Système d'Exploitation (SE)
et
Environnement de Programmation
-9-

Fragmentation
Amorce

Emmanuel Paviot-Adet
IUT Paris Descartes

Stockage d'un fichier physique

- Problèmes
  - Comment stocker les informations sur le disque ?
  - Comment pouvoir les retrouver ?
- Solution simple
  - Stockage contiguë
    - Toutes les informations sont stockées à la suite les unes des autres

Stockage d'un fichier physique

- Problèmes
  - Comment stocker les informations sur le disque ?
  - Comment pouvoir les retrouver ?
- Solution simple
  - Stockage contiguë
    - Toutes les informations sont stockées à la suite les unes des autres

Fichier1 (3 secteurs)	Fichier2 (3 secteurs)	Fichier3 (3 secteurs)	Fichier4 (3 secteurs)

Stockage d'un fichier physique

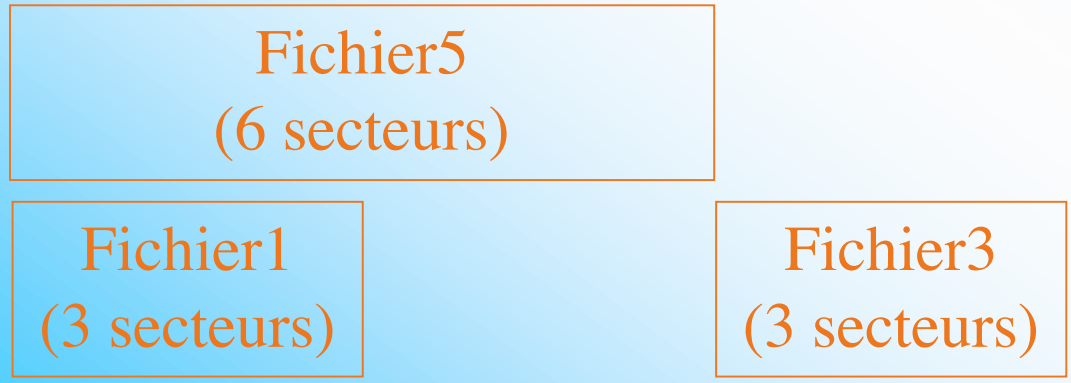
- Problèmes
  - Comment stocker les informations sur le disque ?
  - Comment pouvoir les retrouver ?
- Solution simple
  - Stockage contiguë
    - Toutes les informations sont stockées à la suite les unes des autres

Fichier1
(3 secteurs)

Fichier3
(3 secteurs)

Stockage d'un fichier physique

- Problèmes
  - Comment stocker les informations sur le disque ?
  - Comment pouvoir les retrouver ?
- Solution simple
  - Stockage contiguë
    - Toutes les informations sont stockées à la suite les unes des autres



Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

Fichier1 (3 secteurs)	Fic.2 (2 sect.)	Fichier3 (3 secteurs)	Fichier4 (4 secteurs)

Stockage d'un fichier physique

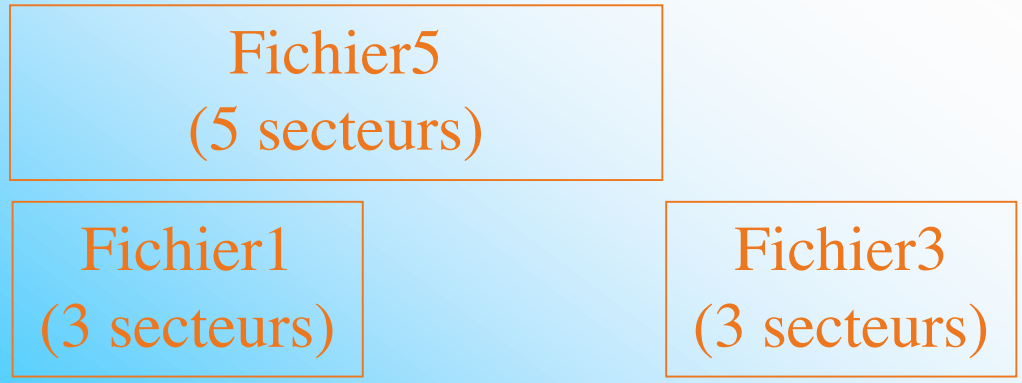
- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

