avant de passer à l'étape suivante.

2. Initialisation des composants :

- UEFI (ou BIOS) active les composants essentiels nécessaires au démarrage, tels que l'affichage vidéo, le clavier et les médias de stockage.

3. Recherche de la partition de démarrage :

- Sous UEFI : L'UEFI recherche une partition GPT (GUID Partition Table) avec un GUID spécifique qui identifie cette partition comme la partition système EFI (ESP - EFI System Partition). Si plusieurs périphériques de démarrage sont disponibles, le gestionnaire de démarrage UEFI sélectionne l'ESP approprié en fonction de l'ordre de démarrage défini dans le firmware UEFI.
- Sous BIOS : La recherche se fait dans le MBR (Master Boot Record) où le BIOS cherche la table de partition pour localiser la partition active ou de démarrage.

4. Chargement du chargeur de démarrage :

- Sous UEFI : Si le démarrage sécurisé n'est pas activé, UEFI exécute directement le chargeur de démarrage GRUB (grubx64.efi) depuis l'ESP. Si le démarrage sécurisé est activé, UEFI charge d'abord l'application shim.efi, qui est responsable de la certification du chargeur de démarrage. Si le certificat est valide, shim.efi charge GRUB. GRUB à son tour valide le noyau à charger.
- Sous BIOS : GRUB (installé sur le MBR) est exécuté pour charger le noyau. Dans les configurations plus anciennes, ce chargeur de démarrage pourrait également être LILO ou un autre chargeur.

5. Chargement et Initialisation du noyau :

- Le chargeur de démarrage charge le fichier image du noyau, vmlinuz, en mémoire. vmlinuz est une version compressée du noyau Linux, qui constitue le cœur du système d'exploitation. Une fois le noyau chargé en mémoire, il commence son processus d'initialisation. Au cours de cette phase, le noyau décompresse et monte initramfs, un système de fichiers temporaire également stocké dans une image compressée. initramfs contient les outils et modules essentiels pour initialiser le système. Il fournit les pilotes et les scripts nécessaires pour détecter et monter le système de fichiers racine.

6. Démarrage du premier processus :

- Le noyau passe le contrôle au programme d'initialisation, init(PID1) pour les systèmes Linux basés sur SysV ou systemd pour les distributions modernes . Ce programme lance les services système essentiels, comme le réseau et les gestionnaires de connexion, ainsi que les scripts de démarrage. Sur les systèmes utilisant systemd, cela se fait à travers des unités et des cibles.

7. Connexion de l’Utilisateur :

- L’environnement de bureau ou l'écran de connexion est présenté, permettant à l'utilisateur de se connecter et de commencer à utiliser le système.