Fichier1
(3 secteurs)

Fichier3
(3 secteurs)

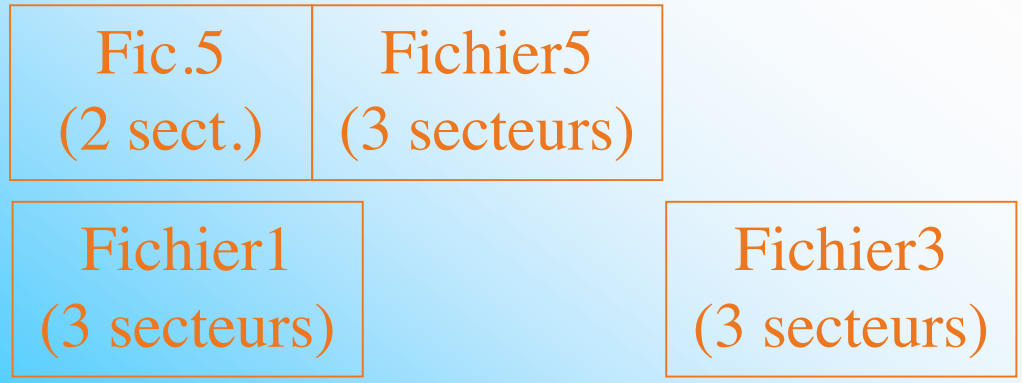
Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace



Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

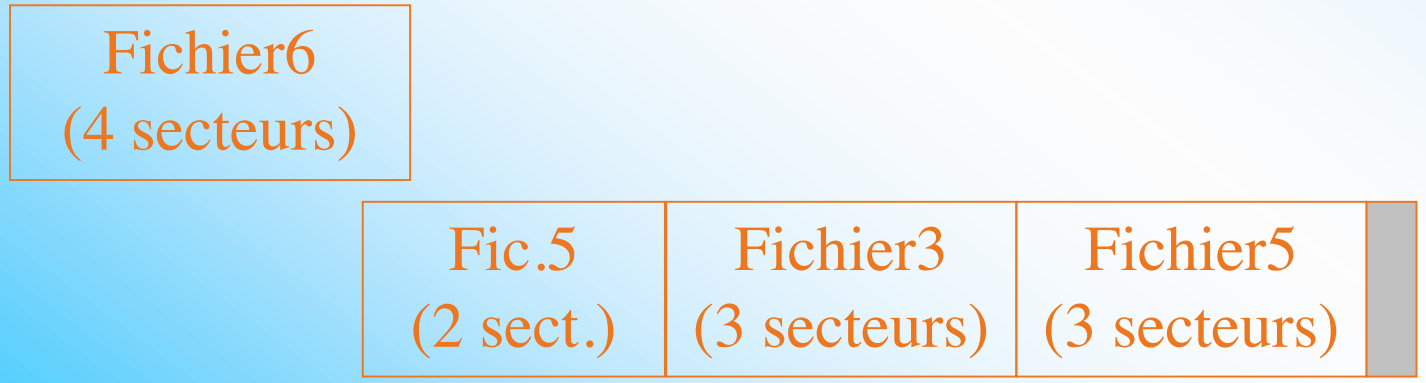


Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

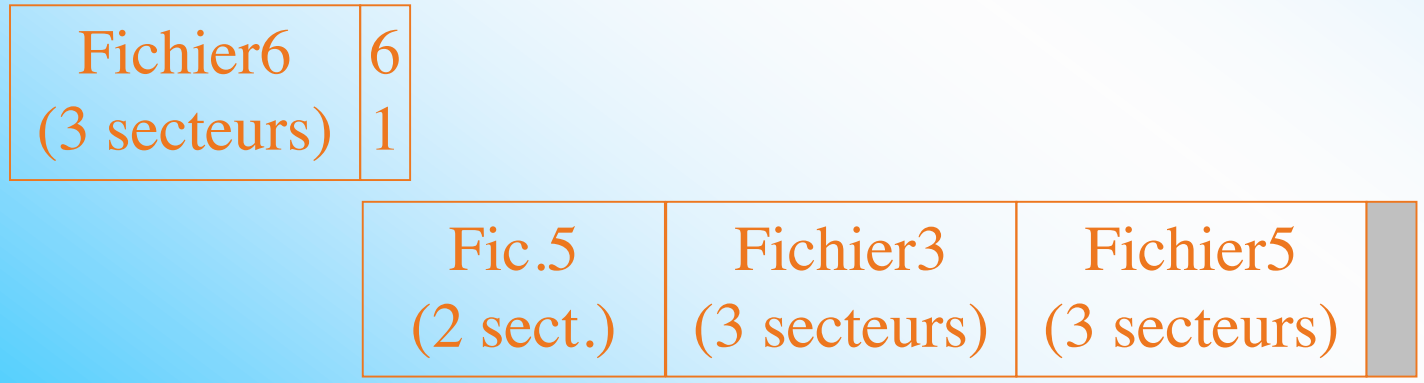
Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace



Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace



Stockage d'un fichier physique

- Conclusion du transparent précédent
  - Il faut pouvoir découper l'information en blocs
    - Les blocs doivent pouvoir être n'importe où sur le disque
    - *Fragmentation*
- Solution complexe
  - Découpage de l'information pour boucher les trous
    - Taille variable de blocs
    - Difficile à gérer
    - On risque de finir avec beaucoup de petits blocs
      - Pas efficace

Stockage d'un fichier physique

- Meilleure solution
  - Découpage en blocs de mêmes tailles
    - Gestion plus simple
      - Tous les éléments ont la même taille
    - Echanges simplifiés avec la mémoire
  - Quelle taille pour les blocs ?
    - Petite taille
      - Taux de remplissage excellent
      - Lenteur des transferts
    - Grande taille
      - Taux de remplissage moins bon
        - Le dernier bloc peut être très vide
      - Rapidité des transferts
    - Un compromis doit être trouvé

File Allocation Table - FAT

- Historique
  - Conçu par Bill Gates et Marc Mc Donald
    - Microsoft Disk Basic
  - Conçu pour des disquettes de 160 Kio
    - 1 Kio = 1 kibioctet = 2^{10} octets
  - Réutilisé dans QDOS
    - Ancêtre de MS-DOS
  - Utilisé dans MS-DOS
  - Utilisé dans les windows 9x
  - Toujours reconnu par les windows actuels
  - Reconnu par Linux
  - Utilisé par les cartes mémoires
    - Appareils photos, lecteurs MP3

Versions de FAT

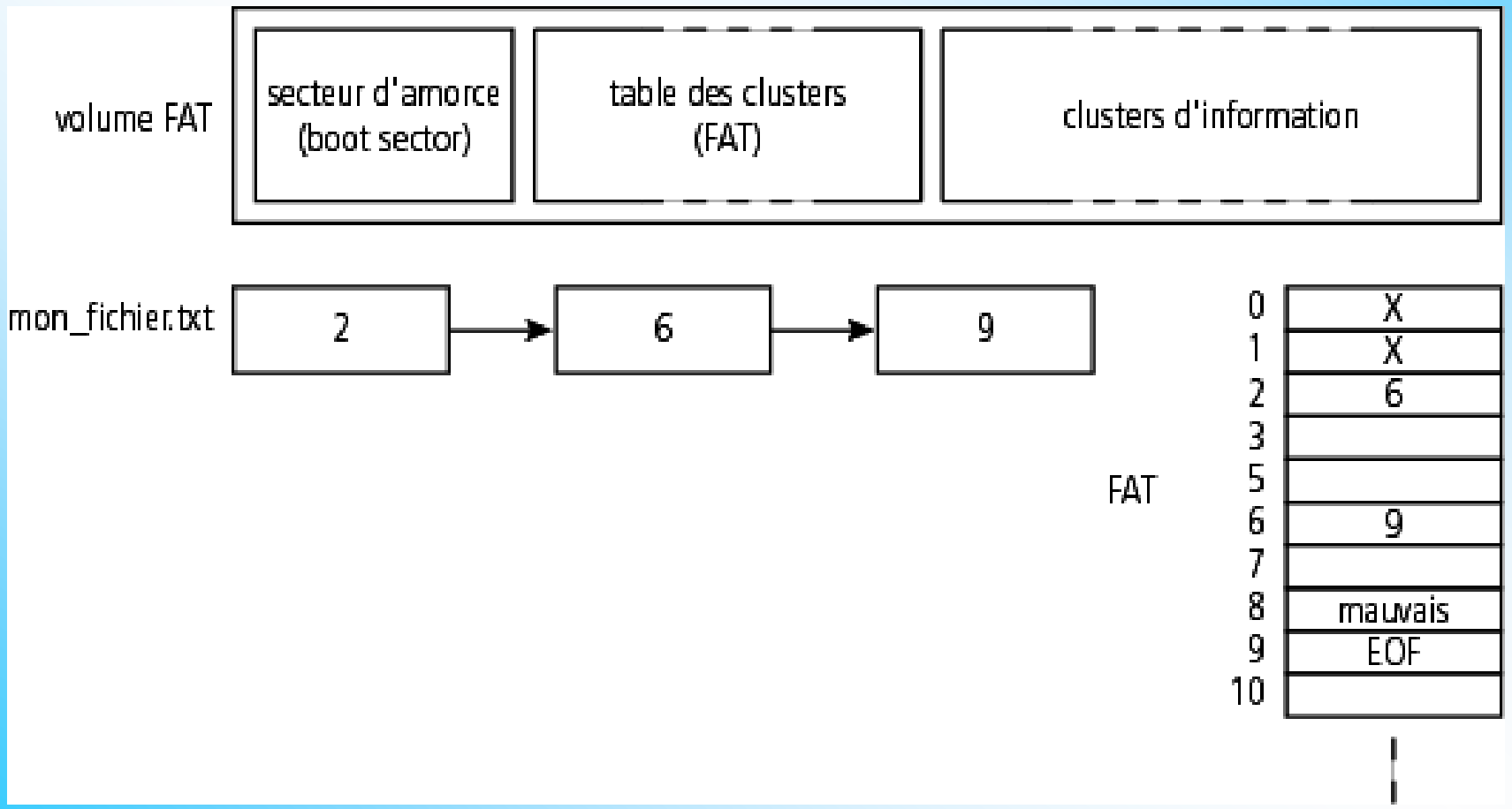
- FAT12
  - Max. 2^{12} = 4 096 clusters
    - 1 cluster fait entre 512 octets et 4 Kio
    - Utilisé sur les disquettes
- FAT16
  - Max. 2^{16} = 65 536 clusters
    - 1 cluster fait entre 2 Kio et 64 Kio
- VFAT
  - Evolution de la FAT qui permet la gestions des noms de 255 caractères
- FAT32
  - Max. 2^{28} soit 268 millions de clusters
    - 1 cluster fait entre 2 Kio et 32 Kio
    - Les 4 premiers bits sont utilisés comme drapeaux
    - Taille max. d'un fichier 4 Gio
    - Apparu avec Windows 95OSR2
- exFAT (ou FAT64)
  - Partitions théoriques de 64 Zio
    - 1 Zio = 1 zébioctet = 2^{70} octets
    - Apparu avec Windows VistaSP1

FAT

- Composé de 3 parties
  - Le BIOS Parameter Block (BPB)
    - Secteur d'amorçage
  - Une table d'allocation des fichiers
  - Clusters d'informations
    - Un cluster est composé d'un ensemble de secteurs
    - Equivalent au bloc sous Unix
    - Nombre de secteurs variable

FAT

Vue synthétique



Le système de fichiers FAT

- Un (sous-)répertoire = un fichier
  - suite d'enregistrements (entrée) de 32 octets
- Entrée de répertoire
  - nom (8 octets)
  - extension (3 octets)
  - caractéristiques du fichier (1 octet)
    - les drapeaux
  - heure et date (4 octets)
  - numéro d'entrée dans la FAT (2 octets)
  - taille (4 octets)
  - etc.
- Pour le détail, lire fatgen103.doc sur le serveur

ext2

- Aussi appelé ext2fs
- Système
  - Linux
  - Système de fichier historique de Linux
    - Second système de Linux après Extended File System
- Nouveauté
  - Limitation de la fragmentation du disque dur

ext2

- Caractéristiques
  - Taille max. de volume
    - 2 Tio - 32 Tio
      - 1 Tio = 1 tébioctet = 2^{40} octets
  - Taille max. d'un fichier
    - 16 Gio - 2 Tio
      - 1 Gio = 1 gibioctet = 2^{30} octets
  - Nombre max. de fichier
    - Variable
      - Configuration à l'installation du système de fichier
- Droits
  - Unix
  - Access Control List (ACL)

ext3

- Sytème
  - Linux
  - 1ère annonce d'un travail sur une amélioration de ext2
    - 17 février 1999
  - Version stable
    - Version 2.4.15 en novembre 2001
- Nouveauté
  - Journalisation
- Compatibilité
  - Une partition ext3 peut être utilisé comme une partition ext2
  - Utilitaires de maintenance de ext2 disponibles sous ext3
    - Avantage sur certains concurrents comme ReiserFS et XFS

ext3

- Comme ext2
- Caractéristiques
  - Taille max. de volume
    - 2 Tio - 32 Tio
      - 1 Tio = 1 tébioctet = 2^{40} octets
  - Taille max. d'un fichier
    - 16 Gio - 2 Tio
      - 1 Gio = 1 gibioctet = 2^{30} octets
  - Nombre max. de fichier
    - Variable
      - Configuration à l'installation du système de fichier
- Droits
  - Unix
  - Access Control List (ACL)

ext4

- Système : Linux
  - Rendu public le 10 octobre 2006
    - Version de linux 2.6.19
  - Version stable à partir de la version de linux 2.6.28
- Nouveauté
  - Allocation par extent
    - Allocation d'une zone de blocs contigus
      - Permet de poursuivre plusieurs écritures successives sur des blocs contigus
- Compatibilité
  - Une partition ext4 peut être montée comme ext3
    - A condition que l'allocation par extent ne soit pas utilisée
  - Compatibilité ascendante avec ext3
    - Une partition ext3 peut être montée comme ext4

ext4

- Caractéristiques
  - Taille max. de volume
    - 1 024 Pio
      - 1 Pio = 1 pébioctet = 2^{50} octets
      - 1 024 Pio = 1 Eio = 1 exbioctet = 2^{60} octets
  - Taille max. d'un fichier
    - 16 Tio
      - 1 Tio = 1 tébioctet = 2^{40} octets
  - Nombre max. de fichiers
    - 4 milliards
- Droits
  - POSIX

Amorce

Démarrage d'un ordinateur

Amorce

- Définition
  - Procédure de démarrage d'un ordinateur
- En anglais
  - *Booting, booting up, boot sequence*
  - *Initial Program Loading (APL)*
- Pourquoi boot ?
  - Vient de boot strap (courroie de botte)
    - Destinée à aider à enfiler une botte en cuir
- Autres termes
  - Amorçage, initialisation, démarrage
- Deux sortes
  - Démarrage à froid
    - On allume la machine
  - Démarrage à chaud (reboot)
    - Proposé par le système d'exploitation
    - On recharge le programme initial
    - La machine n'est pas éteinte

Principe

- Etape 1
  - On allume l'ordinateur
- Etape 2
  - L'ordinateur exécute un mini-programme cablé ou stocké en mémoire ROM ou flash
    - Bootstrap loader
    - Permet de déterminer quoi faire ensuite
    - Exemple : BIOS (voir plus loin)
- Etape 3
  - Chargement du système d'exploitation
    - Peut nécessiter plusieurs étapes

Historique - bootstrap

- Dans un premier temps
  - Le processeur était mis en attente
  - Un opérateur positionnait des interrupteurs
    - Ecriture de la 1ère instruction
  - Il appuyait sur un bouton
    - L'ordinateur lisait l'instruction et la stockait dans la 1ère case mémoire libre de la RAM
  - Il recommençait jusqu'à l'écriture du programme entier
  - Enfin, le processeur recevait l'ordre d'exécuter le programme
- Plus tard
  - Lecture d'une bande de papier perforé
    - Ressemblait à une lanière de botte
- Encore plus tard
  - Lecture d'une carte permettant de charger un programme lisant les cartes suivantes et les exécutant

De nos jours - bootstrap

- Exécution du programme d'amorçage
  - Bootstrap loader
  - Stocké en ROM
  - Possibilités de vérification du matériel
  - Lecture de données
    - Stockées en ROM
    - Permettent de déterminer les supports non volatiles pouvant contenir un système
      - Volume d'amorçage (ou bootable en jargon)
    - Généralement paramétrables
  - Analyse des volumes potentiellement amoçables
    - Sont-ils présents ou non ?
  - Sélection d'un volume
    - Souvent le premier de la liste à être présent
  - Chargement du boot loader (ou bootloader)
    - Stocké dans une zone particulière du volume
  - Lancement de l'exécution du bootloader

Chargement du boot loader

- Chargement en plusieurs étapes
  - Un premier boot loader est stocké dans une zone particulière du volume d'amorçage
    - Généralement petite
      - 446 octets pour un disque sur un PC
    - Utilité
      - Charger en mémoire le 2ème boot loader
        - Stocké sur le volume
  - Exécution du 2ème boot loader
    - Peut ne pas exister pour les boot loaders simples
      - Remplacé par un chargement du SE

Exécution du boot loader

- Utilité
  - Permet de lancer l'exécution d'un SE
  - Permet souvent de sélectionner un SE
    - *Multiboot*
    - Choix du SE
      - Windows, Linux…
    - Choix de la version du noyau
    - Choix d'une option de démarrage
      - Mode sans échec de Windows
    - Programmes ne nécessitant pas de SE
      - Testeurs de mémoires
      - Jeux
  - Un choix par défaut existe généralement
    - Lancement de ce choix au bout d'un certain temps
    - Ou apparition du menu sur une certaine séquence de touches

Exemples de boot loaders

- GNU GRUB
  - *GNU GRand Unified Bootloader*
  - Implémentation de référence de la "spécification du multiboot"
    - Standard ouvert de la Free Software Foundation
- LILO
  - *Linux Loader*
  - Initiallement conçu pour charger Linux
- BOOTMGR
  - *Windows Boot Manager*
  - Introduit avec Windows VISTA
- NTLDR
  - *NT Loader*
  - Introduit avec Windows NT, utilisé jusqu'à Windows XP et Windows server 2003

Cas particuliers

- Systèmes embarquées
  - Téléviseurs numériques, GPS…
  - L'amorçage normal est trop long
    - De l'ordre de la minute
  - Système directement en ROM ou mémoire flash
- Amorçage par réseau
  - Routeurs
  - Stations sans disques (cluster…)
  - Le firmware des machines actuelles permettent l'amorçage par le réseau
    - Sinon volume externe contenant le logiciel (clé USB…)
  - Déroulement
    - Recherche d'un serveur sur le réseau
      - Diffusion (broadcast) ou diffusion groupé (multicast)
    - Réception (d'une partie) du système d'exploitation

Amorçage du PC

- Boostrap
  - BIOS (Basic Input Output System)
  - Aussi appelé le *firmware*
  - Stocké en mémoire ROM ou flash
    - Il est actuellement possible de le mettre à jour
      - ⇒ Vulnérables à certains virus
  - Développé par le fabricant de la carte mère
  - Coreboot (anciennement LinuxBIOS)
    - BIOS libre sous licence GPL
    - Destiné à remplacer les BIOS propriétaires

Exécution du BIOS

- Exécution du *Power Good*
  - Vérification de la tension délivrée
    - Pour ne pas risquer d'abîmer le matériel
- Exécution du POST
  - Power-On-Self-Test
    - Auto-test au démarrage
  - Vérification du matériel et des périphériques
    - Affichage d'un message permettant l'accès au *setup*
      - Paramétrage de l'amorçage
- Activation des BIOS secondaires
  - Certaines cartes possèdent leur propre BIOS
    - Exemple : carte vidéo
- Affichage de la configuration rencontrée
- Lecture des 1er secteurs des périphériques amorçables
- Chargement du boot loader

Zone d'amorce

- En anglais
  - Master boot record (MBR)
- Localisation
  - Premier secteur adressable d'un disque dur
- Taille
  - 512 octets
- Contenu
  - Table des partitions
  - Routine d'amorçage (boot loader)
    - Si elle existe

Structure de la zone d'amorce

- Octets 0 à 444
  - Routine d'amorçage
    - Peut être plus courte que 444 octets
  - Les 4 derniers octets peuvent être utilisés pour une signature
- Octets 444 et 445
  - Habituellement nuls
- Octets 446 à 509
  - Table des partitions primaires
- Octets 510 et 511
  - Contiennent respectivement 55h et AAh
    - Nombre magique AA55h
    - Condition indispensable pour que le BIOS charge la routine d'amorçage en mémoire